Verification and Monitoring for First-Order LTL with Persistence-Preserving Quantification over Finite and Infinite Traces

Diego Calvanese[1,2] , **Giuseppe De Giacomo**[3] , **Marco Montali**[1] , **Fabio Patrizi**[3]
[1]Free University of Bozen-Bolzano, Italy
[2]Umeå University, Sweden
[3]Sapienza University of Rome, Italy
calvanese@inf.unibz.it, degiacomo@diag.uniroma1.it, montali@inf.unibz.it, patrizi@diag.uniroma1.it

Abstract

We address the problem of model checking first-order dynamic systems where new objects can be injected in the active domain during execution. Notable examples are systems induced by a first-order action theory expressed, e.g., in the situation calculus. Recent results show that, under state-boundedness, such systems, in spite of having a first-order representation of the state, admit decidable model checking for full first-order mu-calculus. However, interestingly, model checking remains undecidable in the case of first-order LTL (LTL-FO). In this paper, we show that in LTL-FO$_p$, the fragment of LTL-FO where quantification ranges only over objects that persist along traces, model checking state-bounded systems becomes decidable over infinite and finite traces. We then employ this result to show how to handle monitoring of LTL-FO$_p$ properties against a trace stemming from an unknown state-bounded dynamic system, simultaneously considering the finite trace up to the current point, and all its possibly infinite future continuations.

1 Introduction

Data-aware dynamic systems are, essentially, transition systems (TS) whose states are labelled by first-order (FO) interpretations, here referred to as *relational TSs* (RTS). Initially studied in database theory [Bhattacharya *et al.*, 2007; Deutsch *et al.*, 2009; Bagheri Hariri *et al.*, 2013b], the AI community has paid growing attention to these systems in the last decade, adopting them as models for logically-specified systems, e.g., action theories in the situation calculus [McCarthy and Hayes, 1969; Reiter, 2001], possibly operating over description-logic knowledge bases [Bagheri Hariri *et al.*, 2013a; Calvanese *et al.*, 2016; Borgwardt *et al.*, 2022]. Such interest concerns, in particular, verification [Calvanese *et al.*, 2018; Belardinelli *et al.*, 2014; Bagheri Hariri *et al.*, 2013b; De Giacomo *et al.*, 2012; De Giacomo *et al.*, 2016] and synthesis (typically, planning) [Baier and McIlraith, 2006; Calvanese *et al.*, 2016; Borgwardt *et al.*, 2022].

This paper focuses on verification of RTSs. Several temporal logics have been previously proposed, and in particular a FO extension of the propositional μ-calculus, the FO μ-calculus, for which decidability results have been obtained, under a condition of *state-boundedness* and *genericity* of the RTS, and syntactic restrictions on the logic which limit the power of quantifiers [Calvanese *et al.*, 2018; Bagheri Hariri *et al.*, 2013b]. State-boundedness holds when the interpretations labeling the states of an RTS contain only a bounded number of distinct objects; genericity essentially captures the inability of distinguishing isomorphic interpretations.

Previous works (see references above) provide a complete picture for branching-time logics (all virtually subsumed by FO μ-calculus) but are poorly conclusive for linear-time. In particular, Calvanese *et al.* [2018] show that verification is undecidable for the FO extension of linear-time temporal logic (LTL-FO) but do not investigate decidable fragments. This work completes the picture for LTL-FO interpreted over both infinite [Pnueli, 1977] and finite traces [De Giacomo and Vardi, 2013]. While, for concreteness, we assume RTSs are described by situation calculus BATs, our results can be applied also to other logical formalisms which guarantee genericity (and boundedness, when needed).

Our approach mirrors the one adopted by Calvanese *et al.* [2018]. Starting from the most expressive logic, LTL-FO, we consider two increasingly stricter fragments, LTL-FO$_a$ and LTL-FO$_p$, obtained by increasingly limiting the power of quantifiers: in LTL-FO$_a$, they range over *active* objects (i.e., occurring in the interpretation of some fluent); in LTL-FO$_p$, they range over active *persisting* objects (i.e., remaining active in consecutive states). After re-establishing undecidability of LTL-FO$_a$, thus of LTL-FO, with a proof which, differently from the one by Calvanese *et al.* [2018], isolates the source of undecidability, we investigate LTL-FO$_p$ and prove its decidability. Interestingly, LTL-FO$_a$ undecidability contrasts with the decidability of its branching-time counterpart $\mu\mathcal{L}_a$ (FO μ-calculus with quantification over active objects) proven by Calvanese *et al.* [2018].

Decidability of verification paves the way to *monitoring*, i.e., the problem of checking whether a trace stemming from the execution of an unknown dynamic system does or does not satisfy an LTL-FO property and/or may/will in the future. We show that, for LTL-FO$_p$, this can be done via a direct application of the obtained results, again under a *boundedness* assumption, naturally fulfilled in the context of *Process Mining* [van der Aalst, 2016], thus making the obtained results

interesting also in that context.

2 Relational Transition Systems

We recall notions and results by Calvanese *et al.* [2018]. A *(first-order) vocabulary* is a pair $\sigma = \langle \mathcal{F}, \mathcal{C} \rangle$, where $\mathcal{F}$ is the set of *predicate fluents* and $\mathcal{C}$ the set of *active (object) constants*. Given an interpretation (object) domain Δ with *standard names* s.t. $\mathcal{C} \subseteq \Delta$, we denote by Int^{σ}_{Δ} the set of all $\mathcal{F}$ and $\mathcal{C}$ interpretations over Δ that are the identity on every $c \in \mathcal{C}$.

Definition 1 (Relational Transition System, RTS)**.** Given a vocabulary $\sigma = \langle \mathcal{F}, \mathcal{C} \rangle$ and an object domain Δ s.t. $\mathcal{C} \subseteq \Delta$, an RTS (over σ and Δ) is a tuple $\mathcal{T} = \langle \Delta, S, s_0, \rightarrow, \mathcal{I} \rangle$, s.t.: *(i)* S is a set of *states*; *(ii)* $s_0 \in S$ is the *initial state*; *(iii)* $\rightarrow \subseteq S \times S$ is a *transition relation*; and *(iv)* $\mathcal{I} : S \mapsto Int^{\sigma}_{\Delta}$ is a labeling function associating each state $s \in S$ with an interpretation $\mathcal{I}(s) = \langle \Delta, \cdot^{\mathcal{I}(s)} \rangle$ (s.t. $c^{\mathcal{I}(s)} = c$, $c \in \mathcal{C}$). ◁

For an RTS $\mathcal{T} = \langle \Delta, S, s_0, \rightarrow, \mathcal{I} \rangle$, a *$\mathcal{T}$-run* is a (possibly infinite) sequence $\rho = s_0\, s_1 \cdots$ of states, s.t.: *(i)* s_0 is $\mathcal{T}$'s initial state, and *(ii)* $s_i \rightarrow s_{i+1}$, for all s_i, s_{i+1} in ρ. A *trace* over Int^{σ}_{Δ} is a possibly infinite sequence $\tau = \mathcal{I}_0\, \mathcal{I}_1 \cdots$ s.t. each $\mathcal{I}_i \in Int^{\sigma}_{\Delta}$. By $\tau(i)$ we denote $\mathcal{I}_i$ and by $|\tau|$ the *length* of τ, i.e., the (possibly infinite) number of FO interpretations in τ. A *trace of* $\mathcal{T}$ is a trace $\tau = \mathcal{I}_0\, \mathcal{I}_1 \cdots$ over Int^{σ}_{Δ} s.t. $\mathcal{I}_i = \mathcal{I}(s_i)$, for some run $\rho = s_0\, s_1 \cdots$; in this case, we also say that ρ *induces* τ. A *maximal $\mathcal{T}$-run* is either an infinite run or a run $s_0 \cdots s_\ell$ s.t. for no state $s_{\ell+1} \in S$ it holds that $s_\ell \rightarrow s_{\ell+1}$. A trace of $\mathcal{T}$ is *maximal* if it is induced by a maximal $\mathcal{T}$-run.

We denote by $adom(\mathcal{I}(s))$ the *active domain* of $\mathcal{I}(s)$, i.e., the set of objects occurring in the interpretation of some fluent in s, plus the constants in $\mathcal{C}$. By $\tilde{\mathcal{I}}(s) = \langle adom(\mathcal{I}(s)), \cdot^{\mathcal{I}(s)} \rangle$, we denote the restriction of $\mathcal{I}(s)$ to its active domain. Two 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 vocabulary $\sigma = \langle \mathcal{F}, \mathcal{C} \rangle$ are said to be *isomorphic*, written $\mathcal{I}_1 \sim \mathcal{I}_2$, if there exists a bijection (called *isomorphism*) $h : \Delta_1 \mapsto \Delta_2$ s.t.: *(i)* for every $F \in \mathcal{F}$, it holds that $\vec{o} \in F^{\mathcal{I}_1}$ if and only if $h(\vec{o}) \in F^{\mathcal{I}_2}$; and *(ii)* for every $c \in \mathcal{C}$, $c^{\mathcal{I}_2} = h(c^{\mathcal{I}_1})$. Intuitively, isomorphic interpretations are the same modulo renaming of the objects in the interpretation domain. When needed, we write $\mathcal{I}_1 \sim_h \mathcal{I}_2$, to make the isomorphism h between $\mathcal{I}_1$ and $\mathcal{I}_2$ explicit. We can now define *generic RTSs*.

Definition 2 (Generic RTS)**.** An RTS $\mathcal{T} = \langle \Delta, S, s_0, \rightarrow, \mathcal{I} \rangle$ is *generic* if: for every $s_1, s_1', s_2 \in S$ and every bijection $h : \Delta \mapsto \Delta$, if $\mathcal{I}(s_1) \sim_h \mathcal{I}(s_2)$ and $s_1 \rightarrow s_1'$, then there exists $s_2' \in S$ such that $s_2 \rightarrow s_2'$ and $\mathcal{I}(s_1') \sim_h \mathcal{I}(s_2')$. ◁

Intuitively, this says that if s_1 and s_2 have the same interpretations modulo renaming, then every successor s_1' of s_1 has the same interpretation, modulo renaming, of a successor s_2' of s_2, and vice-versa (as states are quantified universally and isomorphisms are invertible). That is, if two states are isomorphic, they induce the same transitions modulo object renaming. As discussed by Calvanese *et al.* [2018], genericity arises when transitions stem from logical specifications, such as in the *situation calculus*, when no extra-logical properties are involved that require predefined domains and relations (e.g., integers and their order).

Next we introduce *state-bounded* RTSs, i.e., RTSs containing a bounded number of objects in the active domain of states, a property that turned out essential for the verification of various temporal logics [Belardinelli *et al.*, 2014; Bagheri Hariri *et al.*, 2013b; De Giacomo *et al.*, 2012; De Giacomo *et al.*, 2016; Calvanese *et al.*, 2018].

Definition 3 (State-Bounded RTS)**.** An RTS $\mathcal{T} = \langle \Delta, S, s_0, \rightarrow, \mathcal{I} \rangle$ is *(state-)bounded by* $b \in \mathbb{N}$ if $|adom(\mathcal{I}(s))| \leq b$, for every $s \in S$. We say that $\mathcal{T}$ is *state-bounded* if it is state-bounded by b for some b. ◁

Notice that state-boundedness allows for the occurrence of infinitely many objects along a run, although only boundedly many at each state. This is a semantic property that naturally arises in situations of practical interest [De Giacomo *et al.*, 2016; Bagheri Hariri *et al.*, 2014].

RTSs can be specified with several formalisms, such as the *situation calculus* [De Giacomo *et al.*, 2016; Calvanese *et al.*, 2018], which we use here; the obtained results, however, can be applied to any other formalisms for representing RTSs. We briefly recall the situation calculus, referring the reader to the book by Reiter [2001] for a more detailed introduction.

The *situation calculus* includes terms from three disjoint sorts: *objects*, i.e., entities in the domain of interest; *actions*, i.e., events that trigger world changes; and *situations*, i.e., sequences of actions applied in the situation resulting from previous applications. The constant S_0 denotes the initial situation (no action performed) and the function symbol do allows for building situation terms: $do(a, s)$ denotes the situation resulting from the execution of action a in situation s. We assume countably infinitely many object constants under the *unique name assumption* (UNA, distinct constants denote distinct objects), but do not assume domain closure for objects (not all objects are denoted by a constant).

Fluents are *predicates* and *functions* whose value depends on the situation. The situation is, by convention, the last argument, as in $Holding(o, s)$ (object o is being held in situation s). W.l.o.g., we assume no functions other than constants and no non-fluent predicates. All (but the last) fluent arguments are of sort object. Finally, we assume a finite number of *action types*, each taking a tuple of objects as arguments.

Situation calculus *(action) theories* describe dynamic domains. We focus on *basic action theories (BATs)*. A BAT $\mathcal{D}$ is the union of the following disjoint sets of first-order (FO) and second-order (SO) axioms:

1. $\mathcal{D}_0$: (FO) *initial situation description axioms* describing the initial configuration (may or may not be complete);
2. $\mathcal{D}_{poss}$: one (FO) *precondition axiom* $Poss(A(\vec{x}), s) \equiv \phi_A(\vec{x}, s)$ per action type A, stating the precondition $\phi_A(\vec{x}, s)$ for legal execution of action $A(\vec{x})$ in situation s, with special predicate $Poss(a, s)$ expressing that action a is executable in s and with $\phi_A(\vec{x}, s)$ a situation calculus formula *uniform in* s [Reiter, 2001], i.e., mentioning only s as a situation term and not mentioning $Poss$;
3. $\mathcal{D}_{ssa}$: one (FO) *successor state axiom* $F(\vec{x}, do(a, s)) \equiv \phi_F(\vec{x}, a, s)$ per fluent F, describing how F changes

when action a is executed in s, with $\phi_F(\vec{x}, a, s)$ uniform in s;
4. $\mathcal{D}_{ca}$: (FO) unique name axioms for actions and (FO) domain closure on action types;
5. $\mathcal{D}_{uno}$: (FO) unique name axioms for object constants;
6. Σ: (SO) foundational, domain independent, axioms of the situation calculus [Reiter, 2001].

A situation s is *executable* if every action performed in reaching s is executable in the situation where it is performed.

Consider a model M of a BAT $\mathcal{D}$, with object domain Δ and executable situation domain $\mathcal{S}$. Intuitively, this is obtained by fixing an interpretation of the initial situation (which is unique, modulo object renaming, if the description is complete), and then using precondition and successor-state axioms to generate the model. Let $\mathcal{C}$ be the set of constants mentioned in $\mathcal{D}$ and assume, w.l.o.g., they all occur in $\mathcal{D}_0$. For every situation $s \in \mathcal{S}$, define the FO interpretation $\mathcal{I}_M(s) \doteq \langle \Delta, \cdot^{\mathcal{I}} \rangle$, where: *(i)* for every $c \in \mathcal{C}$, $c^{\mathcal{I}} = c^M$ and *(ii)* for every (situation-suppressed) fluent F of $\mathcal{D}$, $F^{\mathcal{I}} = \{\vec{o} \mid \langle \vec{o}, s \rangle \in F^M\}$. The RTS *induced by* M [Calvanese *et al.*, 2018] is defined as the RTS $\mathcal{T}_M = \langle \Delta, S, s_0, \mathcal{I}, \rightarrow \rangle$, where: *(i)* $S = \mathcal{S}$; *(ii)* $s_0 = S_0^M \in S$; *(iii)* $\rightarrow \subseteq S \times S$ is s.t. $s \rightarrow s'$ iff there exists some action a s.t. $\langle a, s \rangle \in Poss^M$ and $s' = do^M(a, s)$; *(iv)* For every $s \in \mathcal{S}$, $\mathcal{I}(s) = \mathcal{I}_M(s)$. Intuitively, $\mathcal{T}_M$ is the tree of M's executable situations with each situations s labelled by an interpretation of fluents (and constants), corresponding to the interpretation that M associates to s. As such, $\mathcal{T}_M$ contains, in general, infinitely many states (i.e., as many as the executable situations of $\mathcal{D}$). Notice that transitions do not carry information about actions. BATs enjoy the following key result.

Theorem 1 (Calvanese *et al.* 2018)**.** *For every model M of a BAT $\mathcal{D}$, the induced RTS $\mathcal{T}_M$ is generic.*

This result states that every RTS induced by a BAT $\mathcal{D}$ (through its models) is generic. Unfortunately, an analogous result does not hold for the other critical property of RTS, i.e., *state-boundedness*. To capture this, De Giacomo *et al.* [2016] introduce a semantical definition. For a situation s, let $Executable(s)$ be a situation calculus formula expressing that s is executable. Given a model M of a BAT $\mathcal{D}$ and a situation s, let $Bounded_b(s)$ be a situation calculus formula expressing that the extensions of all fluents at s according to M contain, overall, at most $b \in \mathbb{N}$ distinct objects (in the original definition, the bound was on the number of tuples for each fluent). Then, a BAT $\mathcal{D}$ is said to be *bounded*, if all of its models M yield that all executable situations are bounded.

Definition 4 (Bounded BAT [De Giacomo *et al.*, 2016])**.** A BAT $\mathcal{D}$ is said to be *bounded* by $b \in \mathbb{N}$, if:

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

The following is an immediate consequence of Definition 4.

Theorem 2. *If $\mathcal{D}$ is a BAT bounded by b, then every model M of $\mathcal{D}$ induces a state-bounded RTS $\mathcal{T}_M$.*

3 Linear-time Temporal Logic FO Variants

We focus on languages that combine linear-time and FO properties, obtaining a hierarchy that mirrors, in the linear-time setting, that of FO μ-calculi studied by Calvanese *et al.* [2018]. In the literature, the semantics of linear-time temporal (LTL) logics is typically provided in two alternative variants, over either infinite traces [Pnueli, 1977; Baier and Katoen, 2008] or finite traces [Baier and McIlraith, 2006; De Giacomo and Vardi, 2013]. We adopt both at once. For a FO vocabulary $\langle \mathcal{F}, \mathcal{C} \rangle$, LTL-FO (FO linear-time logic) is as follows:

$$\varphi ::= \top \mid \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists x.\varphi \mid \mathbf{X}\,\varphi \mid \varphi_1\,\mathbf{U}\,\varphi_2,$$

where ϕ is an atomic FO formula $F(t_1, \ldots, t_r)$, with $F \in \mathcal{F}$ and each t_i a variable or a constant from $\mathcal{C}$. Formulae where all variables are quantified, which are the only ones of interest here, are said to be *closed*. We use the standard abbreviations: *(i)* $\bot = \neg\top$, *(ii)* $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, *(iii)* $\varphi \supset \psi = \neg\varphi \vee \psi$, *(iv)* $\forall x.\varphi = \neg\exists x.\neg\varphi$, *(v)* $\mathbf{F}\,\varphi = \top\,\mathbf{U}\,\varphi$, *(vi)* $\mathbf{G}\,\varphi = \neg\,\mathbf{F}\,\neg\varphi$.

LTL-FO formulae are interpreted over finite or infinite traces over Int^{σ}_{Δ}, for some domain Δ, with all constants in $\mathcal{C}$ interpreted rigidly, and equality interpreted as the identity. The interpretation is relative to a position $i \geq 0$ and a valuation v mapping variables into Δ. By $v[x/o]$ we denote a valuation that matches v on all variables except x, with $v[x/o](x) = o$. We inductively define when *a trace τ satisfies an* LTL-FO *formula φ at position $i \geq 0$ under valuation* v, written $(\tau, i, v) \models \varphi$, as follows ($|\tau| = \infty$ for inifinite τ):

1. $(\tau, i, v) \models \top$, if $0 \leq i < |\tau|$;
2. $(\tau, i, v) \models \phi$, if $0 \leq i < |\tau|$ and $(\tau(i), v) \models \phi$;
3. $(\tau, i, v) \models \neg\varphi$, if $(\tau, i, v) \not\models \varphi$;
4. $(\tau, i, v) \models \varphi_1 \wedge \varphi_2$, if $(\tau, i, v) \models \varphi_1$ and $(\tau, i, v) \models \varphi_2$;
5. $(\tau, i, v) \models \exists x.\varphi$, if there exists $o \in \Delta^{\tau(i)}$ s.t. $(\tau, i, v[x/o]) \models \varphi$;
6. $(\tau, i, v) \models \mathbf{X}\,\varphi$, if $0 \leq i < |\tau| - 1$ and $(\tau, i+1, v) \models \varphi$;
7. $(\tau, i, v) \models \varphi_1\,\mathbf{U}\,\varphi_2$, if:
   - there exists $i \leq k < |\tau|$ s.t. $(\tau, k, v) \models \varphi_2$, and
   - for every j, if $i \leq j < k$ then $(\tau, j, v) \models \varphi_1$;

If φ is closed, we omit v and simply write $(\tau, i) \models \varphi$. Observe that the *next* operator $\mathbf{X}$ is interpreted as *strong next*, which requires existence of a successor state in τ. This is important if τ is finite and irrelevant otherwise. If needed, the *weak next* operator $\mathbf{X_w}$, which does not require existence of the successor state, can be expressed as $\mathbf{X_w}\,\varphi = \neg\,\mathbf{X}\,\neg\varphi$.

As we shall see, the central problem of this paper, i.e., checking whether the maximal traces of an RTS satisfy an LTL-FO formula, is undecidable. Thus, the need arises for identifying decidable classes. Following the same approach as the one by Bagheri Hariri *et al.* [2013b] and Calvanese *et al.* [2018], by imposing syntactic restrictions, we define two notable fragments of LTL-FO. The fragment LTL-FO$_a$ is as follows:

$$\varphi ::= \top \mid \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists x.\textsc{live}(x) \wedge \varphi \mid \mathbf{X}\,\varphi \mid \varphi_1\,\mathbf{U}\,\varphi_2,$$

where LIVE is a FO-definable predicate denoting membership of x to the active domain, defined as an abbreviation for the disjunction $\bigvee P(\ldots, x, \ldots)$ over all predicates $P \in \mathcal{F}$ and all positions of x in $P(\ldots, x, \ldots)$. Observe that quantification in LTL-FO$_a$ is allowed only on objects in the active domain. For instance, the logic cannot express that some object not occurring in the current state will occur in a future state.

Example 1. The following LTL-FO$_a$ formula expresses that along a trace, it is always the case that if for some *live* object x, $P(x)$ holds, then eventually there exists a live object y s.t. $Q(x, y)$ holds:

$$\varphi_a = \mathbf{G}(\forall x.(\textsc{live}(x) \wedge P(x)) \supset (\mathbf{F}\, \exists y.\textsc{live}(y) \wedge Q(x, y))). \quad \triangleleft$$

The other fragment of interest is LTL-FO$_p$, which restricts quantification to *persisting* objects, i.e., occurring in the active domain of consecutive states. For example, one cannot say that some object enters and leaves the active domain in different states. Formulae of LTL-FO$_p$ are defined as follows, where $\vec{x}$ and $\vec{y}$ can have common variables:

$$\varphi ::= \top \mid \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists x.\textsc{live}(x) \wedge \varphi \mid \textsc{live}(\vec{x}) \wedge \mathbf{X}\, \varphi(\vec{x}) \mid (\textsc{live}(\vec{x}, \vec{y}) \wedge \varphi_1(\vec{x}))\, \mathbf{U}\, \varphi_2(\vec{y}).$$

Example 2. The following LTL-FO$_p$ formula expresses a property analogous to that of φ_a in Example 1, except that it requires x to *persist* in the active domain, until a state is reached where there exists y s.t. $Q(x, y)$ holds:

$$\varphi_{p1} = \mathbf{G}(\forall x.(\textsc{live}(x) \wedge P(x)) \supset (\textsc{live}(x)\, \mathbf{U}\, \exists y.\textsc{live}(y) \wedge Q(x, y))).$$

Observe that φ_a, instead, does not require x to persist, i.e., x can leave and re-enter the active domain along the trace (although it must eventually show up to make $Q(x, y)$ true).$\triangleleft$

Notice that LTL-FO$_p$ can express properties over objects that are present in the current state but disappear in the next one, while it cannot predicate over the same object in non-adjacent states, unless that object is asserted to persist in between.

Example 3. Formula $\exists x.\textsc{live}(x) \wedge \mathbf{X}\, \neg\textsc{live}(x)$ is a legal LTL-FO$_p$ formula expressing that an object in the current active domain disappears next. Instead, the LTL-FO$_a$ formula $\exists x.\textsc{live}(x) \wedge \mathbf{X}\, \mathbf{X}\, \neg\textsc{live}(x)$ (with two unrestricted nested strong next operators) is not a legal LTL-FO$_p$ formula. $\triangleleft$

Finally, observe that the case of the until operator $\mathbf{U}$ works analogously to that of strong next. In fact, formulae of the form $(\textsc{live}(\vec{x}, \vec{y}) \wedge \varphi_1(\vec{x}))\, \mathbf{U}\, \varphi_2(\vec{y})$ require objects $\vec{x}$ and $\vec{y}$ to persist in the active domain in all states from the current one up to the one immediately preceding the state in which $\varphi_2(\vec{y})$ holds.

4 Model Checking

We now focus on *model checking* (or verification) of RTSs against LTL-FO. For concreteness, we assume RTSs are specified by BATs (and fixed object domains), but the obtained results can be applied to any logical formalism that yields genericity and boundedness (when needed) in the induced RTS. In order to specify RTSs, we use BATs with a complete initial situation description (i.e., s.t. $\mathcal{D}_0$ admits exactly one model up to object renaming), interpreted over a given object domain Δ. This implies that the BAT $\mathcal{D}$ has a unique model (up to object renaming), thus defines a unique RTS.

Given an RTS $\mathcal{T}$ and an LTL-FO formula φ, we say that $\mathcal{T}$ *satisfies* φ, written $\mathcal{T} \models \varphi$, if $\tau \models \varphi$, for all *maximal* traces τ of $\mathcal{T}$. Model checking φ over $\mathcal{T}$ consists in checking whether $\mathcal{T} \models \varphi$. Observe that, since LTL-FO can express trace finiteness with formula $\varphi_{fin} = \mathbf{F} \neg \mathbf{X} \top$ (there exists a point with no successor), one can specialize model checking to consider either finite or infinite traces only. This leads to the definitions of *finite* and *infinite* model checking, which respectively amount to checking whether $\mathcal{T} \models \varphi_{fin} \supset \varphi$ and $\mathcal{T} \models \varphi_{fin} \vee \varphi$.

4.1 Undecidability

Unsurprisingly, verification is undecidable in general, even for RTSs specified by severely restricted BATs and formulae.

Theorem 3. *Model checking of a formula $\mathbf{F}\, p$, with p a proposition, is undecidable over RTSs of the form $\mathcal{T} = \langle \Delta, S, s_0, \rightarrow, \mathcal{I} \rangle$ induced by BATs over Δ with complete initial situation description and two unary fluents.*

Proof (Sketch). By reduction from non-emptiness of deterministic 2-counter Minsky automata [Minsky, 1967], which is undecidable. A BAT $\mathcal{D}$ that models a generic counter automaton $\mathcal{A}$ can be written s.t.: *(i)* each counter is modeled by a unary fluent; *(ii)* the number of objects in each fluent represents the value of the counters; *(iii)* the automaton states are modeled as propositions, with all counters null in the initial state and the final state denoted by special proposition p; *(iv)* BAT actions simulate increment, conditional decrement, and zero-testing over counters. It can be proven that $\mathcal{A}$ is non-empty iff $\mathcal{T} \models \mathbf{F}\, p$, for $\mathcal{T}$ the (unique) RTS induced by $\mathcal{D}$. $\square$

Notice that Theorem 3 seamlessly applies also to finite and infinite model checking. In all cases, undecidability is ultimately due to the possibility of storing unboundedly many objects in fluents, which allows for modeling counters. It is natural to ask oneself whether limiting this ability by requiring *state-boundedness* [Belardinelli *et al.*, 2012; Bagheri Hariri *et al.*, 2013b] yields decidability, as is the case in the branching-time setting for FO μ-calculus [Calvanese *et al.*, 2018]. Unfortunately, this is not the case: model checking for LTL-FO$_a$, thus LTL-FO, is undecidable even for state-bounded BATs. This is proven by Calvanese *et al.* [2018, Thm. 18] by a reduction from satisfiability of LTL with freeze-quantifiers over infinite data-words and using one register only [Demri and Lazic, 2009]. With a minimal variation, that reduction can be lifted to encode satisfiability of LTL with freeze-quantifiers over finite data-words, which is also undecidable, but requires two registers [Demri and Lazic, 2009]. This leads us to the following undecidability result, which holds in particular for finite model checking.

Theorem 4. *There exists 1-bounded BAT $\mathcal{D}$ with complete initial situation description and two unary fluents P_1, P_2 s.t. finite model checking an LTL-FO$_a$ formula using P_1 and P_2 is undecidable over the RTS $\mathcal{T} = \langle \Delta, S, s_0, \rightarrow, \mathcal{I} \rangle$ induced by $\mathcal{D}$ over Δ.*

Calvanese *et al.* [2018, Thm. 18] and Theorem 4 do not help in singling out the essential features of LTL-FO$_a$ that cause undecidability, as encoding LTL with freeze-quantifiers requires to employ the whole LTL-FO$_a$ logic, with the only restriction that atomic FO formulae are unary. For infinite model checking, we can actually identify a more restricted,

informative undecidable fragment of LTL-FO$_a$. To this end, we propose an alternative proof that recasts the proof by Demri and Lazic [2009, Thm. 5.2] into our setting. This allows us to show undecidability of infinite model checking for a fixed 1-bounded BAT already for the fragment of LTL-FO$_a$ consisting of unary recurrence formulae, defined next.

Unary recurrence formulae are LTL-FO$_a$ formulae as below, where p, q_1, q_2, q_3 are propositional formulae, P is a unary fluent, both occurrences of $\bigodot$ denote either $\mathbf{X}\,\mathbf{F}$ or $\mathbf{F}$, and $[\cdot]$ denotes optionality (observe that $P(x)$ implies $\text{LIVE}(x)$):

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{X}\,\varphi \mid \varphi_1\,\mathbf{U}\,\varphi_2 \mid \mathbf{G}(\forall x.(P(x) \wedge q_1) \supset [\neg]\bigodot(q_2 \wedge \bigodot(P(x) \wedge q_3))).$$

Theorem 5. *There exists 1-bounded BAT $\mathcal{D}$ with complete initial situation description and a unary fluent P s.t. infinite model checking unary recurrence formula using P is undecidable over the RTS $\mathcal{T} = \langle \Delta, S, s_0, \to, \mathcal{I}\rangle$ induced by $\mathcal{D}$ over Δ.*

Proof (Sketch). By reduction from non-emptiness of counter automata with incrementing errors over infinite executions, which is undecidable even though these automata are weaker than Minsky automata [Demri and Lazic, 2009]. $\mathcal{D}$ over-approximates the infinite executions of all counter machines with incrementing errors, operating over a given set of control states. Unary recurrence formulae are used to isolate the executions that encode those of a given automaton, and to express its non-emptiness. □

4.2 Persistence-Preserving Trace Equivalence

Unary recurrence formulae exploit the essential feature of LTL-FO$_a$ to quantify over objects across states. In this section, we investigate whether limiting further this ability, as done in LTL-FO$_p$, which restricts quantification to persisting objects only, is beneficial for obtaining decidability. This will turn out to be the case.

We start by introducing a notion of equivalence among traces which captures the intuition that two traces match modulo renaming of the active objects wrt every pair of consecutive states. In other words, only persisting objects are required to maintain their renaming.

Definition 5. Two traces τ_1 and τ_2 over σ and Δ are *persistence-preserving equivalent* (*p-equivalent*) under a sequence $\eta = h_0\, h_1 \cdots$ of bijections, written $\tau_1 \asymp^p_\eta \tau_2$, if:

1. for every $i \geq 0$, we have $\tau_1(i) \sim_{h_i} \tau_2(i)$;
2. for every $i > 0$, the restriction $h_{i+1}|_{adom(\tau_1(i))}$ of h_{i+1} to $adom(\tau_1(i))$ coincides with $h_i|_{adom(\tau_1(i))}$. ◁

We write $\tau_1 \asymp^p \tau_2$ if τ_1 and τ_2 are p-equivalent under some sequence of bijections. It is immediate to see that $\asymp^p$ is an equivalence relation.

P-equivalence preserves the identity of an object across adjacent states only if that object is present in both of them. This implies that those objects that persist along a sequence of pairwise adjacent states also preserve their identity throughout that sequence. At the same time, p-equivalence allows us to single out objects in the current active domain that may disappear in the next one, as well as objects that appear in the next active domain but are not present in the current one. However, p-equivalence cannot distinguish an object that is not present in the current state and appears in the next one from any other object exhibiting the same pattern.

This leads to an interesting property of p-equivalence, namely that it cannot distinguish between *globally* and *locally* fresh objects, in the sense formalized next. Let the *historical active domain* at position i of τ, denoted $hadom_\tau(i)$, be the set $hadom_\tau(i) = \bigcup_{j\in\{0,\ldots,i\}} adom(\tau(j))$ of objects that have been active in some interpretation at position $j \leq i$. The set of *globally fresh objects* at position i of τ is $\Delta \setminus hadom_\tau(i)$. The set of *locally fresh objects* at position i of τ is $\Delta \setminus adom(\tau(i))$. While a locally fresh object at position i might have occurred in some position $j < i-1$, this cannot be the case for globally fresh objects. The following remark, a direct consequence of Definition 5, shows that $\asymp^p$ does not distinguish global and local freshness. Hence, past active objects that have been forgotten can be "recycled" and used as if globally fresh.

Remark 1. Consider two finite traces τ_1 and τ_2 of length n and such that $\tau_1 \asymp^p_\eta \tau_2$. Let τ_1' and τ_2' be the extensions of such traces with one further interpretation each. If there is a bijection h s.t.: *(i)* $\tau_1'(n) \sim_h \tau_2'(n)$; *(ii)* $h(o) = \eta(n-1)(o)$ for $o \in adom(\tau_1'(n-1))$, and $h(o) \in \Delta \setminus adom(\tau_2'(n-1))$ for $o \in \Delta \setminus hadom_{\tau_1'}(n-1)$, then $\tau_1' \asymp^p_{\eta h} \tau_2'$. ◁

Example 4. Consider the following traces:

$$\tau_1 = \tau_2 = \{P(o_1), P(o_2)\}\ \{P(o_1), Q(o_2)\}\ \{R(o_1)\}.$$

Let τ_1' be the extension of τ_1 with $\{P(o_3)\}$, and τ_2' the extension of τ_2 with $\{P(o_2)\}$. Since τ_1' and τ_2' match but in the last position, o_3 is globally fresh at position 2 of τ_1, and o_2 is locally fresh at position 2 of τ_2, by Remark 1, $\tau_1' \asymp^p \tau_2'$. ◁

We now connect the notion of p-equivalence to the notion of *persistence-preserving bisimulation* (p-bisimulation) studied by Bagheri Hariri *et al.* [2013b] and Calvanese *et al.* [2018]. This will in turn prove essential when studying p-equivalence of traces induced by RTSs.

Definition 6 (Calvanese *et al.* 2018)**.** Let $\mathcal{T}_1 = \langle \Delta_1, S_1, s_{10}, \to_1, \mathcal{I}_1\rangle$ and $\mathcal{T}_2 = \langle \Delta_2, S_2, s_{20}, \to_2, \mathcal{I}_2\rangle$ be two RTSs over the same vocabulary, and let H be the set of all possible bijections $h : D_1 \mapsto D_2$, for all possible $D_1 \subseteq \Delta_1$ and $D_2 \subseteq \Delta_2$. A relation $R \subseteq S_1 \times H \times S_2$ is a *p-bisimulation* between $\mathcal{T}_1$ and $\mathcal{T}_2$, if $\langle s_1, h, s_2\rangle \in R$ implies that:

1. $\tilde{\mathcal{I}}_1(s_1) \sim_h \tilde{\mathcal{I}}_2(s_2)$;
2. for each $s_1' \in S_1$, if $s_1 \to_1 s_1'$, there exists $s_2' \in S_2$ s..t.:
   (a) $s_2 \to_2 s_2'$, and
   (b) there exists a bijection $h' : adom(\mathcal{I}_1(s_1)) \cup adom(\mathcal{I}_1(s_1')) \mapsto adom(\mathcal{I}_2(s_2)) \cup adom(\mathcal{I}_2(s_2'))$ s.t its restriction $h'|_{adom(\mathcal{I}_1(s_1))}$ coincides with $h|_{adom(\mathcal{I}_1(s_1))}$ and $\langle s_1', h'|_{adom(\mathcal{I}_1(s_1'))}, s_2'\rangle \in R$;
3. for each $s_2' \in S_2$, if $s_2 \to_2 s_2'$, there exists $s_1' \in S_1$ s.t.:
   (a) $s_1 \to_1 s_1'$, and
   (b) there exists a bijection $h' : adom(\mathcal{I}_1(s_1)) \cup adom(\mathcal{I}_1(s_1')) \mapsto adom(\mathcal{I}_2(s_2)) \cup adom(\mathcal{I}_2(s_2'))$ s.t its restriction $h'|_{adom(\mathcal{I}_1(s_1))}$ coincides with $h|_{adom(\mathcal{I}_1(s_1))}$ and $\langle s_1', h'|_{adom(\mathcal{I}_1(s_1'))}, s_2'\rangle \in R$. ◁

A state $s_1 \in S_1$ is *p-bisimilar* to $s_2 \in S_2$, written $s_1 \approx^p s_2$, if there exists a p-bisimulation R between $\mathcal{T}_1$ and $\mathcal{T}_2$ such that $\langle s_1, h, s_2 \rangle \in R$, for some h; when needed, we also write $s_1 \approx^p_h s_2$, to explicitly name h. Finally, $\mathcal{T}_1$ and $\mathcal{T}_2$ are *p-bisimilar*, written $T_1 \approx^p T_2$, if $s_{10} \approx^p s_{20}$. P-bisimilarity is obviously an equivalence relation.

We are now ready to relate p-equivalence to p-bisimilarity and to the LTL-FO$_p$ logic.

Lemma 1. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two RTSs s.t. $\mathcal{T}_1 \approx^p \mathcal{T}_2$. Then, for every maximal trace τ_1 of $\mathcal{T}_1$ there exists a maximal trace τ_2 of $\mathcal{T}_2$ s.t. $\tau_1 \asymp^p \tau_2$.*

Theorem 6. *Let τ_1 and τ_2 be traces s.t. $\tau_1 \asymp^p \tau_2$. Then, for every* LTL-FO$_p$ *formula φ, we have $\tau_1 \models \varphi$ iff $\tau_2 \models \varphi$.*

By Theorem 6 and Remark 1 we get that LTL-FO$_p$ cannot capture global freshness, and by combining Lemma 1 with Theorem 6, we obtain the following key result.

Theorem 7. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two RTSs s.t $\mathcal{T}_1 \approx^p \mathcal{T}_2$. Then, for every* LTL-FO$_p$ *formula φ, we have $\mathcal{T}_1 \models \varphi$ iff $\mathcal{T}_2 \models \varphi$.*

4.3 Decidability

We are now ready to address decidability of model checking. Using Theorem 7, we can import in our setting the data abstraction techniques studied by Bagheri Hariri *et al.* [2013a] and De Giacomo *et al.* [2016] for state-bounded, generic RTSs. For such systems, it is shown that for every RTS $\mathcal{T}$ there exists a p-bisimilar, finite-state $\mathcal{T}'$, referred to as an *abstraction* of $\mathcal{T}$. Interestingly, $\mathcal{T}'$ can be obtained by recycling, as much as possible, locally fresh objects whenever fresh ones are needed; the bound on the active domain of each state guarantees that boundedly many old objects are enough to ensure that no further globally fresh objects are needed, in turn guaranteeing that the resulting abstraction RTS is finite-state.

Theorem 8 (De Giacomo *et al.* 2016)**.** *For every generic and state-bounded RTS $\mathcal{T} = \langle \Delta, S, s_0, \rightarrow, \mathcal{I} \rangle$ over a vocabulary σ, there exists a finite-state and state-bounded RTS $\mathcal{T}' = \langle \Delta', S', s'_0, \rightarrow', \mathcal{I}' \rangle$ over σ with finite Δ', s.t. $\mathcal{T} \approx^p \mathcal{T}'$.*

Observe that since the abstract RTS $\mathcal{T}'$ of Theorem 8 is finite-state and state-bounded, $\mathcal{T}'$ is essentially a transition system with propositional labeling. While it can be shown that checking whether $\mathcal{T}' \models \varphi$, for φ an LTL-FO$_p$ formula, is decidable, this does not imply decidability of $\mathcal{T} \models \varphi$, as Theorem 8 does not state effective computability of $\mathcal{T}'$. To obtain such a desirable result, we need to take advantage of the (logical) representation of $\mathcal{T}$, in our case as a BAT. Specifically, let $\mathcal{D}$ be the BAT used, together with the infinite object domain Δ, to specify $\mathcal{T}$. De Giacomo *et al.* [2016] show that, in order to achieve decidability, it is enough to replace the infinite object domain Δ with a suitable finite subset $\Delta' \subset \Delta$ and then use this and the BAT $\mathcal{D}$ to specify $\mathcal{T}'$. Under this condition, one can effectively compute the desired $\mathcal{T}'$, thus obtaining decidability of model checking.

Theorem 9 (De Giacomo *et al.* 2016)**.** *Let $\mathcal{D}$ be a BAT bounded by b with complete initial situation description, M its model (unique, up to object renaming) for an infinite object domain Δ, and $\mathcal{T}_M$ the corresponding induced RTS. Moreover, let n be the maximal number of parameters occurring in the action types of $\mathcal{D}$. Then, for any possibly finite Δ' s.t. $|\Delta'| \geq 2b + n + |\mathcal{C}|$, one can effectively compute a finite-state RTS $\mathcal{T}'$ s.t. $\mathcal{T}_M \approx^p \mathcal{T}'$.*

Intuitively, the construction of $\mathcal{T}'$ consists in constructing a finite fragment of M, starting from the initial situation and expanding all situations currently in the fragment by applying all executable actions. State-finiteness is obtained by including a new situation only if the corresponding interpretation does not match any other of some situation already in the fragment. By finiteness of Δ' and boundedness of $\mathcal{D}$, the procedure terminates, as there exist only finitely many distinct interpretations. $\mathcal{T}'$ is then easily obtained from the constructed fragment. An analogous result can be obtained with some care, also for BATs with incomplete initial situation description [De Giacomo *et al.*, 2016]. This leads to the following main result, which, we stress once again, holds for every logical formalism other than the situation calculus, as long as it guarantees genericity of the modeled RTS.

Theorem 10. *Model checking* LTL-FO$_p$ *formulae over state-bounded, generic RTSs specified by BATs is decidable.*

5 Monitoring

Monitoring amounts to verifying whether the ongoing execution of an unknown dynamic system satisfies a property of interest, considering also its possible future continuations [Bauer *et al.*, 2010; Ly *et al.*, 2015]. A 4-valued semantics, called RV-LTL, has been introduced by Bauer *et al.* [2010], recast by De Giacomo *et al.* [2014] over finite traces, and employed by De Giacomo *et al.* [2020] for reinforcement learning.

We lift RV-LTL to our FO setting. The current (finite) execution is modeled as a finite trace τ. For simplicity, we consider as possible execution continuations only finite, unbounded extensions of τ, but our results hold also for infinite extensions. Let $RV = \{\text{CS}, \text{PS}, \text{CV}, \text{PV}\}$ be the set of 4 RV-LTL states, with the following meaning: (*i)* CS (*current satisfaction*): (formula) satisfied now but violable in some future; (*ii)* PS (*permanent satisfaction*): satisfied now and in all futures; (*iii)* CV (*current violation*): violated now but satisfiable in some future; and (*iv)* PV (*permanent violation*): violated now and in all futures. This is formalized below.

Definition 7. Given a finite trace τ, an LTL-FO formula φ over σ, and an interpretation domain Δ, we say that φ *is in monitoring state $s \in RV$ after τ*, written $\tau \models [\![\varphi = s]\!]$, if:

- $s = \text{CS}$, $\tau \models \varphi$, and $\tau\tau' \not\models \varphi$, for some finite trace τ';
- $s = \text{PS}$, $\tau \models \varphi$, and $\tau\tau' \models \varphi$, for every finite trace τ';
- $s = \text{CV}$, $\tau \not\models \varphi$, and $\tau\tau' \models \varphi$ for some finite trace τ';
- $s = \text{PV}$, $\tau \not\models \varphi$, and $\tau\tau' \not\models \varphi$ for every finite trace τ'. $\triangleleft$

It is immediate to see that, for a finite trace τ and an LTL-FO formula φ, there exists exactly one $s \in RV$ such that $\tau \models [\![\varphi = s]\!]$. LTL-FO *monitoring* consists in computing the monitoring state $s \in RV$ s.t. $\tau \models [\![\varphi = s]\!]$.

By Definition 7 it follows that monitoring is at least as hard as LTL-FO satisfiability and validity. To see this, it is enough to apply Definition 7 when τ is the empty trace. As a consequence, we can obtain for monitoring a strong undecidability result analogous to Theorem 4. To this end, we first need

to adapt to traces the notion of state-boundedness for RTSs (Definition 3).

Definition 8. A trace τ is *(state-)bounded by* $b \in \mathbb{N}$ if $|adom(\tau(i))| \leq b$, for $0 \leq i < |\tau|$. We say that τ is *(state-)bounded* if it is bounded by b for some b. ◁

Theorem 11. LTL-FO$_a$ *monitoring over 1-bounded traces is undecidable.*

Again, however, analogously to the case of model checking, we obtain decidability for state-bounded traces and LTL-FO$_p$.

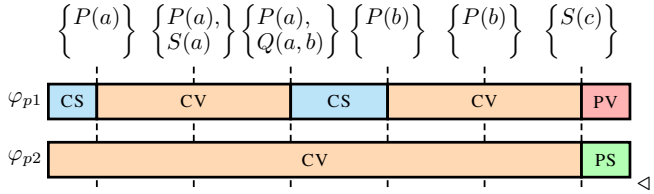
Theorem 12. LTL-FO$_p$ *monitoring over state-bounded traces is decidable.*

Proof (Sketch). Consider a finite trace τ bounded by b and an LTL-FO$_p$ formula φ. The proof consists in characterizing the RV-LTL states through two verification problems on φ or $\neg\varphi$, based on the state under consideration. Checking whether $\tau \models \varphi$ or $\tau \models \neg\varphi$ can be done via LTL-FO$_p$ model checking, by replacing τ with a trivial generic state-bounded RTS with τ as unique trace, which is decidable by Theorem 10. For the extensions of τ, we define a BAT $\mathcal{D}$ state-bounded by b, with a unique model M s.t. the sequence of interpretations assigned by M to each path in the tree of executable situations corresponds either to a prefix or a possible b-bounded extension of τ, over the given vocabulary σ. We then check whether $\mathcal{D} \models \varphi_{fin} \supset \varphi$, using φ_{fin} to restrict to finite extensions (recall that φ_{fin} expresses trace finiteness), which is decidable by Theorem 10. □

Example 5. Consider formula φ_{p1} from Example 2, and the following formula φ_{p2} expressing that at some point in the trace, there must be a live object x for which $P(x)$ holds and then, as long as x persists, $S(x)$ should never hold:

$$\varphi_{p2} = \mathbf{F}(\exists x.(\textsc{live}(x) \wedge P(x) \wedge (\textsc{live}(x) \wedge \neg S(x)) \,\mathbf{U}\, \neg\textsc{live}(x))).$$

The table below shows the result of the monitoring of φ_{p1} and φ_{p2} for the following trace:



◁

Bounded traces are not only induced by state-bounded systems, but naturally arise also in the context of *Process Mining* (PM) [van der Aalst, 2016]. An integral part of PM deals with operational support for running operational processes instantiated on case objects, such as orders, packages, or claims. Such objects spawn executions, producing traces whose positions contain logged *events*, which are essentially tuples of instantiated attributes. Since each position in a case trace contains exactly one logged event, every trace is state-bounded by the maximum number of attributes of the events occurring therein. Hence, by Theorem 12, LTL-FO$_p$ can be effectively used to monitor operational processes, a main task in operational support [van der Aalst, 2016; Ly *et al.*, 2015]. We can also extend the application of our technique in this setting by considering a *bounded set* of *related cases*, using LTL-FO$_p$ to express so-called instance spanning constraints [Fdhila *et al.*, 2016].

6 Conclusions

We have addressed the problem of verification of FO extensions of temporal logics in the linear-time setting, for both infinite and finite traces, left almost unexplored by previous work, which focused mainly on branching-time. We have shown that, for generic and state-bounded RTSs, verification of LTL-FO$_p$ is decidable, whereas it is undecidable for LTL-FO$_a$. The latter contrasts with a previous result for branching-time, which states that restricting quantification to the active domain yields decidability. To obtain our results, we have isolated the undecidability source for the most expressive logic LTL-FO, a point left unclear so far, and then have investigated the increasingly stricter fragments LTL-FO$_a$ and LTL-FO$_p$. While our work focuses on RTSs specified as situation calculus basic action theories, the obtained results seamlessly apply to all formalisms that induce generic RTSs.

The decidability result opens the way to monitoring for LTL-FO$_p$, a problem of practical interest naturally arising in Process Mining, which consists in checking whether the ongoing execution of an unknown system currently satisfies an LTL-FO$_p$ formula and/or will do in the future, as the execution carries on. We have recasted the problem in our setting and shown its decidability under state-boundedness.

As future work, we intend to develop automata-based techniques for verification and monitoring of LTL-FO$_p$. To this end, thanks to the abstraction techniques exploited here, we can directly import algorithmic approaches based on FO extensions of LTL interpreted over fixed, finite domains, such as those by Baier and McIlraith [2006] and by De Masellis and Su [2013].

Acknowledgements

This work is partly supported by the ERC Advanced Grant WhiteMech (No. 834228), the EU ICT-48 2020 project TAILOR (No. 952215), the Sapienza Project DRAPE, the Italian Basic Research (PRIN) projects HOPE and PINPOINT, the unibz ID 2020 project SMART-APP, and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[Bagheri Hariri *et al.*, 2013a] Babak Bagheri Hariri, Diego Calvanese, Giuseppe De Giacomo, Riccardo De Masellis, Paolo Felli, and Marco Montali. Description logic knowledge and action bases. *J. of Artificial Intelligence Research*, 46:651–686, 2013.

[Bagheri Hariri *et al.*, 2013b] Babak Bagheri Hariri, Diego Calvanese, Giuseppe De Giacomo, Alin Deutsch, and Marco Montali. Verification of relational data-centric dynamic systems with external services. In *Proc. of the 32nd ACM Symp. on Principles of Database Systems (PODS)*, pages 163–174, 2013.

[Bagheri Hariri *et al.*, 2014] Babak Bagheri Hariri, Diego Calvanese, Alin Deutsch, and Marco Montali. State-boundedness in data-aware dynamic systems. In *Proc. of the 14th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*. AAAI Press, 2014.

[Baier and Katoen, 2008] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[Baier and McIlraith, 2006] Jorge A. Baier and Sheila A. McIlraith. Planning with first-order temporally extended goals using heuristic search. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI)*, pages 788–795. AAAI Press, 2006.

[Bauer *et al.*, 2010] Andreas Bauer, Martin Leucker, and Christian Schallhart. Comparing LTL semantics for runtime verification. *J. of Logic and Computation*, 20(3):651–674, 2010.

[Belardinelli *et al.*, 2012] Francesco Belardinelli, Alessio Lomuscio, and Fabio Patrizi. An abstraction technique for the verification of artifact-centric systems. In *Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, pages 319–328, 2012.

[Belardinelli *et al.*, 2014] Francesco Belardinelli, Alessio Lomuscio, and Fabio Patrizi. Verification of agent-based artifact systems. *J. of Artificial Intelligence Research*, 51:333–376, 2014.

[Bhattacharya *et al.*, 2007] K. Bhattacharya, N. S. Caswell, S. Kumaran, A. Nigam, and F. Y. Wu. Artifact-centered operational modeling: Lessons from customer engagements. *IBM Systems J.*, 46(4):703–721, 2007.

[Borgwardt *et al.*, 2022] Stefan Borgwardt, Jörg Hoffmann, Alisa Kovtunova, Markus Krötzsch, Bernhard Nebel, and Marcel Steinmetz. Expressivity of planning with Horn description logic ontologies. In *Proc. of the 36th AAAI Conf. on Artificial Intelligence (AAAI)*, 2022.

[Calvanese *et al.*, 2016] Diego Calvanese, Marco Montali, Fabio Patrizi, and Michele Stawowy. Plan synthesis for knowledge and action bases. In *Proc. of the 25th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1022–1029. AAAI Press, 2016.

[Calvanese *et al.*, 2018] Diego Calvanese, Giuseppe De Giacomo, Marco Montali, and Fabio Patrizi. First-order μ-calculus over generic transition systems and applications to the situation calculus. *Information and Computation*, 259(3):328–347, 2018.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 854–860. IJCAI/AAAI, 2013.

[De Giacomo *et al.*, 2012] Giuseppe De Giacomo, Yves Lesperance, and Fabio Patrizi. Bounded situation calculus action theories and decidable verification. In *Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, pages 467–477, 2012.

[De Giacomo *et al.*, 2014] Giuseppe De Giacomo, Riccardo De Masellis, Marco Grasso, Fabrizio Maria Maggi, and Marco Montali. Monitoring business metaconstraints based on LTL and LDL for finite traces. In *Proc. of the 12th Int. Conf. on Business Process Management (BPM)*, volume 8659 of *LNCS*, pages 1–17. Springer, 2014.

[De Giacomo *et al.*, 2016] Giuseppe De Giacomo, Yves Lesperance, and Fabio Patrizi. Bounded situation calculus action theories. *Artificial Intelligence*, 237:172–203, 2016.

[De Giacomo *et al.*, 2020] Giuseppe De Giacomo, Marco Favorito, Luca Iocchi, Fabio Patrizi, and Alessandro Ronca. Temporal logic monitoring rewards via transducers. In *Proc. of the 17th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, pages 860–870, 2020.

[De Masellis and Su, 2013] Riccardo De Masellis and Jianwen Su. Runtime enforcement of first-order LTL properties on data-aware business processes. In *Proc. of the 11th Int. Joint Conf. on Service Oriented Computing (ICSOC)*, volume 8274 of *LNCS*, pages 54–68. Springer, 2013.

[Demri and Lazic, 2009] Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. on Computational Logic*, 10(3), 2009.

[Deutsch *et al.*, 2009] Alin Deutsch, Richard Hull, Fabio Patrizi, and Victor Vianu. Automatic verification of data-centric business processes. In *Proc. of the 12th Int. Conf. on Database Theory (ICDT)*, pages 252–267, 2009.

[Fdhila *et al.*, 2016] Walid Fdhila, Manuel Gall, Stefanie Rinderle-Ma, Juergen Mangler, and Conrad Indiono. Classification and formalization of instance-spanning constraints in process-driven applications. In *Proc. of the 14th Int. Conf. on Business Process Management (BPM)*, volume 9850 of *LNCS*, pages 348–364. Springer, 2016.

[Ly *et al.*, 2015] Linh Thao Ly, Fabrizio Maria Maggi, Marco Montali, Stefanie Rinderle-Ma, and Wil M. P. van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Information Systems*, 54:209–234, 2015.

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

[Minsky, 1967] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *Proc. of the 18th Annual Symp. on the Foundations of Computer Science (FOCS)*, pages 46–57. IEEE, 1977.

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

[van der Aalst, 2016] Wil M. P. van der Aalst. *Process Mining – Data Science in Action*. Springer, 2nd edition, 2016.