Information and Computation 272 (2020) 104502

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

SMT-based satisfiability of first-order LTL with event freezing functions and metric operators [☆]

Alessandro Cimatti, Alberto Griggio, Enrico Magnago, Marco Roveri, Stefano Tonetta [*]

FBK-irst, via Sommarive 18, Trento, Italy

ARTICLE INFO

Article history:
Received 27 February 2018
Received in revised form 15 February 2019
Accepted 15 February 2019
Available online 10 December 2019

Keywords:
First-order linear-time temporal logic
Metric temporal logic
SMT-based model checking
Temporal satisfiability

ABSTRACT

In this paper, we propose to extend First-Order Linear-time Temporal Logic with Past adding two operators “at next” and “at last”, which take in input a term and a formula and return the value of the term at the next state in the future or last state in the past in which the formula holds. The new logic, named *LTL-EF*, can be interpreted with different models of time (including discrete, dense, and super-dense time) and with different first-order theories (à la Satisfiability Modulo Theories (SMT)). We show that the “at next” and “at last” can encode (first-order) $MTL_{0,\infty}$ with counting. We provide rewriting procedures to reduce the satisfiability problem to the discrete-time case (to leverage on the mature state-of-the-art corresponding verification techniques) and to remove the extra functional symbols. We implemented these techniques in the nuXmv model checker enabling the analysis of *LTL-EF* and $MTL_{0,\infty}$ based on SMT-based model checking. We show the feasibility of the approach experimenting with several non-trivial valid and satisfiable formulas.

1. Introduction

In formal verification, the quest for a suitable formal specification language that provides the right trade-off among expressiveness, usability, and automated reasoning, is always a difficult task. In the context of real-time embedded systems, the functional specification of input/output variables is often entangled with timing aspects. Moreover, when a system or component is seen as a black box, the properties must be specified in terms of the observable variables or messages exchanged with the system environment. This is, for example, the case of properties of monitors (which trigger alarms based on some condition on the observed variables/messages) or contract-based specifications (which formalize the assumptions/guarantees of components independently of the implementation). In these cases, the properties must be able to express the relationship between the observable variables at different points of time, without referring to internal system variables that store the corresponding values. It is therefore necessary to have suitable mechanisms to refer to the value of variables at different points of time.

[☆] This paper extends the work presented in [1] with a translation from $MTL_{0,\infty}$ with counting operators and with a more robust tool implementation and evaluation. This work has received funding from the European Union’s Horizon 2020 research and innovation program under the Grant Agreement No. 700665 (project CITADEL).

[*] Corresponding author.

E-mail addresses: cimatti@fbk.eu (A. Cimatti), griggio@fbk.eu (A. Griggio), magnago@fbk.eu (E. Magnago), roveri@fbk.eu (M. Roveri), tonettas@fbk.eu (S. Tonetta).

https://doi.org/10.1016/j.ic.2019.104502

One of the most popular logics used in computer science to specify properties for formal verification is Linear-time Temporal Logic (*LTL*) [2]. The temporal domain is typically discrete and models are discrete, linear sequences of states. In the case of real-time systems, *LTL* is interpreted over a dense or super-dense models of time [3]. In particular, in super-dense time models, the temporal domain combines dense sets of real time points with (discrete) sequences of instantaneous time points, as needed for example for asynchronous real-time systems. When considering real models of time, it becomes natural to have constraints on the time elapsing between different events. Therefore, *LTL* has been extended either with clocks/freezing operators, as in *TPTL* [3], or with metric operators as in [4–7]. These extensions have been combined with first-order logic to represent message passing [8] or for monitoring specification [9].

In this paper, we consider First-Order *LTL* [10] with future as well as past operators [11]. The system state is described by individual variables, and first-order functions/predicates express their relationship. In the spirit of Satisfiability Modulo Theories (SMT) [12], the formulas are interpreted modulo a background first-order theory as in [13] (here, we restrict to a quantifier-free fragment where all the symbols in the signature are rigid).

We propose a new logic that extends the quantifier-free fragment of First-Order Linear-Time Temporal Logic with Past operators by adding the "at next" $u@\tilde{F}(\phi)$ and the "at last" $u@\tilde{P}(\phi)$ function symbols, which are used (resp.) to represent the value of a term u at the next state in the future or at the last state in the past in which the formula ϕ holds. For example, the formula $G(alarm \leftrightarrow x@\tilde{P}(read) = (x@\tilde{P}(read))@\tilde{P}(read))$ says that *alarm* is true iff in the last two points in which *read* was true the variable x had the same value. We refer to this extension of *LTL* as *LTL* with Event-freezing Functions (*LTL-EF*). As for *LTL*, it can be interpreted over different temporal domains (including discrete, dense, and super-dense time) and with different first-order theories (à la SMT). The "at next" and "at last" functions can be seen as a generalization of Event-Clock Temporal Logic (*ECTL*) operators [14,15,6] (which, in turn, are the logical counterpart of event clocks [16]) and can encode the fragment $MTL_{0,\infty}$ [16] of Metric Temporal Logic (*MTL*) [4], also with counting [17,18].

We provide rewriting procedures to reduce the satisfiability problem of *LTL-EF* to the discrete-time case without the extra function symbols. This reduction enables for the use of all the mature and state-of-the-art satisfiability checking techniques developed for the discrete-time case (see e.g. [19,20]). We implemented these techniques in the nuXmv model checker, which provides efficient SMT-based model checking procedures to verify temporal properties on systems described with first-order formulas (as discussed for instance in [19,20]). This provides an effective tool support for the analysis of *LTL-EF* and $MTL_{0,\infty}$.

The main contributions of the paper are the following. First, we identify an extension of *LTL* that can express interesting properties relating variables at different points of time. Second, we define the new operators in a very rich setting that includes first-order constraints, past operators, dense and super-dense semantics; this gives also a uniform treatment of *LTL* satisfiability modulo theories in the case of real-time models. Third, we show how $MTL_{0,\infty}$ formulas (extended with counting) can be expressed in *LTL-EF*. Fourth, we implemented the approach in a mature model checker, showing that it can effectively prove interesting properties, while many logics in the real-time setting lack tool support.

The rest of the paper is organized as follows: Section 2 provides an analysis of the related work. Section 3 introduces the considered temporal domains, the logics *LTL* and $MTL_{0,\infty}$, and their satisfiability modulo theories; Section 4 defines the extension with the new event freezing functions; Section 5 details how to translate an $MTL_{0,\infty}$ formula into an equivalent *LTL-EF* formula; Section 6 shows how to translate an *LTL-EF* formula over super-dense time into an equisatisfiable one over discrete time; Section 7 describes how to translate an *LTL-EF* formula over discrete time into an equisatisfiable one in *LTL*; Section 8 presents the experimental results; finally, Section 9 concludes the paper and draws directions for future work.

2. Related work

The most natural alternative to the new operators proposed in this paper would be to use registers [21] or freezing quantifiers [3]. *LTL-EF* adopts a more declarative style with functions that directly return the value of variables at the next or last state in which a formula will be/was true, resulting in a more natural specification. Despite the fact that freezing quantifiers provide a higher expressiveness (also with respect to *MTL* [22]), they are not so common in industrial applications (at least compared to *LTL* and *MTL*), either because they are less intuitive to use, or because they lack tool support.

The "at next" and "at last" function symbols of *LTL-EF* can be seen as a generalization of the Event-Clock Temporal Logic (*ECTL*) operators [14,15,6] and can be used to encode the fragment $MTL_{0,\infty}$ [16] of Metric Temporal Logic (*MTL*) [4] also with counting [17,18]. We remark that the *ECTL* operators are the logical counterpart of event clocks [16]. Our semantics for *LTL-EF* is close to the one defined in [15] for event clocks in Event-Clock Timed Automata and for the corresponding quantifiers in the equally-expressive monadic logic. However, differently from [15], we do not require the use of any nonstandard real number, which is instead used in [15] to handle operators with left-open intervals.

The rewriting we adopted to reduce the satisfiability problem for *LTL-EF* to the discrete time case [2] is inspired by the one described in [23]: we also split the time evolution into a sequence of singular or open intervals in such a way that an execution trace is fine for the input formula on such intervals. However, the considered temporal logic is different and the discretization of formulas differs even for the common fragment: in fact, in this paper, we consider a standard semantics of *LTL* where time points are quantified over a dense or super-dense temporal domain, while in [23] the semantics of temporal operators quantifies over sequences of intervals.

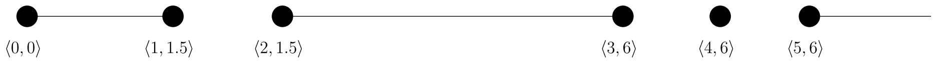

Fig. 1. Illustration of a super-dense time model (from [29]).

Regarding tool support, we are not aware of any other tool that is able to prove the validity of $MTL_{0,\infty}$ with parametric bounds in the case of dense or super-dense time, while instead our implementation in nuXmv supports also first-order constraints. We are aware of only three related tools, namely MigthyL [24], Zot [25], and ATMOC [26,27]. However, MigthyL uses a (discrete) pointwise semantics for *MTL*, while Zot and ATMOC only support bounded satisfiability of *MTL* properties.

3. Background

In this section, we provide the background material on which the later technical sections are based. We begin with a formal description of the temporal domains we consider (§3.1), and then proceed to introducing the syntax and semantics of first-order Linear Temporal Logic (§3.2 and §3.4) and Metric Temporal Logic (§3.6), making explicit the assumptions on traces on which we rely for our results (§3.5).

3.1. Time models

$\mathbb{R}_0^+$is the set of non-negative real numbers. A time interval is a convex subset of $\mathbb{R}_0^+$. The left endpoint of an interval I is denoted by $l(I)$, while the right endpoint by $r(I)$. Two intervals I and I' are almost adjacent iff $r(I) = l(I')$ (so they may overlap in at most one point). A singular interval is an interval in the form $[a, a]$ for some $a \in \mathbb{R}_0^+$. A time interval sequence is a sequence $I_0, I_1, I_2, \ldots$ of time intervals such that, for all $i \geq 0$, I_i and I_{i+1} are almost adjacent and $\bigcup_{i\geq 0} I_i = \mathbb{R}_0^+$.

We consider different models of time [28,3]. A time model is a structure $\tau = \langle T, <, \mathbf{0}, \nu\rangle$ with a temporal domain T, a total order $<$ over T, a minimum element $\mathbf{0} \in T$, and a function $\nu : T \to \mathbb{R}_0^+$ that represents the real time of a time point in T. The ν function is used instead of a distance (e.g., as in [4]) to treat the weakly-monotonic case in a more uniform way. A *time point* is an element of T. In particular, we consider the following models:

- discrete time model, where $T = \mathbb{N}$, $\mathbf{0}$ and $<$ are the standard zero and order over natural numbers, $\nu(0) = 0$ and $\nu(0), \nu(1), \nu(2), \ldots$ is a non-decreasing divergent sequence (this is also called the pointwise semantics; we use these models also for discrete-time *LTL* ignoring these real-time timestamps);
- dense (strictly-monotonic) time model, where $T = \mathbb{R}_0^+$, $\mathbf{0}$ and $<$ are the standard zero and order over the real numbers, and ν is the identity function;
- super-dense (weakly-monotonic) time model, where 1) $T \subset \mathbb{N} \times \mathbb{R}_0^+$ such that the sequence of sets $I_0, I_1, I_2, \ldots$ where, for all $i \geq 0$, the set $I_i := \{r \mid \langle i, r\rangle \in T\}$, is a time interval sequence (thus subsequent intervals can overlap in at most one point), 2) $\langle i, r\rangle < \langle i', r'\rangle$ iff $i < i'$ or $i = i'$ and $r < r'$, 3) $\mathbf{0} = \langle 0, 0\rangle \in \mathbb{N} \times \mathbb{R}_0^+$, and 4) $\nu(\langle i, r\rangle) = r$.

Intuitively, super-dense time models consist of alternating discrete sequences of points in the form $\{\langle i, r\rangle\}, \{\langle i+1, r\rangle, \ldots$ (with the same timestamp r) and dense sets (containing an uncountable number of time points with different time stamps) of the form $\langle i, r\rangle, \langle i, r'\rangle, \langle i, r''\rangle, \ldots$ (with the same counter i). This is illustrated in Fig. 1 (taken from [29]), where discrete points are represented by circles (and are labeled with their corresponding time points), and dense sets are represented by continuous lines.

Note that any super-dense time model where the intervals in the underlying time interval sequence are not overlapping is isomorphic to the dense time model.

3.2. First-order linear-time temporal logic

We consider First-Order Linear-time Temporal Logic with Past Operators, which we refer to for simplicity as *LTL*. We use a non-strict version of "until" and "since" operators and add a dense-time version of X and Y, denoted $\tilde{X}$ and $\tilde{Y}$ respectively. This allows us to define a simpler discretization, since discrete-time *LTL* typically uses non-strict version of "until" and "since" (although the approach can be extended to handle also the strict version of "until" and "since" as shown in [1]). Intuitively, the semantics of the "discrete" X operator and its dense-time counterpart $\tilde{X}$ in the super-dense setting is the following: $X\phi$ is true at a time point t if and only if t is immediately followed by another time point t' with the same timestamp in which ϕ holds (e.g. if ϕ holds at $\langle 4, 6\rangle$ in Fig. 1, then $X\phi$ holds at $\langle 3, 6\rangle$); instead, $\tilde{X}\phi$ holds at t if and only if t is immediately followed by a dense set of time points in which ϕ holds (e.g. in the example of Fig. 1, $\tilde{X}\phi$ holds at $\langle 2, 1.5\rangle$ if ϕ holds in the interval between $\langle 2, 1.5\rangle$ and $\langle 3, 6\rangle$).

Given a first-order signature Σ and a set V of variables, we define the syntax of Σ-formulas as follows:

$$\phi := p(u, \ldots, u) \mid \phi \wedge \phi \mid \neg\phi \mid \phi U \phi \mid \phi S \phi \mid X\phi \mid \tilde{X}\phi \mid Y\phi \mid \tilde{Y}\phi$$

$$u := c \mid x \mid f(u, \ldots, u)$$

where p is a predicate symbol in Σ, u is a term, f is a function symbol in Σ, c is a constant symbol in Σ, and x is a variable in V.

Σ-formulas are viewed as a first-order structure interpreting the symbols in Σ and assignments to variables that vary along time. More specifically, a state $s = \langle M, \mu\rangle$ is given by a first-order structure M and an assignment μ of variables of V into the domain of M. Given a state $s = \langle M, \mu\rangle$ and a symbol c in Σ or variable $x \in V$, we use $s(c)$ to denote the interpretation of c in M and $s(x)$ to denote the value $\mu(x)$ assigned by μ to x. Given M, let V^M be the set of states with first-order structure M. A trace $\sigma = \langle M, \tau, \overline{\mu}\rangle$ is given by a first-order structure M, a time model τ, and a mapping $\overline{\mu}$ from the domain of τ into V^M. Given a trace $\sigma = \langle M, \tau, \overline{\mu}\rangle$ and $t \in \tau$, we denote by $\sigma(t)$ the state $\langle M, \overline{\mu}(t)\rangle$.

In our definition of trace, the first-order structure M is shared by all time points, meaning that the interpretation of the symbols in the signature Σ is rigid, that is, it does not vary with time. However, note that the interpretation of symbols may not be "fixed" by the background theory. These "uninterpreted" symbols are also called *parameters*. For example, the signature Σ can include the symbols of the theory of reals (including the constants 0 and 1) and an additional constant symbol p, whose value (i) is not determined by the theory but (ii) does not vary with time (thus, p is a parameter).

We assume a Σ first-order theory $\mathcal{T}$ to be given. Given a Σ first-order structure M, an assignment μ to variables of V, and a Σ first-order formula ϕ over V, we use the standard notion of $\langle M, \mu\rangle \models_{\mathcal{T}} \phi$. In the rest of the paper, we omit the first-order signature Σ and theory $\mathcal{T}$ for simplicity.

Given a trace $\sigma = \langle M, \tau, \overline{\mu}\rangle$, a time point t of τ, and a Σ formula ϕ, we define $\sigma, t \models \phi$ recursively on the structure of ϕ:

$$\sigma, t \models p(u, \dots, u) \text{ iff } \sigma(t) \models p(u, \dots, u)$$

$$\sigma, t \models \phi_1 \wedge \phi_2 \text{ iff } \sigma, t \models \phi_1 \text{ and } \sigma, t \models \phi_2$$

$$\sigma, t \models \neg\phi \text{ iff } \sigma, t \not\models \phi$$

$$\sigma, t \models \phi_1 U \phi_2 \text{ iff there exists } t' \geq t, \sigma, t' \models \phi_2 \text{ and for all } t'', t \leq t'' < t', \sigma, t'' \models \phi_1$$

$$\sigma, t \models \phi_1 S \phi_2 \text{ iff there exists } t' \leq t, \sigma, t' \models \phi_2 \text{ and for all } t'', t' < t'' \leq t, \sigma, t'' \models \phi_1$$

$$\sigma, t \models X\phi \text{ iff there exists } t' > t, \sigma, t' \models \phi \text{ and there exists no } t'', t < t'' < t'$$

$$\sigma, t \models \tilde{X}\phi \text{ iff for all } t' > t, \text{ there exists } t'', t < t'' < t', \sigma, t'' \models \phi$$

$$\sigma, t \models Y\phi \text{ iff } t > 0 \text{ and there exists } t' < t, \sigma, t' \models \phi \text{ and there exists no } t'', t' < t'' < t$$

$$\sigma, t \models \tilde{Y}\phi \text{ iff } t > 0 \text{ and for all } t' < t, \text{ there exists } t'', t' < t'' < t', \sigma, t'' \models \phi$$

Finally, $\sigma \models \phi$ iff $\sigma, \mathbf{0} \models \phi$. We say that ϕ is satisfiable iff there exists σ such that $\sigma \models \phi$. We say that ϕ is valid iff, for all σ, $\sigma \models \phi$.

We say that a formula ϕ_1 globally entails another formula ϕ_2, denoted by $\phi_1 \models_G \phi_2$, iff for all σ, for all t, $\sigma, t \models \phi_1$ implies that $\sigma, t \models \phi_2$. We say that two formulas ϕ_1 and ϕ_2 are globally equivalent, denoted by $\phi_1 \equiv_G \phi_2$ iff $\phi_1 \models_G \phi_2$ and $\phi_2 \models_G \phi_1$.

3.3. Next and yesterday

As discussed at the beginning of previous section, we define the semantics of X ("next") and Y ("yesterday") also in the case of dense or super-dense time. In the case of weakly-monotonic time, $X\phi$ can be true only on a discrete step (i.e., in $\langle n, t\rangle$ if $\langle n+1, t\rangle$ is also in T). In the case of strictly-monotonic time, $X\phi$ is always false.

As for the "dense" counterpart of X and Y, in the super-dense time case, $\tilde{X}\phi$ is false in the discrete steps, while it is true on right accumulation points for points satisfying ϕ. Thus, $\tilde{X}\phi$ is always false in the case of discrete time, while in the case of dense time it is true in the right accumulation points of points satisfying ϕ.

In the discrete-time setting, we often use also the functional counterpart of X, here denoted by *next*, as in [10]. Given a term u, the interpretation of $next(u)$ in a trace σ at the time point t is equal to the value of u assigned by σ at the time point $t+1$. "next" does not typically have a counterpart in the dense time case.

3.4. Abbreviations

We use the following standard abbreviations:

$$\phi_1 \vee \phi_2 := \neg(\neg\phi_1 \wedge \neg\phi_2)$$

$$\top := p \vee \neg p \qquad \bot := \neg\top$$

$$F\phi := \top U \phi \qquad G\phi := \neg(F\neg\phi)$$

$$P\phi := \top S \phi \qquad H\phi := \neg(P\neg\phi)$$

$Z\phi$, which in the discrete-time setting is usually defined as $\neg Y \neg \phi$ (which is equivalent to $Y\top \rightarrow Y\phi$), is a weak version of Y that is always true in the initial point. Here, it is extended to take into account the (super-)dense time models:

$$Z\phi := (Y\top \vee \tilde{Y}\top) \rightarrow Y\phi \qquad \tilde{Z}\phi := (Y\top \vee \tilde{Y}\top) \rightarrow \tilde{Y}\phi$$

Finally, we define further abbreviations to represent the strict versions of F and P also with counting:

$$\begin{aligned} &\tilde{F}\phi := \tilde{X}F\phi | XF\phi && \tilde{P}\phi := \tilde{Y}P\phi | YP\phi \\ &\tilde{F}^1\phi := \tilde{F}\phi && \tilde{P}^1\phi := \tilde{P}\phi \\ &\tilde{F}^k\phi := \tilde{F}(\phi \wedge \tilde{F}^{k-1}\phi) && \tilde{P}^k\phi := \tilde{P}(\phi \wedge \tilde{P}^{k-1}\phi) \end{aligned}$$

$\tilde{G}$ and $\tilde{H}$ are defined as:

$$\begin{aligned} &\tilde{G}\phi := \neg(\tilde{F}\neg\phi) && \tilde{H}\phi := \neg(\tilde{P}\neg\phi) \\ &\tilde{G}^k\phi := \neg(\tilde{F}^k\neg\phi) && \tilde{H}^k\phi := \neg(\tilde{P}^k\neg\phi) \end{aligned}$$

3.5. *Finite variability*

As usual in many works on real-time temporal logics (e.g., [5,30]), we assume the “finite variability” of traces, i.e., that the evaluation of predicates by a trace changes from true to false or vice versa only finitely-often in any finite interval of time. This can be lifted to temporal formulas in the sense that temporal operators preserve the finite variability property (as proved, for example, in [5]). Formally, we say that a trace σ is *fine* for ϕ in a time interval I iff for all $t, t' \in I$, $\sigma, t \models \phi$ iff $\sigma, t' \models \phi$. A trace σ has the *finite variability* property iff for every formula ϕ there exists a sequence of points $t_0, t_1, t_2 \ldots$ of σ such that σ is *fine* for ϕ in every interval (t_i, t_{i+1}), for $i \geq 0$.

Note that, in the case of finite variability, $\tilde{X}\phi$ means that ϕ is true in a right neighborhood (thus coinciding with the definition given in [1])[1]and that $\tilde{X}\phi \vee \tilde{X}\neg\phi$ is always true in any non-discrete point. Moreover, the following equivalences hold:

$$\begin{aligned} &\neg\tilde{X}\phi = X\top \vee \tilde{X}\neg\phi \\ &\neg X\phi = \tilde{X}\top \vee X\neg\phi. \end{aligned}$$

In the following, we assume that traces have the finite variability property.

3.6. *Metric temporal logic with counting operators*

In this section, we define an extension of Metric Temporal Logic (*MTL*) [4,16] extended with first-order predicates, parametric intervals (intervals whose bounds are defined by expressions over parameters), and counting operators.

Metric Temporal Logic with Counting (*MTLC*) formulas are built with the following grammar:

$$\begin{aligned} &\phi := p(u, \ldots, u) \mid \phi \wedge \phi \mid \neg\phi \mid \phi U_I \phi \mid \phi S_I \phi \mid \overrightarrow{C}^k_{<cu}\phi \mid \overleftarrow{C}^k_{<cu}\phi \\ &I := [cu, cu] \mid (cu, cu] \mid [cu, cu) \mid (cu, cu) \mid [cu, \infty) \mid (cu, \infty) \\ &cu := c \mid f(cu, \ldots, cu) \end{aligned}$$

where the terms u are defined as before and cu are terms that do not contain variables. Thus, the bounds of intervals used in *MTLC* are rigid and may contain parameters. We assume here that the background first-order theory contains the theory of reals and that the terms cu have real type.

Intuitively, $\overrightarrow{C}^k_{<cu}\phi$ is true if and only if ϕ holds at least k times in the interval $[0, cu)$ (and similarly for $\overleftarrow{C}^k_{<cu}\phi$). The abbreviations F_I, G_I, P_I, H_I and their strict versions are defined in the usual way. Moreover, for all logics defined in this section, we abbreviate the intervals $[0, a]$, $[0, a)$, $[a, \infty)$, (a, ∞), $[a, a]$, by respectively $\leq a$, $< a$, $\geq a$, $> a$, $= a$. Thus, for example, $F_{=p}b$ is an abbreviation of $F_{[p,p]}b$.

Let $\sigma = \langle M, \tau, \overline{\mu} \rangle$. We give the semantics just for the metric operators:

$$\begin{aligned} \sigma, t \models \phi_1 U_I \phi_2 \text{ iff there exists } & t' \geq t, v(t') - v(t) \in M(I), \sigma, t' \models \phi_2 \\ & \text{and for all } t'', t \leq t'' < t', \sigma, t'' \models \phi_1 \end{aligned}$$

[1] However, the definition of the semantics of $\tilde{X}$ given in the previous section is general, and dual to the semantics of X.

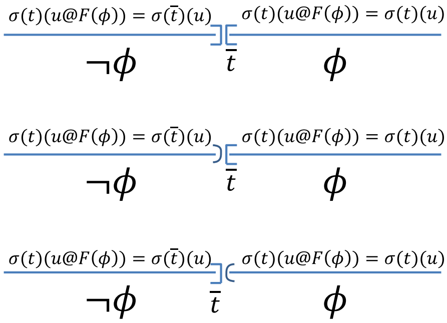
$\sigma(t)(u@F(\phi)) = \sigma(\bar{t})(u)$
$\sigma(t)(u@F(\phi)) = \sigma(t)(u)$
$\neg\phi$
$\bar{t}$
ϕ
$\sigma(t)(u@F(\phi)) = \sigma(\bar{t})(u)$
$\sigma(t)(u@F(\phi)) = \sigma(t)(u)$
$\neg\phi$
$\bar{t}$
ϕ
$\sigma(t)(u@F(\phi)) = \sigma(\bar{t})(u)$
$\sigma(t)(u@F(\phi)) = \sigma(t)(u)$
$\neg\phi$
$\bar{t}$
ϕ

Fig. 2. Graphical view of different cases in which ϕ holds in the future. $\bar{t}$ represents "the next point in the future in which ϕ holds".

$$\begin{aligned}
&\sigma, t \models \phi_1 S_I \phi_2 \text{ iff there exists } t' \leq t, v(t) - v(t') \in M(I), \sigma, t' \models \phi_2\\
&\qquad\qquad\qquad \text{and for all } t'', t' < t'' \leq t, \sigma, t'' \models \phi_1\\
&\sigma, t \models \overrightarrow{C}^k_{<cu}(\phi) \text{ iff there exist } t_1, \ldots, t_k,\\
&\qquad\qquad\qquad t < t_1 < t_2 < \ldots < t_k, v(t_k) - v(t) < M(cu)\\
&\qquad\qquad\qquad \text{such that for all } i \in [1,k], \sigma, t_i \models \phi\\
&\sigma, t \models \overleftarrow{C}^k_{<cu}(\phi) \text{ iff there exist } t_1, \ldots, t_k,\\
&\qquad\qquad\qquad t_k < t_{k-1} < \ldots < t_1 < t v(t) - v(t_k) < M(cu)\\
&\qquad\qquad\qquad \text{such that for all } i \in [1,k], \sigma, t_i \models \phi
\end{aligned}$$

where $M(I)$ is the set obtained from I by substituting the terms at the endpoints with their interpretation (thus it may be also the empty set).

$MTLC_{0,\infty}$is the subset of $MTLC$ where the intervals in metric operators are in the form $[0,a]$, $(0,a]$, $[0,a)$, $(0,a)$, $[a,\infty)$, (a,∞).

We consider Event-Clock operators as abbreviations:

$$\triangleright_I \phi := (\neg\phi) U_I \phi$$
$$\triangleleft_I \phi := (\neg\phi) S_I \phi$$

4. LTL with event freezing functions

4.1. Until next occurrence

Before introducing the new operators, we observe some subtleties of the dense-time semantics. In the discrete-time setting, $F\phi$ and $(\neg\phi)U\phi$ are equivalent. In other words, if ϕ is true in the future, there exists a first point in which it is true, while ϕ is false in all preceding points. This is not the case in the dense-time setting, since, for example, the third trace of Fig. 2 satisfies $F\phi$ but not $(\neg\phi)U\phi$: for every time in which ϕ holds, there exists a left open interval in which ϕ holds as well.

We can instead use another variant of the until operator defined as:

$$\phi_1 U_C \phi_2 := \phi_1 U (\phi_2 \vee (\phi_1 \wedge \tilde{X}\phi_2))$$

Thus, with U_C we are requiring that ϕ_2 holds in a point or in every point of a right interval. In this case, we are guaranteed that there exists a minimum point that satisfies such a condition. In fact, since we are assuming finite variability, $F\phi$ is equivalent to $(\neg\phi)U_C\phi = (\neg\phi)U(\phi \vee \tilde{X}\phi)$. In the next sections, we will use this condition to characterize the next point in the future that satisfies ϕ. In particular, when we say "the next point in the future in which ϕ holds", we actually mean that ϕ holds in that point or in a right left-open interval (see also Fig. 2). Similarly, for the past case.

Note that this is related to the issue of U in the dense time setting raised first by Bouajjani and Lakhnech in [31] and later by Raskin and Schobbens in [14], namely, that $\phi_1 U \phi_2$ is satisfied only if the time interval in which ϕ_2 holds is left-closed. In [31], this is solved by considering $(\phi_1 \vee \phi_2)U\phi_2$. However, this does not solve our issue of characterizing the first point in which ϕ_2 holds. In [14], the issue was solved at the semantic level, by defining the U operator on timed state sequences that are fine for the subformulas, and quantifying over the time intervals of the sequence instead of over the points of the time domain. We instead choose a more classical approach to define the semantics, which seems to clarify better what we mean for "the next point in which ϕ holds". Our semantics is closer to the one defined in [15] for event clocks in Event-Clock Timed Automata and for the corresponding quantifiers in the equally-expressive monadic logic. Differently from [15], however, our solution does not require the use of any nonstandard real number, which is instead used in [15] to handle the case in which ϕ holds in a left-open interval.

4.2. *Event freezing functions*

We extend the logic with two binary operators, "at next" $u@\tilde{F}(\phi)$ and "at last" $u@\tilde{P}(\phi)$, which take in input a term u and a formula ϕ and represent the value of u at the next point in the future, respectively at the last point in the past, in which ϕ holds. If such point does not exist, we consider a default value represented by a variable $def_{u@\tilde{F}(\phi)}$ or $def_{u@\tilde{P}(\phi)}$. We also use an if-then-else operator $ite(\phi, u, v)$, extended to the temporal case, which evaluates to u or v depending on whether ϕ holds or not.

The set of *LTL* with Event Freezing Functions (*LTL-EF*) formulas is therefore defined as follows:

$$\phi := p(u,\dots,u) \mid \phi \wedge \phi \mid \neg\phi \mid \phi U \phi \mid \phi S \phi \mid X\phi \mid \tilde{X}\phi \mid Y\phi \mid \tilde{Y}\phi$$
$$u := c \mid x \mid f(u,\dots,u) \mid u@\tilde{F}(\phi) \mid u@\tilde{P}(\phi) \mid ite(\phi, u, u)$$

The semantics of *LTL* is extended as follows:

- $\sigma(t)(u@\tilde{F}(\phi)) = \sigma(t')(u)$ if there exists $t' > t$ such that, for all t'', $t < t'' < t'$, $\sigma, t'' \not\models \phi$ and $\sigma, t' \models \phi$;
  $\sigma(t)(u@\tilde{F}(\phi)) = \sigma(t')(u)$ if there exists $t' \geq t$ such that, for all t'', $t < t'' \leq t'$, $\sigma, t'' \not\models \phi$ and $\sigma, t' \models \tilde{X}\phi$;
  otherwise, $\sigma(t)(u@\tilde{F}(\phi)) = \sigma(t)(def_{u@\tilde{F}(\phi)})$
- $\sigma(t)(u@\tilde{P}(\phi)) = \sigma(t')(u)$ if there exists $t' < t$ such that, for all t'', $t' < t'' < t$, $\sigma, t'' \not\models \phi$ and $\sigma, t' \models \phi$;
  $\sigma(t)(u@\tilde{P}(\phi)) = \sigma(t')(u)$ if there exists $t' \leq t$ such that, for all t'', $t' < t'' \leq t$, $\sigma, t'' \not\models \phi$ and $\sigma, t' \models \tilde{Y}\phi$;
  otherwise, $\sigma(t)(u@\tilde{P}(\phi)) = \sigma(t)(def_{u@\tilde{P}(\phi)})$
- $\sigma(t)(ite(\phi, u_1, u_2)) = \sigma(t)(u_1)$ if $\sigma, t \models \phi$, else $\sigma(t)(ite(\phi, u_1, u_2)) = \sigma(t)(u_2)$

where $def_{u@\tilde{F}(\phi)}$ and $def_{u@\tilde{P}(\phi)}$ are extra variables added for the event-freezing functions that are used to represent a default value when there is no future/past occurrence of ϕ.

The "if-then-else" operator *ite* can be used to define the non-strict version:

$$u@F(\phi) := ite(\phi, u, u@\tilde{F}(\phi))$$
$$u@P(\phi) := ite(\phi, u, u@\tilde{P}(\phi)).$$

We define the following abbreviations:

$$u@\tilde{F}^1(\phi) := u@\tilde{F}(\phi) \qquad u@\tilde{F}^{i+1}(\phi) := (u@\tilde{F}(\phi))@\tilde{F}^i(\phi) \text{ for } i \geq 1$$
$$u@\tilde{P}^1(\phi) := u@\tilde{P}(\phi) \qquad u@\tilde{P}^{i+1}(\phi) := (u@\tilde{P}(\phi))@\tilde{P}^i(\phi) \text{ for } i \geq 1$$

In principle, we can also define the function *next* as an abbreviation: $next(u) = u@\tilde{F}(\top)$. This would have the standard meaning on discrete steps, and a default value on other points.

4.3. *Extension with explicit time (XLTL-EF)*

In this section, we extend the language with an explicit notion of time that can be constrained using the event freezing functions defined above. In particular, we introduce an explicit symbol *time*, which represents the time elapsed from the initial state. We allow *time* to be compared with constant terms.

The new set of *LTL-EF* formulas with explicit time (*XLTL-EF*) is defined as follows:

$$\phi := p(u,\dots,u) \mid tu \bowtie cu \mid \phi \wedge \phi \mid \neg\phi \mid \phi U \phi \mid \phi S \phi \mid X\phi \mid \tilde{X}\phi \mid Y\phi \mid \tilde{Y}\phi$$
$$u := c \mid x \mid f(u,\dots,u) \mid u@\tilde{F}(\phi) \mid u@\tilde{P}(\phi) \mid ite(\phi, u, u)$$
$$cu := c \mid f(cu,\dots,cu) \mid cu@\tilde{F}(\phi) \mid cu@\tilde{P}(\phi) \mid ite(\phi, cu, cu)$$
$$tu := time \mid time@\tilde{F}(\phi) \mid time@\tilde{P}(\phi) \mid time@\tilde{F}(\phi) - time \mid time - time@\tilde{P}(\phi)$$
$$\bowtie := < \mid > \mid \leq \mid \geq \mid = \mid \neq$$

The semantics of *LTL-EF* is extended by simply setting $\sigma(t)(time) := \nu(t)$. Moreover, we define two additional abbreviations: $time\_until(\phi) := time@\tilde{F}(\phi) - time$ and $time\_since(\phi) := time - time@\tilde{P}(\phi)$.

Note that we assume that the signature Σ contains the real arithmetic operators and that the underlying theory contains the theory of reals.

4.4. Sensor example

Consider a sensor with input y and output x and a Boolean flag *correct* that represents whether or not the value reported by the sensor is correct. Let us specify that the output x is always equal to the last correct input value with $G(x = y@P(correct))$. We assume that a failure is permanent: $G(\neg correct \rightarrow G\neg correct)$. Consider also a Boolean variable *read* that represents the event of reading the variable x. Let us say that the reading happens periodically with period p: $p > 0 \wedge read \wedge G(read \rightarrow \rhd_{=p} read)$. Finally, let us say that an alarm a is true if and only if the last two read values are the same: $G(a \leftrightarrow x@\tilde{P}(read) = x@\tilde{P}^2(read))$.

We would like to prove that, given the above scenario, every point in which the sensor is not correct is followed within $2 * p$ by an alarm:

$$\begin{aligned}
&(G(x = y@P(correct)) \wedge G(\neg correct \rightarrow G\neg correct)\wedge \\
&p > 0 \wedge read \wedge G(read \rightarrow \rhd_{=p} read) \wedge G(a \leftrightarrow x@\tilde{P}(read) = x@\tilde{P}^2(read))) \\
&\rightarrow G(\neg correct \rightarrow F_{\leq 2*p} a)
\end{aligned}$$

In the following, we show that problems of this kind can be indeed solved automatically with SMT-based techniques.

5. From $MTLC_{0,\infty}$ to *LTL-EF*

In this section, we show how $MTLC_{0,\infty}$formulas can be expressed in *XLTL-EF*.

The following equivalences show the coverage of the cases with bounded intervals (note that we are using global equivalences $\equiv_G$ so that $MTLC_{0,\infty}$formulas can be rewritten recursively into *XLTL-EF* formulas):

$$\phi_1 U_{<p} \phi_2 \equiv_G \phi_1 U \phi_2 \wedge time@F(\phi_2) - time < p \tag{1}$$

$$\phi_1 S_{<p} \phi_2 \equiv_G \phi_1 S \phi_2 \wedge time - time@P(\phi_2) < p \tag{2}$$

$$\begin{aligned}
\phi_1 U_{\leq p} \phi_2 \equiv_G \phi_1 U \phi_2 \wedge \qquad & \\
& (\neg\phi_2 U \phi_2 \wedge time@F(\phi_2) - time \leq p \vee \\
& \neg\phi_2 U \tilde{X} \phi_2 \wedge time@F(\phi_2) - time < p)
\end{aligned} \tag{3}$$

$$\begin{aligned}
\phi_1 S_{\leq p} \phi_2 \equiv_G \phi_1 S \phi_2 \wedge \qquad & \\
& (\neg\phi_2 S \phi_2 \wedge time - time@P(\phi_2) \leq p \vee \\
& \neg\phi_2 S \tilde{Y} \phi_2 \wedge time - time@P(\phi_2) < p)
\end{aligned} \tag{4}$$

$$\phi_1 U_{(0,p)} \phi_2 \equiv_G \phi_1 U (\tilde{X} \phi_1 U \phi_2 \wedge time@\tilde{F}(\phi_2) - time < p) \tag{5}$$

$$\phi_1 S_{(0,p)} \phi_2 \equiv_G \phi_1 S (\tilde{Y} \phi_1 S \phi_2 \wedge time - time@\tilde{P}(\phi_2) < p) \tag{6}$$

$$\begin{aligned}
\phi_1 U_{(0,p]} \phi_2 \equiv_G \phi_1 U ((\tilde{X} \phi_1 U \phi_2) \wedge \qquad & \\
& (((\tilde{X} \neg\phi_2 U \phi_2) \wedge (time@\tilde{F}(\phi_2) - time \leq p)) \vee \\
& ((\tilde{X} \neg\phi_2 U \tilde{X} \phi_2) \wedge (time@\tilde{F}(\phi_2) - time < p))))
\end{aligned} \tag{7}$$

$$\begin{aligned}
\phi_1 S_{(0,p]} \phi_2 \equiv_G \phi_1 S ((\tilde{Y} \phi_1 S \phi_2) \wedge \qquad & \\
& (((\tilde{Y} \neg\phi_2 S \phi_2) \wedge (time - time@\tilde{P}(\phi_2) \leq p)) \vee \\
& ((\tilde{Y} \neg\phi_2 S \tilde{Y} \phi_2) \wedge (time - time@\tilde{P}(\phi_2) < p))))
\end{aligned} \tag{8}$$

$$\overrightarrow{C}^{k}_{<p}(\phi) \equiv_G time@\tilde{F}^k(\phi) - time < p \wedge \tilde{F}^k(\phi) \tag{9}$$

$$\overleftarrow{C}^{k}_{<p}(\phi) \equiv_G time - time@\tilde{P}^k(\phi) < p \wedge \tilde{P}^k(\phi) \tag{10}$$

We can reduce the remaining metric operators to the above ones with the following reduction:

$$\begin{aligned}
\phi_1 U_{>p} \phi_2 \equiv_G & (p = 0 \wedge \phi_1 U_{\neq 0} \phi_2) \vee \\
& (p > 0 \wedge G_{\leq p}(\phi_1 \wedge \phi_1 U_{\neq 0} \phi_2))
\end{aligned} \tag{11}$$

$$\begin{aligned}
\phi_1 S_{>p} \phi_2 \equiv_G & (p = 0 \wedge \phi_1 S_{\neq 0} \phi_2) \vee \\
& (p > 0 \wedge time > p \wedge H_{\leq p}(\phi_1 \wedge \phi_1 S_{\neq 0} \phi_2))
\end{aligned} \tag{12}$$

$$\begin{aligned}\phi_1 U_{\geq p}\phi_2 \equiv_G &(p = 0 \wedge \phi_1 U \phi_2)\vee \\ &\qquad (p > 0 \wedge G_{<p}\phi_1 \wedge G_{\leq p} P_{\leq 0}(\tilde{H}_{\leq 0}\phi_1 \wedge \phi_1 U \phi_2)) \qquad (13)\\ \phi_1 S_{\geq p}\phi_2 \equiv_G &(p = 0 \wedge \phi_1 S \phi_2)\vee \\ &\qquad (p > 0 \wedge time \geq p \wedge H_{<p}\phi_1 \wedge H_{\leq p} F_{\leq 0}(\tilde{G}_{\leq 0}\phi_1 \wedge \phi_1 S \phi_2)) \qquad (14)\end{aligned}$$

where $\phi_1 U_{\neq 0}\phi_2$ and $\phi_1 S_{\neq 0}\phi_2$ are used here as abbreviations for $\phi_1 U \tilde{X}(\phi_1 U \phi_2)$ and $\phi_1 S \tilde{Y}(\phi_1 U \phi_2)$, respectively.

Note, in particular, the need of adding $P_{=0}\phi$ in the case the time distance is p due to the super-dense semantics. In case of dense time, the last two equivalences can be simplified to $\phi_1 U_{\geq p}\phi_2 \equiv_G G_{<p}\phi_1 \wedge G_{\leq p}\phi_1 U \phi_2$ and $\phi_1 S_{\geq p}\phi_2 \equiv_G H_{<p}\phi_1 \wedge H_{\leq p}\phi_1 S \phi_2$.

Theorem 1. *Every MTLC$_{0,\infty}$ formula can be rewritten into an equivalent XLTL-EF formula.*

Proof. It is sufficient to prove that all above equivalences are correct. We first prove the future cases. In the following, M is the first-order structure of the trace σ and thus $M(p)$ is the evaluation of p in σ.
Equivalence (1)

- **Case $\boldsymbol{\phi_1 U_{<p}\phi_2 \models_G \phi_1 U \phi_2 \wedge time@F(\phi_2) - time < p}$.**
If $\sigma, t \models \phi_1 U_{<p}\phi_2$ then there exists $t' \geq t$ such that $v(t') - v(t) < M(p)$, $\sigma, t' \models \phi_2$, and for all t'', $t \leq t'' < t'$, $\sigma, t'' \models \phi_1$. Thus, $\sigma, t \models \phi_1 U \phi_2$. Moreover, there exists $\bar{t}'$, $t \leq \bar{t}' \leq t'$, $\sigma, \bar{t}' \models \phi_2 \vee \tilde{X}\phi_2$, $v(\bar{t}') - v(t) < M(p)$, and for all t'', $t \leq t'' < \bar{t}'$, $\sigma, t'' \models \neg\phi_2$. Thus, $\sigma, t \models time@F(\phi_2) - time < p$.
- **Case $\boldsymbol{\phi_1 U \phi_2 \wedge time@F(\phi_2) - time < p \models_G \phi_1 U_{<p}\phi_2}$.**
If $\sigma, t \models \phi_1 U \phi_2$, then there exists t', $t \leq t'$, $\sigma, t' \models \phi_2 \vee \tilde{X}\phi_2$, and for all t'', $t \leq t'' < t'$, $\sigma, t'' \models \neg\phi_2$. If $\sigma, t \models time@F(\phi_2) - time < p$, then $v(t') - v(t) < M(p)$. Thus, $\sigma, t \models \phi_1 U_{<p}\phi_2$.

Equivalence (3)

- **Case $\boldsymbol{\phi_1 U_{\leq p}\phi_2 \models_G \phi_2 \equiv_G \phi_1 U \phi_2 \wedge (\neg\phi_2 U \phi_2 \wedge time@F(\phi_2) - time \leq p \vee \neg\phi_2 U \tilde{X}\phi_2 \wedge time@F(\phi_2) - time < p)}$.**
If $\sigma, t \models \phi_1 U_{\leq p}\phi_2$ then there exists $t' \geq t$ such that $v(t') - v(t) \leq M(p)$, $\sigma, t' \models \phi_2$, and for all t'', $t \leq t'' < t'$, $\sigma, t'' \models \phi_1$. Thus, $\sigma, t \models \phi_1 U \phi_2$. Moreover, there exists $\bar{t}'$, $t \leq \bar{t}' \leq t'$, $\sigma, \bar{t}' \models \phi_2 \vee \tilde{X}\phi_2$, and for all t'', $t \leq t'' < \bar{t}'$, $\sigma, t'' \models \neg\phi_2$. Also, either $\sigma, \bar{t}' \models \phi_2$ and $v(\bar{t}') - v(t) \leq M(p)$, or $\sigma, \bar{t}' \models \tilde{X}\phi_2$ and $v(\bar{t}') - v(t) < M(p)$. Thus, $\sigma, t \models (\phi_1 \wedge \neg\phi_2) U \phi_2 \wedge time@F(\phi_2) - time \leq p \vee (\phi_1 \wedge \neg\phi_2) U (\phi_1 \wedge \tilde{X}\phi_2) \wedge time@F(\phi_2) - time < p$.
- **Case $\boldsymbol{\phi_1 U \phi_2 \wedge (\neg\phi_2 U \phi_2 \wedge time@F(\phi_2) - time \leq p \vee \neg\phi_2 U \tilde{X}\phi_2 \wedge time@F(\phi_2) - time < p) \models_G \phi_1 U_{\leq p}\phi_2}$.**
If $\sigma, t \models \phi_1 U \phi_2$, then there exists t', $t \leq t'$, $\sigma, t' \models \phi_2$, and for all t'', $t \leq t'' < t'$, $\sigma, t'' \models \phi_1$. If $\sigma, t \models \neg\phi_2 U \phi_2 \wedge time@F(\phi_2) - time \leq p$, then there exists $\bar{t}'$, $t \leq \bar{t}' \leq t'$, such that $\sigma, \bar{t}' \models \phi_2$, and for all t'', $t \leq t'' < \bar{t}'$, $\sigma, t'' \models \neg\phi_2$ and $v(\bar{t}') - v(t) \leq M(p)$. Since $\bar{t}' \leq t'$, $\sigma, t \models \phi_1 U_{\leq p}\phi_2$. Similarly, if $\sigma, t \models \neg\phi_2 U \tilde{X}\phi_2 \wedge time@F(\phi_2) - time < p$, then there exists $\bar{t}'$, $t \leq \bar{t}' \leq t'$, $\sigma, \bar{t}' \models \tilde{X}\phi_2$, and for all t'', $t \leq t'' < \bar{t}'$, $\sigma, t'' \models \neg\phi_2$ and $v(\bar{t}') - v(t) \leq M(p)$. Thus, $\sigma, t \models \phi_1 U_{\leq p}\phi_2$.

Equivalence (5)

- **Case $\boldsymbol{\phi_1 U_{(0,p)}\phi_2 \models_G \phi_1 U(\tilde{X}\phi_1 U \phi_2 \wedge time@\tilde{F}(\phi_2) - time < p)}$.**
If $\sigma, t \models \phi_1 U_{<p}\phi_2$ then there exists $t' < t$ such that $0 < v(t') - v(t) < M(p)$, $\sigma, t' \models \phi_2$, and for all t'', $t \leq t'' < t'$, $\sigma, t'' \models \phi_1$. Let t_0 the greatest point such that $t \leq t_0$ and $v(t) = v(t_0)$. Thus, for all t'', $t \leq t'' \leq t_0$, $\sigma, t \models \phi_1$. Moreover, there exists $\bar{t}'$, $t_0 < \bar{t}' \leq t'$, $\sigma, \bar{t}' \models \phi_2$, $v(\bar{t}') - v(t_0) < M(p)$, and for all t'', $t_0 \leq t'' < \bar{t}'$, $\sigma, t'' \models \neg\phi_2$. Thus, $\sigma, t_0 \models \tilde{X}\phi_1 U \phi_2 \wedge time@\tilde{F}(\phi_2) - time < p$. Thus, $\sigma, t \models \phi_1 U(\tilde{X}\phi_1 U \phi_2 \wedge time@\tilde{F}(\phi_2) - time < p)$.
- **Case $\boldsymbol{\phi_1 U(\tilde{X}\phi_1 U \phi_2 \wedge time@\tilde{F}(\phi_2) - time < p) \models_G \phi_1 U_{(0,p)}\phi_2}$.**
If $\sigma, t \models \phi_1 U(\tilde{X}\phi_1 U \phi_2 \wedge time@\tilde{F}(\phi_2) - time < p)$, then there exists t', $t \leq t'$, $\sigma, t' \models \tilde{X}\phi_1 U \phi_2 \wedge time@\tilde{F}(\phi_2) - time < p)$ and for all t'', $t \leq t'' < t'$, $\sigma, t'' \models \phi_1$. Thus, there exists $\bar{t}' > t'$, such that $\sigma, \bar{t}' \models \phi_2$, $0 < v(\bar{t}') - v(t') < M(p)$, and for all t'', $t' < t'' < \bar{t}'$, $\sigma, t'' \models \phi_1$. Thus $\sigma, t \models \phi_1 U_{(0,p)}\phi_2$.

Equivalence (7)

- **Case $\boldsymbol{\phi_1 U_{(0,p]}\phi_2 \models_G \phi_1 U((\tilde{X}\phi_1 U \phi_2) \wedge (((\tilde{X}\neg\phi_2 U \phi_2) \wedge (time@\tilde{F}(\phi_2) - time \leq p)) \vee ((\tilde{X}\neg\phi_2 U \tilde{X}\phi_2) \wedge (time@\tilde{F}(\phi_2) - time < p))))}$.**
If $\sigma, t \models \phi_1 U_{(0,p]}\phi_2$ then there exists $t' > t$ such that $0 < v(t') - v(t) \leq M(p)$, $\sigma, t' \models \phi_2$, and for all t'', $t \leq t'' < t'$, $\sigma, t'' \models \phi_1$. Let t_0 be the greatest point such that $t \leq t_0$ and $v(t) = v(t_0)$. Thus, for all t'', $t \leq t'' \leq t_0$, $\sigma, t \models \phi_1$. Moreover, $\sigma, t_0 \models \tilde{X}\phi_1 U \phi_2$. Thus, either $\sigma, t_0 \models \tilde{X}\neg\phi_2 U \phi_2$ or $\sigma, t_0 \models \tilde{X}\neg\phi_2 U \tilde{X}\phi_2$. In the first case, there exists $\bar{t}'$, $t_0 < \bar{t}' \leq t'$, $\sigma, \bar{t}' \models \phi_2$, and for all t'', $t_0 \leq t'' < \bar{t}'$, $\sigma, t'' \models \neg\phi_2$. Since $\bar{t}' \leq t'$, $\sigma, t_0 \models time@\tilde{F}(\phi_2) - time \leq p$. Similarly, in the second case, there exists $\bar{t}'$, $t_0 < \bar{t}' \leq t'$, $\sigma, \bar{t}' \models \tilde{X}\phi_2$, and for all t'', $t_0 \leq t'' < \bar{t}'$, $\sigma, t'' \models \neg\phi_2$. Since $\bar{t}' \leq t'$, $\sigma, t_0 \models time@\tilde{F}(\phi_2) -$

$time \leq p$. Thus, $\sigma, t \models \phi_1 U((\tilde{X}\phi_1 U \phi_2) \wedge (((\neg\phi_2 U \phi_2) \wedge (time@\tilde{F}(\phi_2) - time \leq p)) \vee ((\neg\phi_2 U \tilde{X}\phi_2) \wedge (time@\tilde{F}(\phi_2) - time < p))))$.

- **Case** $\boldsymbol{\phi_1}\mathbf{U}((\tilde{\mathbf{X}}\boldsymbol{\phi_1}\mathbf{U}\boldsymbol{\phi_2}) \wedge (((\tilde{\mathbf{X}}\neg\boldsymbol{\phi_2}\mathbf{U}\boldsymbol{\phi_2}) \wedge (\mathbf{time@\tilde{F}}(\boldsymbol{\phi_2}) - \mathbf{time} \leq \mathbf{p})) \vee ((\tilde{\mathbf{X}}\neg\boldsymbol{\phi_2}\mathbf{U}\tilde{\mathbf{X}}\boldsymbol{\phi_2}) \wedge (\mathbf{time@\tilde{F}}(\boldsymbol{\phi_2}) - \mathbf{time} < \mathbf{p}))))$ $\models_{\mathbf{G}} \boldsymbol{\phi_1}\mathbf{U}_{(\mathbf{0},\mathbf{p}]}\boldsymbol{\phi_2}$**.**
  If $\sigma, t \models \phi_1 U((\tilde{X}\phi_1 U \phi_2) \wedge (((\neg\phi_2 U \phi_2) \wedge (time@\tilde{F}(\phi_2) - time \leq p)) \vee ((\neg\phi_2 U \tilde{X}\phi_2) \wedge (time@\tilde{F}(\phi_2) - time < p))))$ then there exists t', $t \leq t'$, $\sigma, t' \models (\tilde{X}\phi_1 U \phi_2 \wedge (\neg\phi_2 U \phi_2 \wedge time@\tilde{F}(\phi_2) - time \leq p \vee \neg\phi_2 U \tilde{X}\phi_2 \wedge time@\tilde{F}(\phi_2) - time < p))$ and for all t'', $t \leq t'' < t'$, $\sigma, t'' \models \phi_1$. Thus, there exists $\bar{t}' > t'$, such that $\sigma, \bar{t}' \models \phi_2$, $0 < v(\bar{t}') - v(t') \leq M(p)$, and for all t'', $t' < t'' < \bar{t}'$, $\sigma, t'' \models \phi_1$. Thus $\sigma, t \models \phi_1 U_{(0,p]}\phi_2$.

Equivalence (9)

- We prove the equivalence $\overrightarrow{C}^{\mathbf{k}}_{<\mathbf{p}}(\phi) \equiv_{\mathbf{G}} \tilde{\mathbf{F}}^{\mathbf{k}}(\phi) \wedge \mathbf{time@\tilde{F}^{k}}(\phi) - \mathbf{time} < \mathbf{p}$ by induction on k. If $k = 1$, $\overrightarrow{C}^{k}_{<p}(\phi) \equiv_G = \tilde{F}_{<p}\phi \equiv_G \tilde{F}^k(\phi) \wedge time@\tilde{F}^k(\phi) - time < p$ (minor variant of the case above). Suppose now that the equivalence holds for $k-1$, $\sigma, t \models \overrightarrow{C}^{k}_{<p}$ iff there exist $t_1, \ldots, t_k$, $t < t_1 < t_2 < \ldots < t_k$, $v(t_k) - v(t) < M(p)$, such that for all $i \in [1,k]$, $\sigma, t_i \models \phi$. Let $t' = t_{k-1}$ and $p' = v(t_k) - v(t_{k-1})$. Then $\sigma, t \models \overrightarrow{C}^{k}_{<p}$ iff there exists t' such that $v(t') - v(t) = p'$ and $\sigma, t' \models \phi \wedge \overrightarrow{C}^{k-1}_{<p-p'}\phi$. By induction, $\sigma, t \models \overrightarrow{C}^{k}_{<p}$ iff there exists t' such that $v(t') - v(t) = p'$ and $\sigma, t' \models \phi \wedge \tilde{F}^{k-1}(\phi) \wedge time@\tilde{F}^{k-1}(\phi) - time < p - p'$, thus iff $\sigma, t \models \phi \wedge \tilde{F}^k(\phi) \wedge time@\tilde{F}^k(\phi) - time < p$.

Equivalence (11)

- **Case** $\boldsymbol{\phi_1}\mathbf{U}_{>\mathbf{p}}\boldsymbol{\phi_2} \models_{\mathbf{G}} (\mathbf{p} = \mathbf{0} \wedge \boldsymbol{\phi_1}\mathbf{U}_{>\mathbf{0}}\boldsymbol{\phi_2}) \vee (\mathbf{p} > \mathbf{0} \wedge \mathbf{G}_{\leq\mathbf{p}}(\boldsymbol{\phi_1} \wedge \boldsymbol{\phi_1}\mathbf{U}_{>\mathbf{0}}\boldsymbol{\phi_2}))$**.**
  If $M(p) = 0$, then the equivalence is trivial. Suppose instead that $M(p) > 0$. If $\sigma, t \models \phi_1 U_{>p}\phi_2$, then there exists $t' > t$ such that $\sigma, t' \models \phi_2$, $v(t') - v(t) > M(p)$, and for all t'', $t \leq t'' < t'$, $\sigma, t'' \models \phi_1$. For all $\bar{t}'' \geq t$, if $v(\bar{t}'') - v(t) \leq M(p)$, then $t' > \bar{t}''$. Thus, $\sigma, \bar{t}'' \models \phi_1 \wedge \phi_1 U_{>0}\phi_2$.
- **Case** $(\mathbf{p} = \mathbf{0} \wedge \boldsymbol{\phi_1}\mathbf{U}_{>\mathbf{0}}\boldsymbol{\phi_2}) \vee (\mathbf{p} > \mathbf{0} \wedge \mathbf{G}_{\leq\mathbf{p}}(\boldsymbol{\phi_1} \wedge \boldsymbol{\phi_1}\mathbf{U}_{>\mathbf{0}}\boldsymbol{\phi_2})) \models_{\mathbf{G}} \boldsymbol{\phi_1}\mathbf{U}_{>\mathbf{p}}\boldsymbol{\phi_2}$**.**
  If $M(p) = 0$, then the equivalence is trivial. Suppose instead that $M(p) > 0$. If $\sigma, t \models G_{\leq p}(\phi_1 \wedge \phi_1 U_{>0}\phi_2)$, then there exists t' such that $v(t') - v(t) = M(p)$ and $\sigma, t' \models \phi_1 U_{>0}\phi_2$. Moreover, for all t'', $t \leq t'' \leq t'$, $\sigma, t'' \models \phi_1$. Thus, $\sigma, t \models \phi_1 U_{>p}\phi_2$.

Equivalence (13)

- **Case** $\boldsymbol{\phi_1}\mathbf{U}_{\geq\mathbf{p}}\boldsymbol{\phi_2} \models_{\mathbf{G}} (\mathbf{p} = \mathbf{0} \wedge \boldsymbol{\phi_1}\mathbf{U}\boldsymbol{\phi_2}) \vee (\mathbf{p} > \mathbf{0} \wedge \mathbf{G}_{<\mathbf{p}}\boldsymbol{\phi_1} \wedge \mathbf{G}_{\leq\mathbf{p}}\mathbf{P}_{\leq\mathbf{0}}(\tilde{\mathbf{H}}_{\leq\mathbf{0}}\boldsymbol{\phi_1} \wedge \boldsymbol{\phi_1}\mathbf{U}\boldsymbol{\phi_2}))$**.**
  If $M(p) = 0$, then the equivalence is trivial. Suppose instead that $M(p) > 0$. If $\sigma, t \models \phi_1 U_{\geq p}\phi_2$, then there exists $t' > t$, such that $\sigma, t' \models \phi_2$, $v(t') - v(t) \geq M(p)$, and for all t'', $t \leq t'' < t'$, $\sigma, t'' \models \phi_1$. For all $\bar{t}'' \geq t$, if $v(\bar{t}'') - v(t) < M(p)$, then $t' > \bar{t}''$. Thus, $\sigma, \bar{t}'' \models \phi_1$. Moreover, if $v(t') - v(t) > M(p)$, then for all t'', $t \leq t''$, if $v(t'') - v(t) \leq M(p)$, then $\sigma, t'' \models \phi_1 U \phi_2$ and thus, $\sigma, t'' \models P_{\leq 0}(\tilde{H}_{\leq 0}\phi_1 \wedge \phi_1 U \phi_2)$. Similarly, if $v(t') - v(t) = M(p)$, then for all t'', $t \leq t'' \leq t'$, then $\sigma, t'' \models \phi_1 U \phi_2$ and thus, $\sigma, t'' \models P_{\leq 0}(\tilde{H}_{\leq 0}\phi_1 \wedge \phi_1 U \phi_2)$. Finally, for all $t'' > t'$, $v(t'') = v(t')$, $\sigma, t'' \models P_{\leq 0}(\tilde{H}_{\leq 0}\phi_1 \wedge \phi_1 U \phi_2)$.
- **Case** $(\mathbf{p} = \mathbf{0} \wedge \boldsymbol{\phi_1}\mathbf{U}\boldsymbol{\phi_2}) \vee (\mathbf{p} > \mathbf{0} \wedge \mathbf{G}_{<\mathbf{p}}\boldsymbol{\phi_1} \wedge \mathbf{G}_{\leq\mathbf{p}}\mathbf{P}_{\leq\mathbf{0}}(\tilde{\mathbf{H}}_{\leq\mathbf{0}}\boldsymbol{\phi_1} \wedge \boldsymbol{\phi_1}\mathbf{U}\boldsymbol{\phi_2})) \models_{\mathbf{G}} \boldsymbol{\phi_1}\mathbf{U}_{\geq\mathbf{p}}\boldsymbol{\phi_2}$**.**
  If $M(p) = 0$, then the equivalence is trivial. Suppose instead that $M(p) > 0$ and $\sigma, t \models G_{<p}\phi_1 \wedge G_{\leq p}P_{\leq 0}(\tilde{H}_{\leq 0}\phi_1 \wedge \phi_1 U \phi_2)$. Let t_0 be the greatest point such that $t \leq t_0$ and $v(t_0) - v(t) = M(p)$. $\sigma, t_0 \models P_{\leq 0}(\tilde{H}_{\leq 0}\phi_1 \wedge \phi_1 U \phi_2)$. Thus there exists t', with $v(t') - v(t) \geq M(p)$, such that $\sigma, t' \models \phi_1 U \phi_2$, and for all $t'' < t'$, if $v(t'') - v(t) >= M(p)$, then $\sigma, t'' \models \phi_1$. Moreover, by hypothesis, also for all $t'' \geq t$, if $v(t'') - v(t) < M(p)$, then $\sigma, t'' \models \phi_1$. Thus, $\sigma, t \models \phi_1 U_{\geq p}\phi_2$.

All cases with past operators are analogous to the future counterpart apart from the following two cases:
Equivalence (12)

- **Case** $(\mathbf{p} = \mathbf{0} \wedge \boldsymbol{\phi_1}\mathbf{S}_{>\mathbf{0}}\boldsymbol{\phi_2}) \vee (\mathbf{p} > \mathbf{0} \wedge \mathbf{time} > \mathbf{p} \wedge \mathbf{H}_{\leq\mathbf{p}}(\boldsymbol{\phi_1} \wedge \boldsymbol{\phi_1}\mathbf{S}_{>\mathbf{0}}\boldsymbol{\phi_2})) \models_{\mathbf{G}} \boldsymbol{\phi_1}\mathbf{S}_{>\mathbf{p}}\boldsymbol{\phi_2}$**.**
  If $M(p) = 0$, then the equivalence is trivial as before. Suppose instead that $M(p) > 0$ and $\sigma, t \models time > p \wedge H_{\leq p}(\phi_1 \wedge \phi_1 S_{>0}\phi_2)$. Since $M(p) > 0$ and $time > M(p)$, there exists $t' < t$ such that $v(t) - v(t') = M(p)$. Thus, $\sigma, t' \models \phi_1 S_{>0}\phi_2$. Moreover, for all t'', $t' \leq t'' \leq t$, $\sigma, t'' \models \phi_1$. Thus, $\sigma, t \models \phi_1 S_{>p}\phi_2$.

Equivalence (14)

- **Case** $(\mathbf{p} = \mathbf{0} \wedge \boldsymbol{\phi_1}\mathbf{S}\boldsymbol{\phi_2}) \vee (\mathbf{p} > \mathbf{0} \wedge \mathbf{time} \geq \mathbf{0} \wedge \mathbf{H}_{<\mathbf{p}}\boldsymbol{\phi_1} \wedge \mathbf{H}_{\leq\mathbf{p}}\mathbf{F}_{\leq\mathbf{0}}(\tilde{\mathbf{G}}_{\leq\mathbf{0}}\boldsymbol{\phi_1} \wedge \boldsymbol{\phi_1}\mathbf{S}\boldsymbol{\phi_2})) \models_{\mathbf{G}} \boldsymbol{\phi_1}\mathbf{S}_{\geq\mathbf{p}}\boldsymbol{\phi_2}$**.**
  If $M(p) = 0$, then the equivalence is trivial as before. Suppose instead that $M(p) > 0$ and $\sigma, t \models time \geq p \wedge H_{<p}\phi_1 \wedge H_{\leq p}F_{\leq 0}(\tilde{G}_{\leq 0}\phi_1 \wedge \phi_1 S \phi_2)$. Since $M(p) > 0$ and $time \geq M(p)$, then there exists a smallest point t_0 such that $t > t_0$ and $v(t) - v(t_0) = M(p)$. Thus, $\sigma, t_0 \models F_{\leq 0}(\tilde{G}_{\leq 0}\phi_1 \wedge \phi_1 S \phi_2)$. Thus there exists t', with $v(t) - v(t') \geq M(p)$, such that

$\sigma, t' \models \phi_1 S \phi_2$, and for all $t'' > t'$, if $v(t) - v(t'') <= M(p)$, then $\sigma, t'' \models \phi_1$. Moreover, by hypothesis, also for all $t'' \le t$, if $v(t) - v(t'') < M(p)$, then $\sigma, t'' \models \phi_1$. Thus, $\sigma, t \models \phi_1 S_{\ge p} \phi_2$. □

6. Discretization

In this section, we show that, given a *XLTL-EF* formula with dense or super-dense time, we can build an equisatisfiable *XLTL-EF* one with discrete time.

The discretization approach is similar to the one described in [23]. The idea is to split the time evolution into a sequence of singular or open intervals in such a way that the trace is fine for the input formula on such intervals.

Thus, given a trace $\sigma = \langle M, \tau, \overline{\mu} \rangle$ satisfying ϕ, we can derive a time interval sequence $I_0, I_1, I_2, \ldots$ that is used to build a trace σ_D with a discrete time model satisfying ϕ_D. This time interval sequence is obtained from the time interval sequence of τ by splitting each interval finitely many times so that 1) σ is fine for all subformulas of ϕ in each interval I_i, 2) each interval I_i in the sequence is singular or open, and 3) for every subformula of ϕ of the form $\phi_1 U \phi_2$ and for every interval I_i, if I_i is open and ϕ_2 holds in I_i, then ϕ_2 must hold in either I_{i-1} or I_{i+1} (this last requirement is used to have a simpler encoding).

In order to encode the time interval sequence, we introduce an extra Boolean variable ι that holds if and only if the current interval is singular. A constraint ψ_ι ensures that the value of this additional variable represents a valid time interval sequence (e.g., after an open interval there must be a singular interval).

Given a formula ϕ over V, we rewrite ϕ into ϕ_D over $V \cup \{\iota\}$ defined as:

$$\phi_D := \mathcal{D}(\phi) \wedge \psi_\iota \wedge \psi_{time}$$

where $\mathcal{D}$, ψ_ι, and ψ_{time} are defined as follows.

$\mathcal{D}(\phi)$ is defined recursively on the structure of ϕ and rewrites the temporal operators distinguishing the case in which the current interval is singular (ι) from the case in which it is open ($\neg\iota$).

Let us consider first $\phi_1 U \phi_2$. In order to hold at a time point t, there must be a future point t' satisfying ϕ_2 and ϕ_1 being true between t and t'; however, in the discretized version, if $t' > t$ and it belongs to an open interval, then also ϕ_1 must hold in that interval. Overall:

$$\mathcal{D}(\phi_1 U \phi_2) := \mathcal{D}(\phi_2) \vee \mathcal{D}(\phi_1) U (\mathcal{D}(\phi_2) \wedge (\mathcal{D}(\phi_1) \vee \iota))$$
$$\mathcal{D}(\phi_1 S \phi_2) := \mathcal{D}(\phi_2) \vee \mathcal{D}(\phi_1) S (\mathcal{D}(\phi_2) \wedge (\mathcal{D}(\phi_1) \vee \iota))$$

Let us consider the cases of $X\phi$ and $\tilde{X}\phi$. $X\phi$ requires to be in a discrete step (i.e., $\iota \wedge X\iota$). $\tilde{X}\phi$ is true if either the current interval is open (i.e., $\neg\iota$) and ϕ holds in this interval, or the next interval is open (i.e., $X\neg\iota$) and ϕ holds in that interval.

$$\mathcal{D}(X\phi) := \iota \wedge X(\iota \wedge \mathcal{D}(\phi))$$
$$\mathcal{D}(\tilde{X}\phi) := (\neg\iota \wedge \mathcal{D}(\phi)) \vee X(\neg\iota \wedge \mathcal{D}(\phi))$$

Let us consider now $u@\tilde{F}(\phi)$. If the current interval is open and ϕ holds in this interval or if the next interval is open and ϕ holds in that interval (i.e., if $\tilde{X}\phi$ holds now), then $u@\tilde{F}(\phi) = u$. Otherwise, $u@\tilde{F}(\phi)$ is translated into the discrete counterpart.

$$\begin{aligned}\mathcal{D}(u@\tilde{F}(\phi)) := ite(&(\neg\iota \wedge \mathcal{D}(\phi)) \vee X(\neg\iota \wedge \mathcal{D}(\phi)),\\ &\mathcal{D}(u),\\ &\mathcal{D}(u)@\tilde{F}(\mathcal{D}(\phi) \vee X(\neg\iota \wedge \mathcal{D}(\phi))))\\ \mathcal{D}(u@\tilde{P}(\phi)) := ite(&(\neg\iota \wedge \mathcal{D}(\phi)) \vee Y(\neg\iota \wedge \mathcal{D}(\phi)),\\ &\mathcal{D}(u),\\ &\mathcal{D}(u)@\tilde{P}(\mathcal{D}(\phi) \vee Y(\neg\iota \wedge \mathcal{D}(\phi))))\end{aligned}$$

The remaining cases can be completed trivially by observing that $\mathcal{D}$ is homomorphic with respect to Boolean connectives, functions, constants, and variables.

ψ_ι encodes the structure of the time model (to enforce for example that after an open interval there must be a singular one and that in a discrete step time does not elapse):

$$\psi_\iota := \iota \wedge G((\iota \wedge \delta = 0 \wedge X(\iota)) \vee (\iota \wedge \delta > 0 \wedge X(\neg\iota)) \vee (\neg\iota \wedge \delta > 0 \wedge X(\iota)))$$

where δ is an abbreviation for $next(time) - time$.

Finally, ψ_{time} forces the uniformity of predicates over $time$ in open intervals:

$$\psi_{time} := \bigwedge_{tu \bowtie cu \in Sub(\phi)} G(\neg\iota \rightarrow ((\mathcal{D}(tu \le cu) \rightarrow X\mathcal{D}(tu \le cu)) \wedge (\mathcal{D}(tu \ge cu) \rightarrow Y\mathcal{D}(tu \ge cu))))$$

where $Sub(\phi)$ denotes the set of subformulas of ϕ.

Note in particular that we require to split the time intervals in such a way that for every constant cu occurring in a time constraint, $[cu, cu]$ is a time interval in the sequence. Note also that cu can be in general a term built with the signature symbols that are interpreted rigidly.

Written as above the discretization clearly produces a formula whose size is exponential in the input. However, since we are interested in equisatisfiability, we can always use extra variables (one for subformula) to obtain a linear-size formula.

We now prove that the translation is correct, i.e., that the new formula is equisatisfiable.

Theorem 2. *ϕ and ϕ_D are equisatisfiable.*

Proof. Given a trace $\sigma = \langle M, \tau, \overline{\mu}\rangle$ satisfying ϕ we can build a trace σ_D with a discrete time model satisfying ϕ_D as follows. Let $I_0, I_1, I_2, \ldots$ be a sequence of time intervals such that 1) σ is fine for all subformulas of ϕ in each interval I_i, 2) each interval I_i in the sequence is singular or open, and 3) in case of super-dense time, for all $i \ge 0$, there exists an integer n_i such that $\langle n_i, t\rangle \in \tau$ for all $t \in I_i$.

We define the discrete time model by setting the value $v(i)$ for all $i \in \mathbb{N}$ based on the time intervals sequence as follows: $v(0) := 0$; $v(i+1) = v(i)$ if I_i is singular and I_{i+1} is singular; $v(i+1) = v(i) + (r(I_{i+1}) - l(I_{i+1}))/2$ if I_i is singular and I_{i+1} is open; $v(i+1) = v(i) + (r(I_i) - l(I_i))/2$ if I_i is open. Let $t_i = v(i)$. Note that $t_i \in I_i$ for every i. Moreover, notice that if $I_i = I_{i+1}$ then $\delta = 0$, if I_i is singular and I_{i+1} is not then δ is equal to half of the length of I_{i+1} and if I_i is not singular then δ is equal to half of the length of I_i.

We build an assignment to ι also based on the time intervals sequence. The value of ι is determined by the sequence of intervals as follows: $\sigma_D(i)(\iota) = \top$ iff I_i is singular. Thus, $\sigma_D \models \psi_\iota$.

Let $\mathbf{t}_i = t_i$ in case σ has a dense time and $\mathbf{t}_i = \langle n_i, t_i\rangle$ in case of super dense time. Let us complete the definition of σ_D by saying that for all $i \ge 0$, $\sigma_D(i)(x) := \sigma(\mathbf{t}_i)(x)$.

We now prove that, for all $i \ge 0$, for all subformulas ψ of ϕ, $\sigma, \mathbf{t}_i \models \psi$ iff $\sigma_D, i \models \mathcal{D}(\psi)$ and for all terms z in ϕ, $\sigma(\mathbf{t}_i)(z) = \sigma_D(i)(\mathcal{D}(z))$. The proof works by induction on the structure of ψ and z.

In the base cases, when z is equal to either a constant or a variable, $\sigma(\mathbf{t}_i)(z) = \sigma_D(i)(z)$ by the definition of σ_D.

In the recursive cases in which $\mathcal{D}$ is applied to Boolean connectives and functions, the proof follows immediately from the inductive hypothesis.

We detail the proof in the other cases focusing on the future operators, while the cases of past operators are similar.

- **Case $\psi = \phi_1 \mathbf{U} \phi_2$.**
  - If $\sigma, \mathbf{t}_i \models \phi_1 U \phi_2$, then either $\sigma, \mathbf{t}_i \models \phi_2$ or there exists $t' > \mathbf{t}_i$ such that $\sigma, t \models \phi_2$ and for all t'', $\mathbf{t}_i \le t'' < t'$, $\sigma, t'' \models \phi_1$. In the first case, $\sigma_D(i) \models \mathcal{D}(\phi_2)$ by induction. In the second case, $t' \in I_j$ for some j. Since σ is fine, $\sigma, \mathbf{t}_j \models \phi_2$ and, if I_j is open then $\sigma, \mathbf{t}_j \models \phi_1$ too. Thus, by induction, $\sigma_D, j \models (\phi_2 \wedge (\phi_1 \vee \iota)$, and thus $\sigma_D, i \models \mathcal{D}(\phi_1 U \phi_2)$.
  - If $\sigma_D, i \models \mathcal{D}(\phi_1 U \phi_2)$, then either $\sigma_D, i \models \mathcal{D}(\phi_2)$ or there exists $j > i$ such that $\sigma_D, j \models \mathcal{D}(\phi_2) \wedge (\mathcal{D}(\phi_1) \vee \iota)$ and for all k, $i \le k < j$, $\sigma_D, k \models \mathcal{D}(\phi_1)$. By induction, either $\sigma, \mathbf{t}_i \models \phi_2$ or $\sigma, \mathbf{t}_j \models \phi_2$ and for all k, $i \le k < j$, $\sigma, \mathbf{t}_k \models \phi_1$. Moreover, if I_j is open, then $\sigma, \mathbf{t}_j \models \phi_1$. Since σ is fine, $\sigma, t'' \models \phi_1$ for all t'', $\mathbf{t}_i \le t'' < \mathbf{t}_j$. Thus, $\sigma, t \models \phi_1 U \phi_2$.
- **Case $\mathbf{z} = \mathbf{u}@\tilde{\mathbf{F}}(\psi)$.**
  - Suppose there exists $t' > \mathbf{t}_i$ such that $\sigma, t' \models \psi$ and for all t'', $\mathbf{t}_i < t'' < t'$, $\sigma, t'' \not\models \psi$. Thus $\sigma(t)(z) = \sigma(t')(u)$. Let $t' \in I_j$ for some $j \ge i$. Since σ is fine for ψ and for all t'', $\mathbf{t}_i < t'' < t'$, $\sigma, t'' \not\models \psi$, then I_j is singular and $\mathbf{t}_j = t'$, thus $z = \sigma(\mathbf{t}_j)(u)$. Moreover, by induction, $\sigma_D, j \models \mathcal{D}(\psi)$ and $\sigma, k \not\models \mathcal{D}(\psi)$ for $i < k < j$. Thus, $\sigma_D(i)(\mathcal{D}(u)@\tilde{F}(\mathcal{D}(\psi))) = \sigma_D(j)(\mathcal{D}(u)) = \sigma(\mathbf{t}_j)(u)$. Moreover, for k, $i \le k \le j$, $\sigma_D, k \not\models (\neg\iota \wedge \mathcal{D}(\psi))$. Thus, $\sigma_D(i)(\mathcal{D}(z)) = \sigma_D(i)(\mathcal{D}(u)@\tilde{F}(\mathcal{D}(\psi))) = \sigma(t)(z)$.
  - Similarly, suppose there exists $t' \le \mathbf{t}_i$ such that $\sigma, t' \models \tilde{X}\psi$ and for all t'', $\mathbf{t}_i < t'' < t'$, $\sigma, t'' \not\models \psi$. Thus $z = \sigma(t')(u)$. If $t' = \mathbf{t}_i$ and I_i is open, then, since σ is fine for ψ, $\sigma, \mathbf{t}_i \models \psi$. By induction $\sigma_D, i \models \mathcal{D}(\psi)$ and thus $\sigma_D(i)(\mathcal{D}(z)) = \sigma(\mathbf{t}_i)(z)$. Similarly, if $t' = \mathbf{t}_i$ and I_i is singular, then I_{i+1} is open $\sigma, \mathbf{t}_{i+1} \models \psi$. By induction $\sigma_D, i+1 \models \mathcal{D}(\psi)$ and thus $\sigma_D(i)(\mathcal{D}(z)) = \sigma(\mathbf{t}_i)(z)$. If instead $t' > \mathbf{t}_i$, let $t' \in I_j$ for some $j > i$. Since σ is fine for ψ and for all t'', $\mathbf{t}_i < t'' < t'$, $\sigma, t'' \not\models \psi$, then I_j is singular, $\mathbf{t}_j = t'$, thus I_{j+1} is open, $\sigma, \mathbf{t}_{j+1} \models \psi$ and $\sigma(\mathbf{t}_i)(z) = \sigma(\mathbf{t}_j)(u)$. Moreover, by induction, $\sigma_D, j \models \mathcal{D}(\tilde{X}\psi)$ and $\sigma, k \not\models \mathcal{D}(\psi)$ for $i < k < j$. Thus, $\sigma_D(i)(\mathcal{D}(u)@\tilde{F}(\mathcal{D}(\psi) \vee X(\neg\iota \wedge \mathcal{D}(\psi)))) = \sigma_D(j)(\mathcal{D}(u)) = \sigma(\mathbf{t}_j)(u)$.
  - Considering now the opposite direction, suppose $\sigma_D, i \models \neg\iota \wedge \mathcal{D}(\psi)$. Then $\sigma_D(i)(\mathcal{D}(z)) = \sigma_D(i)(\mathcal{D}(u))$, which by induction is equal to $\sigma(\mathbf{t}_i)(u)$. Also by induction, $\sigma, \mathbf{t}_i \models \psi$. Since I_i is open, $\sigma, \mathbf{t}_i \models \tilde{X}\psi$ and thus $\sigma(\mathbf{t}_i)(z) = \sigma(\mathbf{t}_i)(u)$.
  - Similarly, suppose $\sigma_D, i \models X(\neg\iota \wedge \mathcal{D}(\psi))$. Then $\sigma_D(i)(\mathcal{D}(z)) = \sigma_D(i)(\mathcal{D}(u))$, which by induction is equal to $\sigma(\mathbf{t}_i)(u)$. Also by induction, $\sigma, \mathbf{t}_{i+1} \models \psi$. Since I_{i+1} is open, $\sigma, \mathbf{t}_i \models \tilde{X}\psi$ and thus $\sigma(\mathbf{t}_i)(z) = \sigma(\mathbf{t}_i)(u)$.

- Finally, suppose that we are not in the previous cases and there exists $j > i$ such that $\sigma_D, j \models \mathcal{D}(\psi) \vee X(\iota \wedge \mathcal{D}(\psi))$ and for all k, $i < k < j$, $\sigma_D, k \not\models \mathcal{D}(\psi)$. Then by induction, $\sigma, \mathbf{t}_j \models \tilde{X}(\psi)$ and for all k, $i < k < j$, $\sigma, \mathbf{t}_k \not\models \psi$. Thus, since interval I_i is singular or $\mathbf{t}_i \not\models \psi$ (considered in the previous cases), for all t'', $\mathbf{t}_i < t'' < \mathbf{t}_j$, $\sigma, t'' \not\models \psi$. Thus $\sigma(\mathbf{t}_i)(z) = \sigma(\mathbf{t}_j)(u)$.

It is routine to prove the cases of X and $\tilde{X}$ and we can conclude that $\sigma_D \models \mathcal{D}(\phi)$.

Finally, $\sigma_D \models \psi_{time}$: in fact, $\sigma_D \models time = 0 \wedge G(next(time) - time = \delta)$ by definition of σ_D; the rest of ψ_{time} is trivially satisfied because σ is fine for ϕ.

In order to prove the other direction of the theorem, suppose that there exists σ with discrete time such that $\sigma \models \mathcal{D}(\phi)$. Then, we can build a σ_C with super-dense time such that $\sigma_C \models \phi$ as follows. Let $t_i = \sum_{0 \leq h < i} \delta_h$, where δ_h is the value of the variable δ in the h-th step of σ. $I_i := [t_i, t_i]$ if $\sigma, i \models \iota$; otherwise $I_i := (t_{i-1}, t_{i+1})$. Let $\sigma(t)(v) = \sigma(i)(v)$ for every $t \in I_i$.

We prove that σ_C is fine for ϕ. Let M be the first-order structure of σ_C. Let us consider an open interval $I_i = (t_{i-1}, t_{i+1})$. For every $t, t' \in I_i$, $M(t)(x) = M(t')(x)$ and $M(t)(c) = M(t')(c)$, and thus, by induction, for every term in form u according to the grammar of *XLTL-EF*, $M(t)(u) = M(t')(u)$. Thus, for every predicate in the form $p(u_1, \ldots, u_n)$, $\sigma_C, t \models p(u_1, \ldots, u_n)$ iff $\sigma_C, t' \models p(u_1, \ldots, u_n)$. Moreover, for every term cu, the interpretation $M(cu)$ is constant; since $\sigma \models \psi_{time}$, if $\sigma_C, t_i \models time \leq cu$ then $\sigma_C, t_{i+1} \models time \leq cu$ and if $\sigma_C, t_i \models time \geq cu$ then $\sigma_C, t_{i-1} \models time \geq cu$; since $t_{i-1} < t_i < t_{i+1}$, then either $t_i < M(cu)$ and so for all $t \in I_i$ $t_{i-1} < t < t_{i+1} \leq M(cu)$, or $t_i > M(cu)$ and thus so for all $t \in I_i$ $M(cu) \leq t_{i-1} < t < t_{i+1}$. Thus, for every predicate in the form $time \bowtie cu$, for all t, t', $sigma_C, t \models time \bowtie cu$ iff $sigma_C, t' \models time \bowtie cu$. The proof for the remaining predicates in the form tu is similar, taking into account that $time@\tilde{F}(\phi)$ and $time@\tilde{P}(\phi)$ are either constant within I_i (when ϕ does not hold in I_i) or they are equivalent to $time$ (when ϕ holds in I_i).

Finally, it is routine to prove that $\sigma_C \models \phi$. □

7. Removing event freezing functions

In the following, we assume that satisfiability is restricted to traces with discrete time. If the term $u@\tilde{F}(\phi)$ occurs in a formula ψ, we can obtain a formula $\mathcal{R}(\psi, u@\tilde{F}(\phi))$ equisatisfiable to ψ, where the term $u@\tilde{F}(\phi)$ has been replaced with a fresh variable $p_{u@\tilde{F}(\phi)}$. More specifically,

$$\begin{aligned}\mathcal{R}(\psi, u@\tilde{F}(\phi)) :=& \psi[p_{u@\tilde{F}(\phi)}/u@\tilde{F}(\phi)] \wedge \\ & G(X\phi \rightarrow p_{u@\tilde{F}(\phi)} = next(u)) \wedge \\ & G(next(p_{u@\tilde{F}(\phi)}) \neq p_{u@\tilde{F}(\phi)} \rightarrow X\phi)\end{aligned}$$

$\mathcal{R}(\psi, u@\tilde{F}(\phi))$ is a formula on an extended set of variables, namely, if ϕ is a formula over variables V, then $\mathcal{R}(\psi, u@\tilde{F}(\phi))$ is a formula over $V \cup \{p_{u@\tilde{F}(\phi)}\}$, where $p_{u@\tilde{F}(\phi)}$ does not occur in ψ. However, the value of $p_{u@\tilde{F}(\phi)}$ is uniquely determined by a trace over V. In other words, given a trace σ over V, we can define a trace $\mathcal{R}(\sigma, u@\tilde{F}(\phi))$ over $V \cup \{p_{u@\tilde{F}(\phi)}\}$ such that $\sigma \models \phi$ iff $\mathcal{R}(\sigma, u@\tilde{F}(\phi)) \models \mathcal{R}(\psi, u@\tilde{F}(\phi))$. $\mathcal{R}(\sigma, u@\tilde{F}(\phi))$ is simply defined as follows:

$$\mathcal{R}(\sigma, u@\tilde{F}(\phi))(t)(x) = \sigma(t)(x), x \in V$$

$$\mathcal{R}(\sigma, u@\tilde{F}(\phi))(t)(p_{u@\tilde{F}(\phi)}) = \sigma(t)(u@\tilde{F}(\phi))$$

Theorem 3. *ψ and $\mathcal{R}(\psi, u@\tilde{F}(\phi))$ are equisatisfiable.*

Proof. We first prove that, if $\sigma \models \psi$, then $\mathcal{R}(\sigma, u@\tilde{F}(\phi)) \models \mathcal{R}(\psi, u@\tilde{F}(\phi))$.

Let us assume that $\sigma \models \phi$.

Given the definition of $\mathcal{R}(\sigma, u@\tilde{F}(\phi))$, the prophecy variable $p_{u@\tilde{F}(\phi)}$ is given the value of the term $u@\tilde{F}(\phi)$, and thus $\mathcal{R}(\sigma, u@\tilde{F}(\phi)) \models \psi[p_{u@\tilde{F}(\phi)}/u@\tilde{F}(\phi)]$.

For every t, if $\sigma, t \models F(\phi)$, then there exists $t' > t$ such that $\sigma, t' \models \phi$ and for all t'', $t < t'' < t'$, $\sigma, t'' \not\models \phi$. Thus, $\sigma(t')(u@\tilde{F}(\phi)) = \sigma(t'')(u@\tilde{F}(\phi)) = \sigma(t')(u)$. Thus $\sigma, t \models (X\phi \rightarrow p_{u@\tilde{F}(\phi)} = next(u)) \wedge (next(p_{u@\tilde{F}(\phi)}) \neq p_{u@\tilde{F}(\phi)} \rightarrow X\phi)$.

Vice versa, let us assume that $\sigma \models \mathcal{R}(\psi, u@\tilde{F}(\phi))$ and prove that $\sigma' \models \phi$, where σ' is obtained from σ by assigning $\sigma'(def_{u@\tilde{F}(\phi)}) := \sigma(p_{u@\tilde{F}(\phi)})$.

It is sufficient to prove that $\sigma(t)(p_{u@\tilde{F}(\phi)}) = \sigma'(t)(u@\tilde{F}(\phi))$.

Let us assume that there exists $t' > t$ such that, for all t'', $t < t'' < t'$, $\sigma, t'' \not\models \phi$ and $\sigma, t' \models \phi$; thus, $\sigma(t)(u@\tilde{F}(\phi)) = \sigma(t')(u)$. Since $\sigma, t'' \models (X\phi \rightarrow p_{u@\tilde{F}(\phi)} = next(u)) \wedge (next(p_{u@\tilde{F}(\phi)}) \neq p_{u@\tilde{F}(\phi)} \rightarrow X\phi)$ for all t'', $\sigma(t' - 1)(p_{u@\tilde{F}(\phi)}) = \sigma(t')(u)$ and $\sigma(t'' - 1)(p_{u@\tilde{F}(\phi)}) = \sigma(t')(p_{u@\tilde{F}(\phi)})$ for all t'', $t < t'' < t'$. Thus $\sigma(t)(p_{u@\tilde{F}(\phi)}) = \sigma(t')(u)$.

If such t' does not exist, then $\sigma'(t)(u@\tilde{F}(\phi)) = \sigma'(t)(def_{u@\tilde{F}(\phi)}) = \sigma(t)(p_{u@\tilde{F}(\phi)})$. This concludes the proof. □

We similarly remove past event freezing operator $@\tilde{P}$ with the following rule:

$$\begin{aligned}\mathcal{R}(\psi, u@\tilde{P}(\phi)) :=& \psi[p_{u@\tilde{P}(\phi)}/u@\tilde{P}(\phi)]\wedge\\ & G(\phi \rightarrow next(p_{u@\tilde{P}(\phi)}) = u)\wedge\\ & G(next(p_{u@\tilde{P}(\phi)}) \neq p_{u@\tilde{P}(\phi)} \rightarrow \phi)\end{aligned}$$

8. Experimental evaluation

We have implemented the techniques described in the previous sections within nuXmv [32], a state-of-the-art symbolic model checker for finite- and infinite-state transition systems. Our implementation supports various first-order theories (thanks to its SMT-based verification engine) such as (combinations of) arithmetic, bit-vectors, arrays and all the models of time mentioned above (discrete, dense and super-dense). It supports *XLTL-EF* as well as the fragment of $MTL_{0,\infty}$ that uses only closed intervals. We remark that, as described in the previous sections, the interval bounds on $MTL_{0,\infty}$ formulas are not required to be numeric constants, but we support arbitrary expressions over parameters of the model. Satisfiability checking is reduced to the problem of checking the emptiness of the language of a symbolic transition system, which is then solved with the algorithm that combines IC3 with Implicit Predicate Abstraction (IC3IA) [33] with k-liveness [34] or with the k-Zeno algorithm [19]. The k-Zeno algorithm builds and extends the algorithm based on IC3IA and k-liveness. The choice of the solving algorithm depends on whether we are interpreting the formula over discrete time (in which case we use IC3IA and k-liveness) or dense/super-dense time (in which case we use k-Zeno). The two algorithms are combined with Bounded Model Checking (BMC) [35] in order to detect satisfiable instances and then compute a witness trace.

In the experimental evaluation we focused on satisfiability checking over super-dense time, using both *XLTL-EF* and $MTL_{0,\infty}$ formulas. Our benchmark set consists of 161 formulas, mostly testing the semantics of the new event freezing operators, the bounded *MTL* operators and the difference of their behavior when using super-dense or dense time.[2] In the analyzed formulas we used the symbols p and q for parameters, a, b and c for Boolean variables (note that we are using Boolean variables as atoms, writing for example a instead of $a = \top$), x and y for real-valued variables and d for a discrete Boolean variable (i.e., variables that can change only in discrete steps[3]). The benchmark set also includes some "scalable" problems, in which the same template formula is instantiated multiple times using different numeric constants and/or parameters. For instance, we consider six instances of the following template, in which the parameters p and q are instantiated with different values or simply left as parameters:

$$(p > 0 \wedge q > 0) \rightarrow ((G(a \rightarrow F_{[0,p]}b) \wedge G(b \rightarrow F_{[0,q]}c)) \rightarrow G(a \rightarrow F_{[0,p+q]}c)) \tag{15}$$

Finally, we also included variants of the sensor example described above.

In the experimental evaluation we analyzed all the formulas in the benchmark set with our implementation within nuXmv, using either IC3IA k-liveness or k-Zeno, depending on whether we are interpreting the formulas over discrete or dense/super-dense time. Both algorithms are interleaved by BMC, which is used to compute counterexample traces for invalid properties. As discussed in §2, we are not aware of any other tool that supports the logics defined in this paper. In particular, MigthyL [24] uses a (discrete) point-wise semantics for *MTL*, while Atmoc [36] and Zot [25] only support bounded satisfiability checking.

We ran our experimental evaluation on a machine equipped with a 2.67 GHz Intel® Xeon® X5650 CPU, running Scientific Linux 7.3. We used a timeout of 120 seconds for each run.

Our implementation within nuXmv and all the needed files to reproduce this experimental evaluation are available at https://es.fbk.eu/people/tonetta/papers/infcomp18.

The complete set of formulas used, the corresponding verification results and the solving time (in seconds) are reported in Tables 1 and 2. In particular, Table 1 reports the set of formulas expected to hold, while Table 2 reports the ones expected to not hold. We remark that nuXmv could decide the satisfiability of all 161 formulas in the considered benchmark set. Most (104 out of 161) formulas could be decided in less than one second. The hardest problem took about 18.2 seconds, and corresponds to a valid problem instance of the template (15) in which $p = 10$ and $q = 10$. The results on the scalable instances confirm also the intuition that the performances are largely independent from the concrete values of the numeric constants used, thanks to the use of fully-symbolic techniques based on SMT. More interestingly, we also found that using parametric bounds as opposed to concrete ones does not cause a significant performance impact in general (at least for the problems we considered): the parametric instances of (15) are solved in less than 15 seconds. For the sensor example presented in Section 4.4, the parametric and the "concrete" versions of the formulas are proved in almost the same amount of time.

We executed our implementation within nuXmv on the invalid instances also using Bounded Model Checking alone to search only for satisfying models. Using this configuration all instances were solved almost instantaneously. We compared

[2] Note that, given a formula φ over super-dense time, we can obtain the equivalent one on dense time simply by conjoining φ with $G\tilde{X}\top$.

[3] nuXmv, like other tools (e.g. Atmoc [36]), natively support this kind of variables, although they can be derived from the normal variables in *LTL-EF* by adding the constraint $G(d = d@F(X\top))$.

Table 1
Some valid *MTL* properties and their verification results.

Formula	Valid	Time
$G((d \wedge \tilde{X}\top) \rightarrow \tilde{X}(\tilde{Y}d))$	Yes	0.01
$(G\tilde{X}\top) \rightarrow ((Gd) \vee (G\neg d))$	Yes	0.0
$((G_{[0,p]}d) \wedge F\neg d) \rightarrow (\mathit{time\_until}(\neg(H_{[0,p]}d)) > p)$	Yes	2.69
$((\neg b \wedge \neg bUb) \wedge (\mathit{time}@\tilde{F}b) = 0) \rightarrow X\top$	Yes	0.19
$G(((Xb) \rightarrow (x@\tilde{F}b) = \mathit{next}(x)))$	Yes	0.09
$G(((X\top) \rightarrow (x@\tilde{F}\top) = \mathit{next}(x)))$	Yes	0.0
$((G(a \rightarrow Fb)) \wedge (G(b \rightarrow F_{[0,1]}c))) \rightarrow G(a \rightarrow Fc)$	Yes	0.19
$((Fb) \wedge (G(b \rightarrow F_{[0,1]}c))) \rightarrow (Fc)$	Yes	0.1
$((Fb) \wedge (G(b \rightarrow Fc))) \rightarrow (Fc)$	Yes	0.0
$(F_{[0,1]}c) \rightarrow (Fc)$	Yes	0.1
$(\neg bUb) \rightarrow Fb$	Yes	0.0
$Fb \rightarrow (\neg bU(b \vee \tilde{X}b))$	Yes	0.0
$(\tilde{X}(\neg b \wedge Fb)) \rightarrow (\mathit{time}@\tilde{F}b) \neq 0$	Yes	0.2
$G((\mathit{time\_until}(b) = p \wedge \mathit{time\_until}(\mathit{time\_until}(b) = p) = q) \rightarrow (\mathit{time\_until}(b) = p + q \vee \mathit{time\_until}(b) \leq p))$	Yes	0.0
$G((F_{[0,0]}b) \;\leftrightarrow (X\top Ub))$	Yes	1.0
$G(O_{[0,0]}b \;\leftrightarrow (Y\top Sb))$	Yes	0.29
$G((O_{[0,0]}Fb) \;\leftrightarrow (Y\top SFb))$	Yes	0.5
$(G(a \rightarrow F_{[0,1]}b) \wedge G(b \rightarrow F_{[0,1]}c)) \rightarrow G(a \rightarrow F_{[0,2]}c)$	Yes	12.89
$(G(a \rightarrow F_{[0,10]}b) \wedge G(b \rightarrow F_{[0,10]}c)) \rightarrow G(a \rightarrow F_{[0,20]}c)$	Yes	18.19
$(G(a \rightarrow F_{[0,100]}b) \wedge G(b \rightarrow F_{[0,10]}c)) \rightarrow G(a \rightarrow F_{[0,110]}c)$	Yes	17.09
$(G(a \rightarrow F_{[0,100]}b) \wedge G(b \rightarrow F_{[0,10000]}c)) \rightarrow G(a \rightarrow F_{[0,10100]}c)$	Yes	14.09
$(p > 0) \rightarrow ((G(a \rightarrow F_{[0,p]}b) \wedge G(b \rightarrow F_{[0,p]}c)) \rightarrow G(a \rightarrow F_{[0,2*p]}c))$	Yes	14.49
$(p > 0 \wedge q > 0) \rightarrow ((G(a \rightarrow F_{[0,p]}b) \wedge G(b \rightarrow F_{[0,q]}c)) \rightarrow G(a \rightarrow F_{[0,p+q]}c))$	Yes	13.79
$(G(a \rightarrow (Fb \wedge (b \vee \mathit{time\_until}(b) \leq p))) \wedge G(b \rightarrow (Fc \wedge (c \vee \mathit{time\_until}(c) \leq p)))) \rightarrow G(a \rightarrow (Fc \wedge (c \vee \mathit{time\_until}(c) \leq 2 * p)))$	Yes	7.4
$(G(x = (y@\tilde{P}c)) \wedge c \wedge G(\neg c \rightarrow G\neg c) \wedge (b \wedge G(b \rightarrow (\mathit{time\_until}(b) = 1 \wedge \tilde{X}(\neg b \wedge \neg bUb)))) \wedge G(a \;\leftrightarrow (x@\tilde{P}b) = ((x@\tilde{P}b)@\tilde{P}b))) \rightarrow G(\neg c \rightarrow F_{[0,2]}a)$	Yes	6.79
$(G(x = (y@\tilde{P}c)) \wedge c \wedge G(\neg c \rightarrow G\neg c) \wedge (b \wedge G(b \rightarrow (\mathit{time\_until}(b) = 10 \wedge \tilde{X}(\neg b \wedge \neg bUb)))) \wedge G(a \;\leftrightarrow (x@\tilde{P}b) = ((x@\tilde{P}b)@\tilde{P}b))) \rightarrow G(\neg c \rightarrow F_{[0,20]}a)$	Yes	6.8
$(G(x = (y@\tilde{P}c)) \wedge c \wedge G(\neg c \rightarrow G\neg c) \wedge (b \wedge G(b \rightarrow (\mathit{time\_until}(b) = 100 \wedge \tilde{X}(\neg b \wedge \neg bUb)))) \wedge G(a \;\leftrightarrow (x@\tilde{P}b) = ((x@\tilde{P}b)@\tilde{P}b))) \rightarrow G(\neg c \rightarrow F_{[0,200]}a)$	Yes	6.9
$(G(x = (y@\tilde{P}c)) \wedge c \wedge G(\neg c \rightarrow G\neg c) \wedge (p > 0 \wedge b \wedge G(b \rightarrow (\mathit{time\_until}(b) = p \wedge \tilde{X}(\neg b \wedge \neg bUb)))) \wedge G(a \;\leftrightarrow (x@\tilde{P}b) = ((x@\tilde{P}b)@\tilde{P}b))) \rightarrow G(\neg c \rightarrow F_{[0,2*p]}a)$	Yes	6.9
$(G(\neg c \rightarrow y = 0) \wedge G(\neg c \rightarrow G\neg c) \wedge (p > 0 \wedge b \wedge G(b \rightarrow (\mathit{time\_until}(b) = 1 \wedge \tilde{X}(\neg b \wedge \neg bUb)))) \wedge G(a \;\leftrightarrow (y@\tilde{P}b) = 0)) \rightarrow G(\neg c \rightarrow F_{[0,2]}a)$	Yes	1.69
$(G(\neg c \rightarrow y = 0) \wedge G(\neg c \rightarrow G\neg c) \wedge (p > 0 \wedge b \wedge G(b \rightarrow (\mathit{time\_until}(b) = 10 \wedge \tilde{X}(\neg b \wedge \neg bUb)))) \wedge G(a \;\leftrightarrow (y@\tilde{P}b) = 0)) \rightarrow G(\neg c \rightarrow F_{[0,20]}a)$	Yes	1.69
$(G(\neg c \rightarrow y = 0) \wedge G(\neg c \rightarrow G\neg c) \wedge (p > 0 \wedge b \wedge G(b \rightarrow (\mathit{time\_until}(b) = 100 \wedge \tilde{X}(\neg b \wedge \neg bUb)))) \wedge G(a \;\leftrightarrow (y@\tilde{P}b) = 0)) \rightarrow G(\neg c \rightarrow F_{[0,200]}a)$	Yes	1.69
$(G(\neg c \rightarrow y = 0) \wedge G(\neg c \rightarrow G\neg c) \wedge (p > 0 \wedge b \wedge G(b \rightarrow (\mathit{time\_until}(b) = p \wedge \tilde{X}(\neg b \wedge \neg bUb)))) \wedge G(a \;\leftrightarrow (y@\tilde{P}b) = 0)) \rightarrow G(\neg c \rightarrow F_{[0,2*p]}a)$	Yes	1.1
$\neg(\tilde{Y}\top)$	Yes	0.0
$(X\neg a) \rightarrow (\neg Xa)$	Yes	0.0
$(\tilde{X}\neg a) \rightarrow (\neg\tilde{X}a)$	Yes	0.0
$(a) \rightarrow ((XG(\mathit{time\_since}(a) \geq 0)) \vee (\tilde{X}G(\mathit{time\_since}(a) \geq 0)))$	Yes	0.29
$G((Ya) \rightarrow \mathit{time\_since}(a) = 0)$	Yes	0.09
$G((\tilde{Y}a) \rightarrow \mathit{time\_since}(a) = 0)$	Yes	0.1
$(GFa) \rightarrow G(\mathit{time\_until}(a) \geq 0)$	Yes	0.2
$G((Xa) \rightarrow \mathit{time\_until}(a) = 0)$	Yes	0.1
$G((\tilde{X}a) \rightarrow \mathit{time\_until}(a) = 0)$	Yes	0.1
$G((a \vee Xa) \rightarrow (F_{[0,0]}a))$	Yes	0.1
$G((F_{[0,0]}a) \rightarrow (F_{[0,p]}a))$	Yes	0.0
$G((F_{[0,0]}a) \rightarrow (F_{[0,\infty)}a))$	Yes	0.9
$(q \geq p) \rightarrow G((F_{[0,p]}a) \rightarrow (F_{[0,q]}a))$	Yes	0.2
$G((F_{[0,p]}a) \rightarrow (F_{[0,\infty)}a))$	Yes	1.09
$(q \geq p) \rightarrow G((F_{[q,\infty)}a) \rightarrow (F_{[p,\infty)}a))$	Yes	3.2
$G((Fa) \rightarrow (F_{[0,\infty)}a))$	Yes	0.3
$G((F_{[0,\infty)}a) \rightarrow (Fa))$	Yes	0.3
$G((F_{[0,\infty)}a) \rightarrow (F_{[0,p]}a \vee F_{[p,\infty)}a))$	Yes	5.6
$G((F_{[0,p]}a \vee F_{[p,\infty)}a) \rightarrow (F_{[0,\infty)}a))$	Yes	2.6
$(q \geq p) \rightarrow G((G_{[0,q]}a) \rightarrow (G_{[0,p]}a))$	Yes	0.29
$G((G_{[0,\infty)}a) \rightarrow (G_{[0,p]}a))$	Yes	1.09
$(q \geq p) \rightarrow G((G_{[p,\infty)}a) \rightarrow (G_{[q,\infty)}a))$	Yes	2.69
$G((Ga) \rightarrow (G_{[0,\infty)}a))$	Yes	0.1
$G((G_{[0,\infty)}a) \rightarrow (Ga))$	Yes	0.09
$G((G_{[0,p]}a) \wedge (G_{[p,\infty)}a) \rightarrow (G_{[0,\infty)}a))$	Yes	5.7
$G((G_{[0,\infty)}a) \rightarrow (G_{[0,p]}a) \wedge (G_{[p,\infty)}a))$	Yes	3.0
$G(((G_{[0,p]}a) \wedge (G_{[p,\infty)}\neg a)) \rightarrow \bot)$	Yes	5.79

(*continued on next page*)

Table 1 (*continued*)

Formula	Valid	Time
$G((a \vee Ya) \rightarrow (O_{[0,0]}a))$	Yes	0.1
$G((O_{[0,0]}a) \rightarrow (O_{[0,p]}a))$	Yes	0.0
$G((O_{[0,0]}a) \rightarrow (O_{[0,\infty)}a))$	Yes	1.2
$(q \geq p) \rightarrow G((O_{[0,p]}a) \rightarrow (O_{[0,q]}a))$	Yes	0.3
$G((O_{[0,p]}a) \rightarrow (O_{[0,\infty)}a))$	Yes	1.0
$(q \geq p) \rightarrow G((O_{[q,\infty)}a) \rightarrow (O_{[p,\infty)}a))$	Yes	3.59
$G((Oa) \rightarrow (O_{[0,\infty)}a))$	Yes	0.2
$G((O_{[0,\infty)}a) \rightarrow (O_{[0,p]}a \vee O_{[p,\infty)}a))$	Yes	6.0
$G((O_{[0,p]}a \vee O_{[p,\infty)}a) \rightarrow (O_{[0,\infty)}a))$	Yes	6.39
$(q \geq p) \rightarrow G((H_{[0,q]}a) \rightarrow (H_{[0,p]}a))$	Yes	0.2
$G((H_{[0,\infty)}a) \rightarrow (H_{[0,p]}a))$	Yes	1.7
$(q \geq p) \rightarrow G((H_{[p,\infty)}a) \rightarrow (H_{[q,\infty)}a))$	Yes	6.18
$G((Ha) \rightarrow (H_{[0,\infty)}a))$	Yes	0.1
$G((H_{[0,\infty)}a) \rightarrow (Ha))$	Yes	0.1
$G((H_{[0,p]}a) \wedge (H_{[p,\infty)}a) \rightarrow (H_{[0,\infty)}a))$	Yes	6.9
$G((H_{[0,\infty)}a) \rightarrow (H_{[0,p]}a) \wedge (H_{[p,\infty)}a))$	Yes	5.19
$(Fa) \leftrightarrow (FO_{[0,\infty)}a)$	Yes	0.29
$(q > p \wedge p \geq 0) \rightarrow ((F_{[0,p]}(Ga)) \rightarrow (G_{[q,\infty)}a))$	Yes	3.3
$G(\neg(F_{[0,0]}O\neg a) \leftrightarrow (G_{[0,0]}Ha))$	Yes	1.1
$(G\tilde{X}\top) \rightarrow G(b \leftrightarrow F_{[0,0]}b)$	Yes	0.29
$(G\tilde{X}\top) \rightarrow G(b \leftrightarrow O_{[0,0]}b)$	Yes	0.2
$(G\tilde{X}\top) \rightarrow G(b \leftrightarrow G_{[0,0]}b)$	Yes	0.3
$(G\tilde{X}\top) \rightarrow G(b \leftrightarrow H_{[0,0]}b)$	Yes	0.3

Table 2
Some invalid *MTL* properties and their verification results.

Formula	Valid	Time	
		IC3IA	BMC only
$(\tilde{Y}\top)$	No	0.1	0.0
$(\neg Xa) \rightarrow (X\neg a)$	No	0.09	0.0
$(\neg\tilde{X}a) \rightarrow (\tilde{X}\neg a)$	No	0.1	0.0
$G(a \rightarrow \mathit{time\_since}(a) = 0)$	No	0.2	0.0
$G((Ya) \rightarrow \mathit{time\_since}(a) > 0)$	No	0.19	0.0
$G((\tilde{Y}a) \rightarrow \mathit{time\_since}(a) > 0)$	No	0.3	0.0
$F(\mathit{time\_since}(a) < 0)$	No	0.2	0.0
$G(a \rightarrow \mathit{time\_until}(a) = 0)$	No	0.19	0.0
$G((Xa) \rightarrow \mathit{time\_until}(a) > 0)$	No	0.2	0.0
$G((\tilde{X}a) \rightarrow \mathit{time\_until}(a) > 0)$	No	0.19	0.0
$F(\mathit{time\_until}(a) < 0)$	No	0.1	0.0
$G((F_{[0,0]}a) \rightarrow (a \vee Xa))$	No	0.5	0.0
$G((F_{[0,p]}a) \rightarrow (F_{[0,0]}a))$	No	0.39	0.0
$G((F_{[0,\infty)}a) \rightarrow (F_{[0,0]}a))$	No	1.0	0.0
$G((F_{[0,q]}a) \rightarrow (F_{[0,p]}a))$	No	0.7	0.0
$G((F_{[0,\infty)}a) \rightarrow (F_{[0,p]}a))$	No	1.4	0.0
$G((F_{[p,\infty)}a) \rightarrow (F_{[q,\infty)}a))$	No	3.0	0.0
$G((G_{[0,p]}a) \rightarrow (G_{[0,q]}a))$	No	0.79	0.0
$G((G_{[0,p]}a) \rightarrow (G_{[0,\infty)}a))$	No	1.7	0.0
$G((G_{[q,\infty)}a) \rightarrow (G_{[p,\infty)}a))$	No	2.8	0.0
$G((O_{[0,0]}a) \rightarrow (a \vee Ya))$	No	0.4	0.0
$G((O_{[0,p]}a) \rightarrow (O_{[0,0]}a))$	No	0.39	0.0
$G((O_{[0,\infty)}a) \rightarrow (O_{[0,0]}a))$	No	1.1	0.0
$G((O_{[0,q]}a) \rightarrow (O_{[0,p]}a))$	No	0.39	0.0
$G((O_{[0,\infty)}a) \rightarrow (O_{[0,p]}a))$	No	1.0	0.0
$G((O_{[p,\infty)}a) \rightarrow (O_{[q,\infty)}a))$	No	6.1	0.09
$G((O_{[0,0]}a) \rightarrow \neg(O_{[0,0]}\neg a))$	No	1.29	0.0
$G((H_{[0,p]}a) \rightarrow (H_{[0,q]}a))$	No	0.5	0.0
$G((H_{[0,p]}a) \rightarrow (H_{[0,\infty)}a))$	No	1.49	0.0
$G((H_{[q,\infty)}a) \rightarrow (H_{[p,\infty)}a))$	No	1.9	0.09
$(G_{[q,\infty)}a) \rightarrow (F_{[0,p]}Ga)$	No	1.6	0.0
$(\mathit{time\_until}(\neg(H_{[0,p]}a)) > p) \rightarrow ((G_{[0,p]}a) \wedge F\neg a)$	No	2.8	0.0
$(\neg Xb) \rightarrow (X\neg b)$	No	0.09	0.0
$(\neg\tilde{X}b) \rightarrow (\tilde{X}\neg b)$	No	0.1	0.0
$F_{[0,\infty)}(b)$	No	0.29	0.0
$F_{[0,4]}(b)$	No	0.29	0.0
$G(((Fb) \wedge (x@\tilde{F}b) = 0) \rightarrow X\top)$	No	0.2	0.0
$G(((Fb) \wedge (\mathit{time}@\tilde{F}b) = 0) \rightarrow X\top)$	No	0.29	0.0

Table 2 (*continued*)

Formula	Valid	Time	
		IC3IA	BMC only
$G(((\neg b \wedge Fb) \wedge (\mathit{time}@\tilde{F}b) = 0) \rightarrow X\top)$	No	0.2	0.0
$G(\neg((Fb) \wedge ((x@\tilde{F}b) = 0)))$	No	0.2	0.0
$G(\neg((Fb)))$	No	0.09	0.0
$\neg((Fb) \wedge ((x@\tilde{F}b) = 0))$	No	0.1	0.0
$\neg F(b \wedge \mathit{time} = 10)$	No	0.1	0.0
$\neg(G(x = (y@\tilde{P}b)) \wedge GF(b \wedge y = 1) \wedge GF(\neg b \wedge y = 0) \wedge GFy \neq x)$	No	0.49	0.0
$Fb \rightarrow (\neg bUb)$	No	0.1	0.0
$G(a \rightarrow (Fc \wedge (c \vee (\mathit{time}@\tilde{F}c) \leq 2 * p)))$	No	0.3	0.0
$\tilde{X}((b \vee a) \rightarrow (Fc))$	No	0.1	0.0
$F_{[3,\infty)}(b)$	No	0.89	0.0
$\neg(G_{[0,3]}(((\mathit{time}@\tilde{P}b)) \leq 0))$	No	0.19	0.0
$((F_{[0,0]}b))$	No	0.09	0.0
$((O_{[0,0]}b))$	No	0.1	0.0
$\neg(G_{[0,3]}(F_{[0,1]}b))$	No	0.49	0.0
$\neg(G_{[0,3]}(O_{[0,0]}b))$	No	1.09	0.0
$\neg(G_{[0,3]}(Y\top SFb))$	No	2.4	0.0
$\neg(G_{[0,3]}(O_{[0,0]}Fb))$	No	0.4	0.0
$\neg(G_{[0,3]}(Y\top SFb) \wedge F\neg b)$	No	13.0	0.1
$\neg F_{[3,\infty)}(b)$	No	0.4	0.0
$\mathit{time\_until}(b) = 1$	No	0.19	0.0
$(\mathit{time\_until}(\mathit{time\_until}(b) = 1)) = 2$	No	0.29	0.0
$\neg(F_{[2,\infty)}b \wedge \mathit{time\_until}(\mathit{time\_until}(b) = 1) = 1 \wedge (\mathit{time}@\tilde{F}b) = 2)$	No	3.09	0.1
$\neg(b \wedge (\mathit{time\_until}(b) = 1 \wedge \tilde{X}(\neg b \wedge \neg bUb)))$	No	0.39	0.0
$\neg(b \wedge \mathit{time\_until}(b) = 1 \wedge \tilde{X}(\neg b \wedge \neg bU(b \wedge \mathit{time\_until}(b) = 1 \wedge \tilde{X}(\neg b \wedge \neg bUb))))$	No	0.59	0.0
$G(((\mathit{time\_until}(b) = p \wedge \mathit{time\_until}(\mathit{time\_until}(b) = p) = q)) \rightarrow (\mathit{time\_until}(b) = p + q \vee \mathit{time\_until}(b) < p))$	No	1.2	0.0
$\neg(GFb \wedge G(b \rightarrow (\mathit{time\_until}(b) = 1 \wedge \tilde{X}(\neg b \wedge \neg bUb))))$	No	0.3	0.0
$\neg(b \wedge G(b \rightarrow (\mathit{time\_until}(b) = 1 \wedge \tilde{X}(\neg b \wedge \neg bUb))))$	No	0.29	0.0
$\neg(GFb \wedge G(b \rightarrow (\mathit{time\_until}(b) = 1)))$	No	0.29	0.0
$\neg(b \wedge G(b \rightarrow (\mathit{time\_until}(b) = 1 \wedge \tilde{X}(\neg b \wedge \neg bUb))))$	No	0.3	0.0
$\neg(G(O_{[0,0]}b))$	No	0.3	0.0
$((GFb) \rightarrow G(\mathit{time\_until}(b) > 0))$	No	0.2	0.0
$(G\tilde{X}\top) \rightarrow ((GFb) \rightarrow G(\mathit{time\_until}(b) > 0))$	No	0.19	0.0
$G(\neg b \rightarrow \neg F_{[0,0]}b)$	No	0.3	0.0
$G(\neg b \rightarrow \neg O_{[0,0]}b)$	No	0.5	0.0
$G((O_{[0,0]}b) \rightarrow (F_{[0,0]}b))$	No	0.7	0.0
$G((F_{[0,0]}b) \rightarrow (O_{[0,0]}b))$	No	1.0	0.0
$G((H_{[0,0]}b) \rightarrow (G_{[0,0]}b))$	No	0.69	0.06
$G((G_{[0,0]}b) \rightarrow (H_{[0,0]}b))$	No	0.7	0.0
$G((a \wedge \tilde{X}\top) \rightarrow \tilde{X}(\tilde{Y}a))$	No	0.09	0.0
$(G\tilde{X}\top) \rightarrow ((Ga) \vee (G\neg a))$	No	0.1	0.0
$((G_{[0,p]}a) \wedge F\neg a) \rightarrow (\mathit{time\_until}(\neg(H_{[0,p]}a)) > p)$	No	3.09	0.0

this result with the time required by Atmoc, which also uses Bounded Model Checking, to find the satisfying model of 13 problem instances expressible in the input languages of both tools. Since Atmoc does not support parameters all *MTL* bounds in these formulas are "concrete" values. Also in these cases the Bounded Model Checking based techniques were able to find the satisfying model almost instantaneously.

9. Conclusions and future work

In this paper, we considered an extension of first-order linear-time temporal logic with two new event freezing functional symbols, which represent the value of a term at the next state in the future or last state in the past in which a formula holds. We defined the semantics in different temporal domains considering discrete time with real timestamps, dense, and super-dense (weakly-monotonic) time. We precisely characterized what we mean for "next point in the future in which ϕ holds" so that, assuming finite variability, such a point always exists also in the dense time setting. We showed that, using these event-freezing functions and an explicit variable that represents time, we can express all $MTL_{0,\infty}$ formulas, also including counting operators and parametric intervals. We provided a reduction to equisatisfiable discrete-time formulas without event freezing functions and we solved the satisfiability of the latter by SMT-based model checking. We implemented the approach in nuXmv and showed that it can effectively prove interesting properties, while many logics in the real-time setting lack of tool support.

The directions for future works are manifold. We want to integrate these techniques in OCRA [37] for contract-based reasoning; we want to extend them to encompass variables with continuous function evolution and constraints on their derivatives as in HRELTL [23]; finally, we want to apply the new logic in industrial use cases within the CITADEL project (http://citadel-project.org/) to specify complex properties of monitoring components.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. Tonetta, Linear-time temporal logic with event freezing functions, in: GandALF, 2017, pp. 195–209.
[2] A. Pnueli, The temporal logic of programs, in: FOCS, 1977, pp. 46–57.
[3] R. Alur, T. Henzinger, Real-time logics: complexity and expressiveness, Inf. Comput. 104 (1) (1993) 35–77.
[4] R. Koymans, Specifying real-time properties with metric temporal logic, Real-Time Syst. (4) (1990) 255–299.
[5] R. Alur, T. Feder, T. Henzinger, The benefits of relaxing punctuality, J. ACM 43 (1) (1996) 116–146.
[6] J. Raskin, P. Schobbens, The logic of event clocks - decidability, complexity and expressiveness, J. Autom. Lang. Comb. 4 (3) (1999) 247–286.
[7] O. Maler, D. Nickovic, Monitoring temporal properties of continuous signals, in: FORMATS-FTRTFT, 2004, pp. 152–166.
[8] R. Koymans, Specifying Message Passing and Time-Critical Systems with Temporal Logic, Lecture Notes in Computer Science, vol. 651, Springer, 1992.
[9] D. Basin, F. Klaedtke, S. Müller, Policy monitoring in first-order temporal logic, in: CAV, 2010, pp. 1–18.
[10] Z. Manna, A. Pnueli, The Temporal Logic of Reactive and Concurrent Systems - Specification, Springer, 1992.
[11] O. Lichtenstein, A. Pnueli, L. Zuck, The glory of the past, in: Logics of Programs, 1985, pp. 196–218.
[12] C. Barrett, R. Sebastiani, S. Seshia, C. Tinelli, Satisfiability modulo theories, in: Handbook of Satisfiability, 2009, pp. 825–885.
[13] S. Ghilardi, E. Nicolini, S. Ranise, D. Zucchelli, Combination methods for satisfiability and model-checking of infinite-state systems, in: CADE, 2007, pp. 362–378.
[14] J. Raskin, P. Schobbens, State clock logic: a decidable real-time logic, in: HART, 1997, pp. 33–47.
[15] T.A. Henzinger, J. Raskin, P. Schobbens, The regular real-time languages, in: ICALP, 1998, pp. 580–591.
[16] R. Alur, L. Fix, T. Henzinger, Event-clock automata: a determinizable class of timed automata, Theor. Comput. Sci. 211 (1–2) (1999) 253–273.
[17] Y. Hirshfeld, A. Rabinovich, An expressive temporal logic for real time, in: MFCS, 2006, pp. 492–504.
[18] J. Ortiz, A. Legay, P. Schobbens, Memory event clocks, in: FORMATS, 2010, pp. 198–212.
[19] A. Cimatti, A. Griggio, S. Mover, S. Tonetta, Verifying LTL properties of hybrid systems with K-liveness, in: CAV, in: LNCS, vol. 8559, Springer, 2014, pp. 424–440.
[20] J. Daniel, A. Cimatti, A. Griggio, S. Tonetta, S. Mover, Infinite-state liveness-to-safety via implicit abstraction and well-founded relations, in: CAV, 2016, pp. 271–291.
[21] S. Demri, R. Lazic, LTL with the freeze quantifier and register automata, ACM Trans. Comput. Log. 10 (3) (2009) 1–30.
[22] P. Bouyer, F. Chevalier, N. Markey, On the expressiveness of TPTL and MTL, Inf. Comput. 208 (2) (2010) 97–116.
[23] A. Cimatti, M. Roveri, S. Tonetta, Requirements validation for hybrid systems, in: CAV, 2009, pp. 188–203.
[24] T. Brihaye, G. Geeraerts, H. Ho, B. Monmege, MightyL: a compositional translation from MITL to timed automata, in: R. Majumdar, V. Kuncak (Eds.), Computer Aided Verification - Proceedings of the 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Part I, in: Lecture Notes in Computer Science, vol. 10426, Springer, 2017, pp. 421–440.
[25] M.M. Bersani, M. Rossi, P.S. Pietro, An SMT-based approach to satisfiability checking of MITL, Inf. Comput. 245 (2015) 72–97.
[26] R. Kindermann, T.A. Junttila, I. Niemelä, SMT-based induction methods for timed systems, in: M. Jurdzinski, D. Nickovic (Eds.), Formal Modeling and Analysis of Timed Systems – Proceedings of the 10th International Conference, FORMATS 2012, London, UK, September 18-20, 2012, in: Lecture Notes in Computer Science, vol. 7595, Springer, 2012, pp. 171–187.
[27] R. Kindermann, T.A. Junttila, I. Niemelä, Bounded model checking of an MITL fragment for timed automata, in: J. Carmona, M.T. Lazarescu, M. Pietkiewicz-Koutny (Eds.), 13th International Conference on Application of Concurrency to System Design, ACSD 2013, Barcelona, Spain, 8-10 July, 2013, IEEE Computer Society, 2013, pp. 216–225.
[28] R. Alur, T. Henzinger, Logics and models of real time: a survey, in: REX Workshop, 1991, pp. 74–106.
[29] O. Maler, Z. Manna, A. Pnueli, From timed to hybrid systems, in: J.W. de Bakker, C. Huizing, W.P. de Roever, G. Rozenberg (Eds.), Real-Time: Theory in Practice, Proceedings of REX Workshop, Mook, The Netherlands, June 3-7, 1991, in: Lecture Notes in Computer Science, vol. 600, Springer, 1991, pp. 447–484.
[30] A. Rabinovich, On the decidability of continuous time specification formalisms, J. Log. Comput. 8 (5) (1998) 669–678.
[31] A. Bouajjani, Y. Lakhnech, Temporal logic + timed automata: expressiveness and decidability, in: CONCUR, 1995, pp. 531–545.
[32] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, S. Tonetta, The nuXmv symbolic model checker, in: CAV, 2014, pp. 334–342.
[33] A. Cimatti, A. Griggio, S. Mover, S. Tonetta, IC3 modulo theories via implicit predicate abstraction, in: TACAS, in: LNCS, vol. 8413, Springer, 2014, pp. 46–61.
[34] K. Claessen, N. Sörensson, A liveness checking algorithm that counts, in: FMCAD, IEEE, 2012, pp. 52–59.
[35] A. Biere, A. Cimatti, E.M. Clarke, O. Strichman, Y. Zhu, Bounded model checking, Adv. Comput. 58 (2003) 117–148.
[36] R. Kindermann, T.A. Junttila, I. Niemelä, Bounded model checking of an MITL fragment for timed automata, in: ACSD, 2013, pp. 216–225.
[37] A. Cimatti, M. Dorigatti, S. Tonetta, OCRA: a tool for checking the refinement of temporal contracts, in: ASE, 2013, pp. 702–705.