Monitoring Temporal Logic with Clock Variables

Adrián Elgyütt, Thomas Ferrère, and Thomas A. Henzinger

IST Austria

Abstract. We solve the offline monitoring problem for timed propositional temporal logic (TPTL), interpreted over dense-time Boolean signals. The variant of TPTL we consider extends linear temporal logic (LTL) with *clock* variables and *reset* quantifiers, providing a mechanism to specify real-time constraints. We first describe a general monitoring algorithm based on an exhaustive computation of the set of satisfying clock assignments as a finite *union of zones*. We then propose a specialized monitoring algorithm for the one-variable case using a partition of the time domain based on the notion of *region equivalence*, whose complexity is linear in the length of the signal, thereby generalizing a known result regarding the monitoring of metric temporal logic (MTL). The region and zone representations of time constraints are known from timed automata verification and can also be used in the discrete-time case. Our prototype implementation appears to outperform previous discrete-time implementations of TPTL monitoring.

1 Introduction

Temporal logic monitoring [20] is a well-studied topic with multiple applications [19,23,32,17]. A *monitor* is a program that verifies the conformance of a single run of the system against the specification; generally speaking monitoring is one of the methods for ensuring that a system meets its specification.[1] There are two types of monitoring – online and offline. The online monitor runs simultaneously with the system, and is suitable for use on a production system to enforce a safety property of that system. The offline monitor verifies a trace of a finite length after the system execution/simulation, and is thus suitable for use in a testing scenario.

In discrete systems such as programs, behaviors can be formalized in linear temporal logic (LTL) [30]. Temporal logic abstracts time into so-called temporal modalities, such as *always*, denoted $\Box$, and its dual *eventually*, denoted $\Diamond$. As an example, the typical property that every request p is followed by a grant q can be written $\Box(p \rightarrow \Diamond\, q)$. In real-time systems, or in the setting of asynchronous communication, the specification not only talks of the temporal ordering of events, but also of their temporal distance. One way to specify such a distance is to

[1] While temporal logic monitoring provides less guarantees than other formal methods such as model checking, the range of applicability of monitoring techniques is wider as it does not suffer from the infamous state-explosion: for monitoring purposes, all that is needed from the system model is its ability to generate execution traces.

integrate timing constraints into temporal modalities, as done in metric temporal logic (MTL) [25]. For instance, in MTL one can write $\Diamond_{[1,2]}\, q$ to specify a trace where proposition q holds eventually within 1 to 2 time units. Another way to specify the temporal distance between events is to use dedicated variables. This approach is advocated by [3] with the introduction of *timed propositional temporal logic* (TPTL). In TPTL, timing and sequential aspects are made orthogonal by the use of dedicated *clock* variables ranging over time, enabling the clean specification of temporal objectives. A clock x is a real-valued variable that measures the time elapsed from the temporal context of a formula to the temporal context of its subformulas. For this one can use *reset* quantifiers $x.\varphi$ over a formula φ, and constraints of the form $x \leq c$ (or $x \geq c$) that compare the time elapsed from the binding quantifier with some integer constant.[2]

Over an integer (discrete) time domain, timing constraints can be emulated in LTL by nesting *next-time* operators, but such an encoding is cumbersome and exponential in nature as durations are represented in unary. Over a real (continuous) time domain, the *next-time* changes its meaning and one must use dedicated logics such as MTL or TPTL in order to specify timing constraints. In this setting, the one-clock fragment of TPTL is more expressive than MTL [9,21]. To translate MTL operators into TPTL, we only need one clock variable, with for instance $\Diamond_{[0,1]}\, p$ translating as $x.\Diamond(p \wedge x \leq 1)$. TPTL timing constraints may not translate to MTL when more than one temporal operator separates quantifiers and bound constraints, as in formula $x.\Diamond(p \wedge \Diamond(q \wedge x \leq 1))$.

The efficient handling of time variables in monitoring tasks is an important open problem, regardless of the underlying time domain. A practice similar to TPTL is indeed recommended in the standard specification language SVA [34], through the use of local variables of type *time*. SVA (or its *simple subset* [4]) can be used for model-checking, but is predominantly used in testing: simulation traces are systematically monitored against SVA specifications. The state-of-the-art online procedures for monitoring SVA incur an additional cost in the presence of time variables, which they often treat by spawning a new instance of the monitor at every possible variable assignment. To our knowledge, the complexity of the offline monitoring problem for SVA has not been studied.

In this paper, we solve the offline TPTL monitoring problem over continuous-time Boolean signals. The satisfaction of TPTL formulas can be characterized in terms of difference constraints on their free variables [28]. In this setting, our contribution is twofold. We first propose to compute such constraints in the form of a *union of zones*. The zone data structure underlies recent advances in continuous-time monitoring and pattern matching [33,6]. Our naïve zone-based implementation of TPTL monitoring appears competitive relative to the existing discrete-time implementations for TPTL monitoring of [15], based on instantiating LTL monitors for every possible value of clock variables. We then

[2] The original presentation of TPTL instead talks of *freeze* quantifiers that store the absolute time in variables x, y later compared using difference constraints $y - x \leq c$. We found it more convenient to work with clocks and associated *reset* quantifiers as in [31], although both presentations are equivalent.

propose to represent difference constraints using a partition of the time domain according to the *region equivalence*. A region is a cell in this partition, and two equivalent regions agree on the value of all subformulas. As for timed automata verification [1], this equivalence relation provides a canonical representation of the state space. The suitable inductive computation of this relation yields an algorithm with linear-time complexity relative to the trace length for monitoring the important fragment of TPTL formulas with one clock.[3]

The practical performance of our zone-based and region-based algorithms is evaluated in a prototype implementation, which we compare with tool AMT [29] as baseline. Our experiments support the theoretical complexity of the region-based algorithm, which also compares to the zone-based algorithm.

Related Work Temporal logic monitoring over continuous time is introduced by [26], who consider the logic MTL and its extension to real-valued signals called STL. Subsequently, [33] proposes an algorithm for the monitoring and matching of timed regular expressions (TRE) [5], that are regular expressions with duration constraints $\langle . \rangle_I$ requiring that the segment matching the enclosed expression also has a duration within some interval I. The work of [6] considers the monitoring of MTL with an additional time parameter standing for the horizon of the property, after which the signal is considered to end. The constructions in [33] and [6] use a representation of the time domain as a union of zones, which we also consider in our naïve implementation of TPTL monitoring. A recent related work [8] considers the monitoring of *metric dynamic logic* (MDL) formulas. This logic introduces modalities $\langle r \rangle_I \varphi$ requiring that φ should occur within timing interval I after a sequence of discrete events matching some regular expression r. The authors consider a weakly-monotonic, discrete model of time and obtain an algorithm with quasi-linear time complexity [8].

The decidability of TPTL offline monitoring over continuous-time domains was proved in [28] with a tight (relative to combined trace and formula size) reduction to difference constraints satisfiability. However, in the absence of fast difference constraints solvers, this does not necessarily provide a practical algorithm for large traces. In contrast our region-based algorithm comes with a linear-time guaranteed complexity relative to trace length. To the best of our knowledge, previous implementations of TPTL monitoring are as follows. The approach of [15] uses a monitor of LTL formulas as sub-routine, called on every possible valuation of time variables. This enables efficient monitoring of the sequential part of the property by reusing off-the-shelf LTL monitors, but the LTL monitor is called for every instantiation of clock variables. The number of LTL monitor instances may grow linearly with the trace length, and as a result this algorithm has a worst-case time complexity quadratic in the trace length [15]. The approach of [12] proceeds by incremental rewriting of TPTL semantics, based on formalization in Maude [13]. The resulting procedure seems to suffer from similar complexity in terms of its number of rewrites.

[3] This does not follow straightforwardly from [1], since TPTL does not translate to timed automata: its satisfiability over dense time is undecidable [3].

2 Background

An essential idea in offline monitoring is that the standard (future time) operators of LTL can be realized as backward-deterministic transducers. Therefore, the whole trace can be parsed once in reverse time-order using finite memory. Let us consider a discrete time domain $\mathbb{T} = \{0, 1, \ldots, n\}$. Assuming a set of atomic propositions AP, a trace w is a function $w : \mathbb{T} \to 2^{AP}$ that we denote $w = w_0 w_1 \ldots w_n$ with $w_i \subseteq AP$ for all $i \in T$. The satisfaction relation of LTL can be characterized by a recursion on the time dimension (backwards) and on the formula structure (top-down). For the *until operator* we have:

base case: $(w, n) \models \varphi_1 \,\mathcal{U}\, \varphi_2$ iff $(w, n) \models \varphi_2$;
inductive case: $(w, i-1) \models \varphi_1 \,\mathcal{U}\, \varphi_2$ iff $(w, i-1) \models \varphi_2$, or $(w, i-1) \models \varphi_1$ and $(w, i) \models \varphi_1 \,\mathcal{U}\, \varphi_2$.

Notice that the satisfaction $\models$ of $\varphi_1 \,\mathcal{U}\, \varphi_2$ at position i only depends on the satisfaction of φ_1 and φ_2 at position i, and on the satisfaction of $\varphi_1 \,\mathcal{U}\, \varphi_2$ at $i+1$.

The LTL monitoring algorithm described in [19] first evaluates the subformulas $\varphi_1, \varphi_2, \ldots, \varphi_m$ of the main formula φ at the end of the input trace w (position n). Then for all $i = n-1, \ldots, 0$ the algorithm evaluates $\varphi_1, \varphi_2, \ldots, \varphi_m$ at time i in a bottom up fashion based on values computed at position i and $i+1$. The overall process is illustrated in Figure 1.

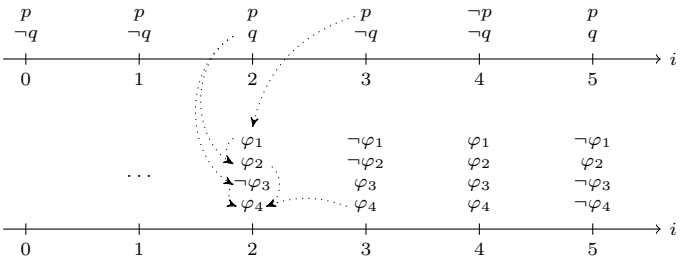

Fig. 1. Monitoring formula $(\bigcirc p \vee q) \,\mathcal{U}\, \neg q$ with subformulas $\varphi_1 \equiv \bigcirc p$, $\varphi_2 \equiv \bigcirc p \vee q$, $\varphi_3 \equiv \neg q$, and $\varphi_4 \equiv (\bigcirc p \vee q) \,\mathcal{U}\, \neg q$ by backward induction. Positions $5, 4, 3$ have been marked with satisfied subformulas, and the marking at position 2 is computed based on input values of p and q, and the values of subformulas at positions 2 and 3.

Metric temporal logic (MTL) [25] extends LTL with timed temporal modalities such as the *timed eventually*, denoted $\Diamond_I$ for timing interval I. Formula $\Diamond_{[a,b]} \varphi$ holds at time t if and only if φ holds at some time $t' \in [t+a, t+b]$. Here we consider $\mathbb{T} = [0, d] \subseteq \mathbb{R}$ to be a dense time domain. Similar to LTL, the truth value of a given formula φ is uniquely determined at time t by the truth value of its main subformulas at times $t' \geq t$.

The evolution of the truth value of a formula φ over time forms a Boolean signal, that we call *satisfaction signal*, denoted $w_\varphi[t]$ for input trace w. Monitoring

MTL offline can be done by computing the entire satisfaction signal of every subformula of φ inductively, as proposed in [27]. For $\Diamond_I$, the inductive step is as follows. Assume that w_φ has value 1 over a finite set of intervals $T_0, \ldots, T_n \subseteq \mathbb{T}$, and value 0 everywhere else. Then $w_{\Diamond_I \varphi}$ will have value 1 over intervals $T_i \ominus I$ for all $i = 0, \ldots, n$, and value 0 everywhere else.[4] For all inductive cases, satisfaction signals can be computed in linear time [27]. We illustrate the resulting algorithm in Figure 2.

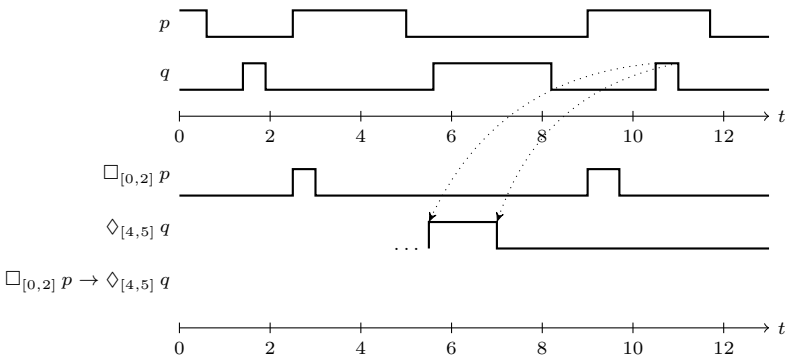

Fig. 2. Monitoring formula $\varphi = \Box_{[0,2]}\, p \rightarrow \Diamond_{[4,5]}\, q$ by inductively constructing the satisfaction signals if its subformulas. The segment of the satisfaction signal of $\Diamond_{[4,5]}\, q$ between times 5.5 and 7 is obtained from the segment of q between times 10.5 and 11.

3 Timed Propositional Temporal Logic

We call *time domain* a subset $\mathbb{T} \subseteq \mathbb{R}$ of the real line of the form $[0, d]$ with *duration* $d > 0$. Assume a set AP of propositional variables. A *trace* $w : \mathbb{T} \rightarrow 2^{AP}$ can be seen as a valuation of variables $p \in AP$ into Boolean signals over $\mathbb{T}$, which we write $w_p : \mathbb{T} \rightarrow \{0, 1\}$. The Boolean value of p at time t in trace w is denoted $w_p[t]$. The length of trace w is the minimal size of any partition of $\mathbb{T}$ into intervals over which the truth status of predicates w_p is constant relative to time for all $p \in AP$. We assume that every trace has a finite length denoted $|w|$.

Let $\mathbb{X}$ be a set of clock variables. An *environment* r is a valuation of clocks $x \in \mathbb{X}$ as elements of the time domain, written $r_x \in \mathbb{T}$.

Definition 1 (TPTL Syntax). *Formulas of TPTL are given by the following grammar:*

$$\varphi ::= p \mid x \leq c \mid x \geq c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\, \mathcal{U}\, \varphi \mid x.\varphi$$

for $p \in AP$, clock variables $x \in \mathbb{X}$, and integer constants $c \in \mathbb{N}$.

[4] The Minkowski difference $T_i \ominus I$ is by definition $\{t - s \in \mathbb{T} \ : \ t \in T_i \text{ and } s \in I\}$.

We also use shorthands such as $x < c$ for $\neg(x \geq c)$. The form '$x.$' in the formula $x.\varphi$ is called a *reset quantifier* and a formula is *closed* when all its variables are bound by a reset quantifier.

Definition 2 (TPTL Semantics). *The satisfaction of a TPTL formula φ relative to a trace w at time t under an environment r is according to the relation $\models$ between the tuple (w, t, r) and φ, inductively defined as follows:*

$$
\begin{array}{lll}
(w,t,r) \models p & \textit{iff} & w_p[t] = 1 \\
(w,t,r) \models x \leq c & \textit{iff} & t - r_x \leq c \\
(w,t,r) \models \neg\varphi & \textit{iff} & (w,t,r) \not\models \varphi \\
(w,t,r) \models \varphi \vee \psi & \textit{iff} & (w,t,r) \models \varphi \textit{ or } (w,t,r) \models \psi \\
(w,t,r) \models \varphi\,\mathcal{U}\,\psi & \textit{iff} & (w,t',r) \models \psi \textit{ for some } t' > t \textit{ such that} \\
 & & (w,t'',r) \models \varphi \textit{ for all } t'' \textit{ with } t < t'' < t' \\
(w,t,r) \models x.\varphi & \textit{iff} & (w,t,r[x \leftarrow t]) \models \varphi
\end{array}
$$

where $r[x \leftarrow t]$ is the environment that assigns t to x and agrees with r for every other clock. For any closed formula φ it holds $(w,t,r) \models \varphi$ iff $(w,t,r') \models \varphi$ for all environments r, r' and thus we simply write $(w,t) \models \varphi$ in that case. We say that w satisfies φ, written $w \models \varphi$, when $(w,0) \models \varphi$.

A clock variable x intuitively stands for the time elapsed from the temporal context of its binding reset quantifier. Observe that reset quantifiers commute with Boolean operators, that is, $x.(\varphi \vee \psi) \Leftrightarrow x.\varphi \vee x.\psi$ and $x.\neg\varphi \Leftrightarrow \neg x.\varphi$. We refer the reader to [3] for a more extensive discussion of the merits of reset (or *freeze*) quantification over existential and universal quantification in the temporal logic context.

The offline monitoring problem for TPTL, which we solve in this paper, can be stated as follows: given a formula φ and a trace w, decide whether $w \models \varphi$.

4 Zone-based Algorithm

Assume a finite set $\mathbb{X} = \{x_1, \ldots, x_k\}$ of clocks with size k, and let $\mathbb{T} = [0, d]$ be a time domain with duration d. With any TPTL formula φ and trace w we associate a *satisfaction set*, consisting of all pairs (t, r) under which w satisfies φ. For convenience we hereafter identify such time-environments pairs (t, r) with vectors in $\mathbb{T}^{k+1}$ whose first component is the value of the reference time, followed by the values of the clocks in $\mathbb{X}$.

Definition 3 (Satisfaction Set). *Let φ be a formula and w a trace. The satisfaction set of φ relative to w, denoted $[\![\varphi]\!]_w$, is defined by letting*

$$[\![\varphi]\!]_w = \{(t,r) \in \mathbb{T}^{k+1} \ : \ (w,t,r) \models \varphi\}.$$

Difference constraints are formulas of the form $t \bowtie a$ and $t - s \bowtie a$ for comparison operator $\bowtie \in \{<, \leq, >, \geq\}$, constant a, and real variables s, t. Satisfaction

sets $[\![\varphi]\!]_w$ are definable in the first order theory of difference constraints. This theory is decidable, in particular it admits quantifier elimination [24].

Since the translation of TPTL into difference constraints can easily be made effective, a monitoring procedure for TPTL can be obtained by constructing a difference constraints formula that holds iff $w \models \varphi$, combined with a decision procedure for the first order theory of difference constraints [28]. Such an algorithm is likely to exhibit an exponential time complexity, since the problem of deciding a difference constraints formula is complete for polynomial space computations [24]. We have no hope on improving the worst-case complexity relative to the combined input size of formula and trace, given that TPTL monitoring requires polynomial space [28], already over discrete models [16]. However, we hope to reduce the complexity relative to the size of the trace alone. For this we use a polyhedral representation of the satisfaction set.

Definition 4 (Zone). *A* zone *is a subset of* $\mathbb{T}^{d+1}$ *definable as a conjunction of difference constraints.*

Zones were introduced in the context of real-time systems verification, in particular in the formal analysis of timed automata [14]. The following theorem, an immediate consequence of the discussion above, underpins our first algorithm:

Theorem 1. *For any trace* w *and formula* φ*, the set* $[\![\varphi]\!]_w$ *can be effectively represented as a finite union of zones.*

Given a formula φ and trace w the set $[\![\varphi]\!]_w$ can in particular be obtained by induction as follows.

- Propositional variables: The satisfaction set is a union of zones orthogonal to the time axis, of the form $[\![p]\!]_w = \bigcup_{i=1}^{n} J_i \times \mathbb{T}^k$.
- Timing constraints: The satisfaction set consists is the zone $[\![x \bowtie c]\!]_w = \{(t, r) \in \mathbb{T}^{k+1} \ : \ t - r_x \bowtie c\}$.
- Boolean operators: Disjunction and negation translate into the corresponding set operations $[\![\neg\varphi]\!]_w = \mathbb{T}^{k+1} \setminus [\![\varphi]\!]_w$ and $[\![\varphi \vee \psi]\!]_w = [\![\varphi]\!]_w \cup [\![\psi]\!]_w$.
- Until: Assume $[\![\varphi]\!]_w$ and $[\![\psi]\!]_w$ are given as sets of zones $\mathcal{Z}_\varphi$ and $\mathcal{Z}_\psi$, respectively. We compute zones of $[\![\varphi \,\mathcal{U}\, \psi]\!]_w$ by constructing the sequence $\mathcal{Y}_0, \ldots, \mathcal{Y}_n$ up to a fixed point n as follows:

$$\mathcal{Y}_0 = \{cl_L(Z) \cap \overleftarrow{cl_R(Z) \cap Y} \ : \ Y \in \mathcal{Z}_\psi, Z \in \mathcal{Z}_\varphi\}$$
$$\mathcal{Y}_i = \{cl_L(Z) \cap \overleftarrow{Z \cap Y} \ : \ Y \in \mathcal{Y}_{i-1}, Z \in \mathcal{Z}_\varphi\} \text{ for } i > 0$$

  where cl_L (respectively cl_R) take the topological closure of a zone to the left (respectively to the right) on the time component, and $\overleftarrow{Z}$ removes all lower bounds on the time component.[5] We then have $[\![\varphi \,\mathcal{U}\, \psi]\!]_w = \bigcup_{i=0}^{n} \mathcal{Y}_i$.
- Reset: We let $[\![x.\varphi]\!]_w = \{(t, r) \in \mathbb{T}^{k+1} \ : \ \exists s, (t, r[r_x \leftarrow s]) \in Z\}$ for $Z = [\![\varphi]\!]_w \cap \{(t, r) \in \mathbb{T}^{k+1} \ : \ t - r_x = 0\})$. All operations involved in this computation commute with $\cup$ and are standard operations over zones.

[5] The fixed point $\cup\mathcal{Y}_{n+1} \subseteq \bigcup_{i=0}^{n} \mathcal{Y}_i$ exists because only finitely many difference constraints over $\mathbb{T}$ can be built from $\mathcal{Z}_\varphi$ and $\mathcal{Z}_\psi$.

Example 1. We consider the formula $\varphi \equiv x.\Diamond(p \wedge \Diamond(q \wedge x \leq 1))$. It has subformulas p, q, $x \leq 1$, $\gamma_1 \equiv q \wedge x \leq 1$, $\gamma_2 \equiv \Diamond\, \gamma_1$, $\gamma_3 \equiv p \wedge \gamma_2$, $\gamma_4 \equiv \Diamond\, \gamma_3$, with $\varphi \equiv x.\gamma_4$. In Figure 3 we show the satisfaction sets of each of its subformulas. Observe that the satisfaction of φ is independent of r.

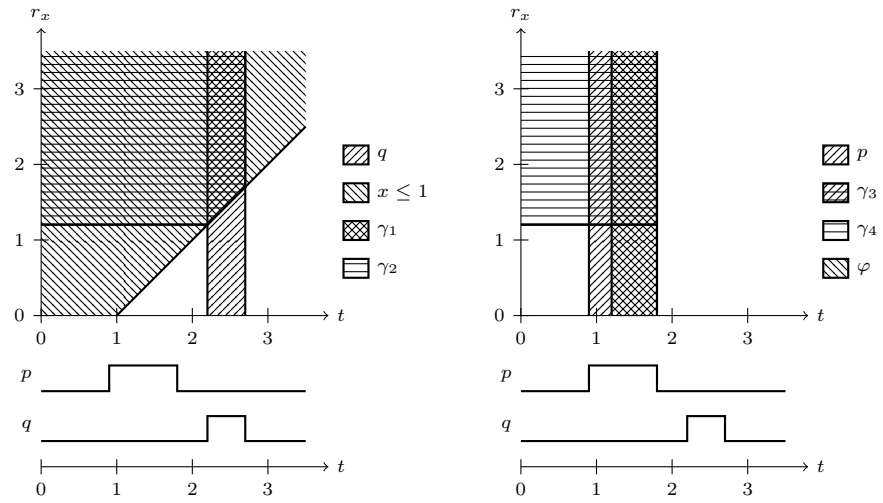

Fig. 3. Computation of the satisfaction set of formula φ on a given trace by structural induction: (**left**) satisfaction sets of subformulas q, $x \leq 1$, γ_1, and γ_2; (**right**) satisfaction sets of subformulas p, γ_3, γ_4, and φ.

For a fixed formula, the worst-case run time of this algorithm is polynomial relative to the trace length. Yet it can be more than linear. The expensive operation of complementation can be avoided by introducing a negation normal form through additional operators of conjunction and *always* (the dual of *until* can be rewritten using *always* and *until* itself). However intersecting two sets of zones can still create a quadratic number of zones. Such a phenomenon can arise when monitoring TPTL with the algorithm in this section.

Example 2. Consider the formula $\psi \equiv p \wedge \Diamond(p \wedge x = 1)$, and the family of periodic Boolean signals w_n, $n > 0$ with fixed duration $d = 2$ and period $\frac{1}{n}$, such that $w_n[t] = 1$ if and only if $\lfloor \frac{t}{2n} \rfloor$ is even. The satisfaction set $[\![\psi]\!]_w$ has $\Omega(n^2)$ zones while signal w_n has $O(n)$ time points (discontinuities).

5 Region-based Algorithm

In this section we improve on our zone-based algorithm by moving to a representation of satisfaction sets using a notion of *region equivalence*. For simplicity we focus on the fragment of TPTL with only one clock variable x, which we denote 1-TPTL in the rest of this paper.

5.1 TPTL Formulas with One Variable

Under the present definitions, 1-TPTL is already more expressive than MTL [9]. Given a time variable x and an integer-bounded interval I, let us write $x \in I$ for the conjunction of constraints enforcing that the value of x lies in I. We can define the *timed until* operator $\mathcal{U}_I$ as the abbreviation $\varphi \mathcal{U}_I \psi \equiv x.(\varphi \mathcal{U}(x \in I \wedge \psi))$. Metric Temporal Logic (MTL) can be seen as the syntactic fragment of TPTL with the grammar $\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \mathcal{U}_I \varphi$ for p ranging over AP and I ranging over integer-bounded intervals.

The 1-TPTL formula $\varphi_1 \equiv \Box(p \rightarrow x.\Diamond(q \wedge \Diamond(r \wedge x \leq 5)))$ was conjectured in [2] not to be expressible in MTL. The property specified by φ_1 is that every request, signified by p holding true, should be followed by two successive grants occurring within 5 time units, respectively signified by q and r holding true. It turns out that φ_1 can be expressed in MTL [9], but not when replacing the constraint $x \leq 5$ by $x \leq 1$ and assuming integer constants [21]. When allowing rational constants and past operators, MTL, 1-TPTL, and TPTL all become equivalent in expressive power [22].

Observe that formulas of 1-TPTL can contain multiple occurrences of clock variable x, as in $\varphi_2 \equiv \Diamond x.(p\mathcal{U}(x > 1 \wedge x.(q\mathcal{U}(r\mathcal{U}\, x \geq 2 \wedge x \leq 3))))$. Formula φ_2 expresses that eventually p holds for more than 1 time unit, after which q holds and then r holds over a period lasting between 2 and 3 time units. It could also be written as $\Diamond\, x.(p\mathcal{U}(x > 1 \wedge y.(q\mathcal{U}(r\mathcal{U}\, y \geq 2 \wedge y \leq 3))))$ for readability's sake.[6]

5.2 Region Equivalence

To improve the worst-case complexity relative to the trace length, we introduce two essential changes in the algorithm of Section 4. We avoid overlapping polytopes, leading to combinatorial explosion, by using a grid over the 2-dimensional time domain. The number of cells (called *regions*) in the grid can still be more than linear in the trace length, as in Example 2. Instead of representing the whole set of zones explicitly, it suffices to construct this set implicitly and according to some equivalence relation. The state is maintained over a single uniform interval on the t-axis, where the input trace stays constant. Over such an interval, the truth value of a formula only depends on the environment and for convenience we will represent it as a signal on the r-axis.

Let w be a trace with time sequence $0 = t_0, \ldots, t_n = d$ and φ a formula with time constants $c_1, \ldots, c_l$ and letting $c_0 = 0$ and $c_{l+1} = +\infty$. We write $r_0, \ldots, r_m$ for the ordered sequence of times in $\mathbb{T}$ each of the form $t_i - c_j$ obtained by considering all pairs of t_i and c_j for $i = 0, \ldots, n$ and $j = 0, \ldots, l$.

Definition 5 (Region). *A* region *relative to w and φ is a subset of $\mathbb{T}^2$ of the form $\{(t, r) \in T \times R \ : \ t - r \in I\}$ where T is of the form $\{t_i\}$ or (t_i, t_{i+1}), R is of the form $\{r_j\}$ or (r_j, r_{j+1}), and I is of the form $\{c_k\}$ or (c_k, c_{k+1}). We call T*

[6] Similar formulas with independent variables were considered in [15] in the context of monitoring. We remark that the fragment of TPTL defined there corresponds to 1-TPTL when clocks are renamed.

the projection *of that region on the t-axis, and if* $T \neq \{d\}$ *we call* successor *the region* $\{(t, r) \in T' \times R \ : \ t - r \in I\}$ *where* T' *is adjacent to* T *on the right.*

Definition 6 (Equivalence). *We say that two regions* A *and* A' *are equivalent relative to* w *and* φ*, denoted* $A \sim_{\varphi,w} A'$*, when the following conditions apply:*

- A *and* A' *have the same t-axis projection;*
- *the satisfaction status of subformulas of* φ *relative* w *are the same on both* A *and* A'*;*
- *if* A *and* A' *have successors* B *and* B' *then the satisfaction status of subformulas of* φ *relative to* w *are the same on both* B *and* B'*.*

Let φ be a *quantifier-free*[7] formula, and let w be a trace. The following proposition is straightforward by structural induction:

Proposition 1. *For all regions* A *and* A' *such that* $A \sim_{\varphi,w} A'$ *and time-environment pairs* $(t, r) \in A$ *and* $(t', r') \in A'$ *we have* $(t, r) \models \varphi$ *iff* $(t', r') \models \varphi$*.*

In order to compute the satisfaction set of a quantifier-free formula φ, time-environment pairs that lie in regions equivalent to $\sim_{\varphi,w}$ can be grouped together. Parsing the trace in reverse time-order, the number of operations per uniform time interval needed to update equivalence classes of $\sim_{\varphi,w}$ remain bounded.

For quantified subformulas we use the following notion:

Definition 7 (Satisfaction Signal). *The satisfaction signal* w_φ *of a closed formula* φ *on a trace* w *is a Boolean signal such that* $w_\varphi[t] = 1$ *if* $(w, t) \models \varphi$*,* $w_\varphi[t] = 0$ *otherwise.*

The satisfaction signal of some formula $x.\varphi$ can be obtained by intersecting the satisfaction set of φ with the diagonal $t = r$. Observe that in general, the satisfaction signal of a closed subformula is sufficient information to construct the satisfaction set of its superformulas. Applying the region equivalence to formulas with quantifiers will be made possible by incrementally replacing quantified subformulas with their satisfaction signal.

5.3 Monitoring Algorithm

For a given formula φ and a trace w, the region-based algorithm computes the satisfaction signal of every subformula of the form $x.\gamma$, starting with inner-most ones (such that γ is quantifier-free). The computation of the satisfaction signal of such a subformula $x.\gamma$ is done by parsing the trace backwards and computing the satisfiability of its subformulas in each region, in a procedure similar to LTL monitoring. The satisfaction signal $w_{x.\gamma}$ is found on the diagonal and obtained

[7] A more general definition of *region equivalence* could be used. Our restriction of this notion to quantifier-free formulas is motivated by efficiency concerns. For instance, we aim to avoid partitioning the satisfaction set of formula $x.\,\Diamond(x \leq 1 \wedge p \wedge x.\,\Diamond(x \leq 2 \wedge q))$ according to timing constant $1 + 2$ for all subformulas. While the constant is relevant in subformula $\Diamond(x \leq 1 \wedge p \wedge x.\,\Diamond(x \leq 2 \wedge q))$, it plays no role in $\Diamond(x \leq 2 \wedge q)$.

by letting $w_\varphi[t] = 1$ if $(t, t) \in [\![\varphi]\!]_w$. Indeed we only need to compute the part of the satisfaction set with $r \leq t$. Once computed, the subformula $x.\gamma$ is replaced by a fresh proposition $p_{x.\gamma}$ and its satisfaction signals is added to the trace w. The satisfaction signal associated to that proposition will be used when computing superformulas, similar to MTL monitoring. Once the main formula φ has been replaced by an atomic proposition p_φ, we can conclude whether w satisfies φ by simply looking at the value of w_{p_φ} at time 0. We assume without loss of generality that the closed formula φ we monitor is of the form $x.\psi$, if this was not the case we could rewrite it as $x.\varphi$, which is equivalent since φ is closed.

Algorithm 1 Monitor

Precondition: A formula $\varphi \equiv x.\psi$, a finite trace w

```
1: function MONITOR(φ, w)
2:     if ψ contains x.γ such that γ is quantifier-free then
3:         v ← SATISFY(x.γ, w)
4:         replace x.γ by p_{x.γ}  in φ
5:         w ← w ∪ (p_{x.γ} ↦ v)
6:         return MONITOR(φ, w)
7:     else
8:         return SATISFY(φ, w)
9:     end if
10: end function
```

As described, Algorithm 1 recursively searches for a subformula that does not contain any reset quantifier (lines 2, 6) until no further reset quantifiers can be found (line 8). In that case, the algorithm proceeds by computing the satisfaction signal of the found subformula by calling Algorithm 2 (line 3) and replacing it with a fresh atomic proposition $p_{x.\gamma}$ (line 4) and in addition, supplementing the trace with a Boolean satisfaction signal v for this proposition $p_{x.\gamma}$ (line 5). For a formula $x.\varphi$ where φ is quantifier-free and a trace $\mathbb{T} \to 2^{AP}$ we compute $w_{x.\varphi}$ by calling Algorithm 2.

Algorithm 2 implicitly computes the satisfaction set of all quantifier-free subformulas of φ. For simplicity, it is written and described to operate over regions rather than region equivalence classes, but operating over a single representative of each region equivalence class can easily be implemented (e.g. by keeping track of regions entering and leaving every diagonal area of the t, r plane). The algorithm starts by initializing the output trace u. Signals in the output trace represent the satisfiability of subformulas at different environment values. The function INITIALIZESATTRACE creates m signals in the output trace u, one per subformula of γ, with time points 0, t_n, and $t_n - c$ for every $c \in C$. The values of those satisfaction signals are computed based on the signal values of w at time t_n. As we iterate over the trace (line 3), we first refine u (lines 4, 5) and we update the output trace backwards (line 7). We proceed by computing the regions relative to w contained within $T = [t_i, t_{i+1})$ and $R = (r_j, r_{j+1})$ (line 8). The function UPDATEREGIONS then only needs to compute the time constants relevant to the

Algorithm 2 Satisfy

Precondition: A formula $x.\gamma$ such that γ is quantifier-free, a trace w on $[0, t_n]$

1: **function** SATISFY($x.\gamma, w$)
2: $u, r \leftarrow$ INITIALIZESATTRACE(w, t_n, C) ▷ r stores all time points of u
3: **for** $t_i \in t_{n-1}, \ldots, t_0$ **do**
4: $r' \leftarrow (r'_1, \ldots, r'_{|C|})$ where $r'_l := t_i - c_l$ for all $1 \leq l \leq |C|$
5: $r \leftarrow \text{merge}(r, r')$ ▷ merge two lists
6: $k \leftarrow$ largest element of r smaller than t_{i+1}
7: **for** $r_j \in r_k, \ldots, r_0$ **do**
8: $A, A' \leftarrow$ UPDATEREGIONS($t_i, t_{i+1}, r_j, r_{j+1}, C$) ▷ open interval
9: **for** $B \in A', \ldots, A$ **do**
10: $u \leftarrow$ UPDATESATTRACE(w, u, r_j, r_{j+1}, B)
11: **end for**
12: $A, A' \leftarrow$ UPDATEREGIONS(t_i, t_{i+1}, r_j, C) ▷ closed interval
13: **for** $B \in A, \ldots, A'$ **do**
14: $u \leftarrow$ UPDATESATTRACE(w, u, r_j, B)
15: **end for**
16: **end for**
17: **end for**
18: **return** $u_{x.\gamma}$ ▷ satisfaction signal of $x.\gamma$
19: **end function**

intervals T and R, i.e. time constants c such that $t_i \leq c + r_j \leq t_{i+1}$. Iterating backwards through the computed regions (for $B = A', \ldots, A$), we compute the satisfiability in each region inductively on the structure of the formula (function UPDATESATTRACE, line 10) and update u once we have processed the region A. The function UPDATESATTRACE updates the respective signals of u at time (r_j, r_{j+1}) based on subformulas γ_l of $x.\gamma$. For instance in the case of $\gamma_l \equiv \gamma_h \,\mathcal{U}\, \gamma_k$ over a region whose t-projection is open, we update the signal u_{γ_l} with the value of $u_{\gamma_h} \wedge (u_{\gamma_k} \vee u'_{\gamma_k} \vee u'_{\gamma_l})$, where u' is the value of u in the adjacent region to the right. Over a region whose t-projection is closed, the value of *until* is the same as in its successor region. Other operators not pose any difficulty. After we have processed the region A, we update u. We repeat the same inductive rules for the regions bounded by $T = [t_i, t_{i+1})$ and $R = \{r_j\}$. After each iteration t_i, the interval $[t_i, t_{i+1})$ of the output trace u is finalized and will remain unchanged until the end of computation, at which point we return the Boolean component of u representing the satisfaction signal of the whole formula $x.\gamma$.

Example 3. We take again the formula $\varphi \equiv x.\, \Diamond(p \wedge \Diamond(q \wedge x \leq 1))$ with subformulas $p, q, x \leq 1, \gamma_1 \equiv q \wedge x \leq 1, \gamma_2 \equiv \Diamond\, \gamma_1$, $\gamma_3 \equiv p \wedge \gamma_2$, and $\gamma_4 \equiv \Diamond\, \gamma_3$ of Example 1 and illustrate the computation of its satisfaction signal in Figure 4.

When instead computing only one representative of each equivalence class of $\sim_{\gamma,w}$, we limit the number of operations to $O(2^k)$ per uniform time interval for a subformula of size k, since there are at most 2^k equivalence classes of $\sim_{\gamma,w}$ over such an interval. The number of time points at most doubles with each time

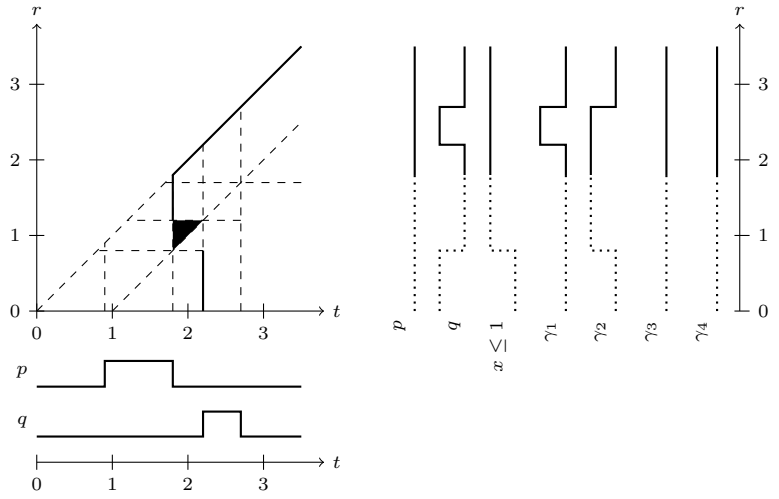

Fig. 4. Computation of the satisfaction signal of φ on a given trace. The state of the algorithm consists of the truth value of regions highlighted on the left; it is shown on the right as signals with a dotted part being updated and a plain part, final. The current time interval is $t_i = 1.8, t_{i+1} = 2.2$ and current region is $\{(r,t) \,:\, t_i < t, r < t_{i+1} - 1, t - r < 1\}$.

constant, so that the satisfaction signal of $w_{x.\gamma}$ has length at most $n2^k$ for a signal w of length n. Therefore we have:

Theorem 2. *The offline monitoring of a 1-TPTL formula φ of size m against a continuous-time Boolean trace w of length n can be computed in time $n2^{O(m)}$.*

6 Experimental Evaluation

We implemented both algorithms in C++. The implementation of the zone-based algorithm uses a library of the toolset IF [10] for zones computations. We then measured the execution time of monitoring several formula/trace combinations. Figures were obtained on Intel Core i5-4210u CPU with 8 GB of RAM. The input traces we considered consist of periodic Boolean signals, in which propositions $p, q, r, \ldots$ hold for 2 time units in turn. The length of a trace is determined by the number of sample points (associated to a Boolean signal changing its value). We generated traces of length 1000, 2000, 5000, 10000 and 20000 samples.

In a first experiment, we evaluate our region-based implementation on formulas

$$\varphi_1 \equiv \Box\, x.(p \rightarrow \Diamond(q \wedge \Diamond(x \leq 5 \wedge r)))$$
$$\varphi_2 \equiv \Diamond\, x.(p \mathcal{U} (x > 1 \wedge x.(q \mathcal{U} (r \mathcal{U}\, x \geq 2 \wedge x \leq 3))))$$
$$\varphi_3 \equiv \Box\, x.(p \rightarrow \Diamond(x \leq 1 \wedge q \wedge x.\Box(x \leq 1 \rightarrow \neg r)))$$
$$\varphi_4 \equiv \Box\, x.(p \rightarrow (\Diamond(q \mathcal{U}\, r) \wedge \Diamond(x \geq 3 \wedge x \leq 5 \wedge s)))$$
$$\varphi_5 \equiv (x.\Diamond(x \leq 10 \wedge p)) \mathcal{U} \Box \neg q.$$

Formula φ_1 and φ_2 are two examples given in Section 5. Formula φ_3 specifies that whenever p holds, q should hold within 1 time unit and r should not hold for another 1 time unit from there on. Formula φ_4 requires that every occurrence of p is followed by q holding until an occurrence of r, and an occurrence of s within 3 to 5 time units. Formula φ_5 roughly says that p holds at least once every 10 time units until q stops holding.

Then, we evaluate our zone-based implementation against the same formulas and formula $\varphi_6 \equiv \Box x.(p \rightarrow \Diamond(q \wedge y.\Diamond(x \leq 5 \wedge y \geq 2 \wedge r)))$. The property expressed by φ_6 is that every request p is followed by two grants q and r within 5 time units, with q occurring at least 2 time units before r. Such a property cannot be monitored by the region-based implementation since it requires two clock variables.

We use the tool AMT [29] for MTL monitoring over continuous-time Boolean signals as our baseline. Formulas φ_3 and φ_5 are part of the MTL syntactic fragment of TPTL, and can be rewritten in MTL as $\Box(p \rightarrow \Diamond_{[0,1]}(q \wedge \Box_{[0,1]} \neg r))$ and $(\Diamond_{[0,10]} p)\,\mathcal{U}\,\Box \neg q$, respectively.[8]

Table 1. Execution times (s) of monitoring formulas against periodic traces for three algorithms: our region-based (`reg`) and zone-based (`zon`) implementations, and the interval-based (`int`) implementation of MTL monitoring in the tool AMT.

$\lvert w\rvert$	1 000			2 000			5 000			10 000			20 000		
alg	`int`	`reg`	`zon`	`int`	`reg`	`zon`	`int`	`reg`	`zon`	`int`	`reg`	`zon`	`int`	`reg`	`zon`
φ_1	–	0.045	0.085	–	0.084	0.168	–	0.217	0.431	–	0.439	0.960	–	0.898	2.105
φ_2	–	0.110	0.059	–	0.160	0.132	–	0.407	0.370	–	0.814	0.739	–	1.660	1.498
φ_3	0.034	0.104	0.032	0.047	0.169	0.077	0.079	0.431	0.173	0.143	0.894	0.344	0.275	1.822	0.644
φ_4	–	0.132	0.087	–	0.259	0.268	–	0.662	0.632	–	1.348	1.416	–	2.756	3.015
φ_5	0.025	0.080	0.040	0.035	0.159	0.151	0.055	0.398	0.366	0.092	0.802	0.783	0.173	1.636	2.235
φ_6	–	–	0.242	–	–	0.390	–	–	1.001	–	–	2.111	–	–	5.009

The results are shown in Table 1. We observe that the zone-based algorithm matches closely the linear-time guaranteed performance of the region-based algorithm, and is sometimes faster. This is achieved by internally keeping zones ordered on the time axis to avoid otherwise quadratic implementation of binary operations such as intersection, see [33]. For large signal sizes the performance degrades, subject to an implementation limitation of IF (the use of a hash table for zones). The interval-based monitoring algorithm of AMT displays a speed advantage of up to 10× when monitoring formulas φ_3 and φ_5.

In a second experiment, we consider the scalability of our region-based algorithm relative to trace and formula dimensions. To demonstrate the impact of the number of operators in the formulas, we consider the family $\sigma_1 \equiv \Diamond x.(p_1 \wedge x \leq 2)$, $\sigma_2 \equiv \Diamond x.(p_1\,\mathcal{U}(p_2 \wedge x \leq 4))$, up to $\sigma_5 \equiv \Diamond x.(p_1\,\mathcal{U}(p_2\,\mathcal{U}\ldots\mathcal{U}(p_5 \wedge x \leq 10)\ldots))$. To demonstrate the impact of the number of constants in the formula, we consider

[8] Formula φ_4 could also be put in MTL form using some additional rewriting, but is not part of the MTL syntactic fragment of TPTL we defined.

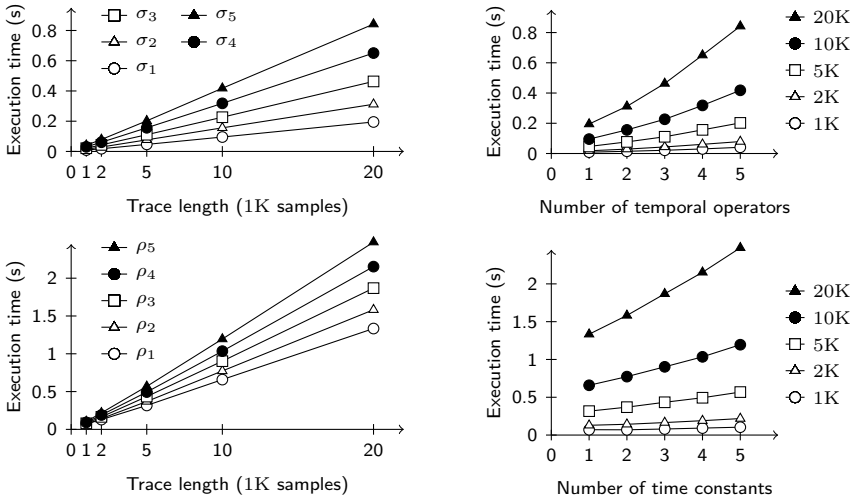

Fig. 5. Execution time for: (**left**) traces of increasing length; (**top-right**) formulas of increasing size; (**bottom-right**) formulas with increasing number of time constants.

the family $\rho_i \equiv \Box\, x.(p_0 \rightarrow \Diamond(p_1 \wedge x \leq c_1^i \wedge \Diamond(p_2 \wedge x \leq c_2^i \wedge \ldots \wedge \Diamond(p_5 \wedge x \leq c_5^i)\ldots)))$ for $i = 1, \ldots, 5$ with constants $c_1^1 = c_2^1 = \ldots = c_5^1 = 10$; $c_1^2 = 8, c_2^2 = \ldots = c_5^2 = 10$; up to $c_1^5 = 2, c_2^5 = 4, \ldots, c_5^5 = 10$. Formulas σ_i contain an increasing number of *until* operators, while formulas ρ_i contain an increasing number of time constants.

The results are shown in Figure 5. In the left-hand side we confirm that the execution time is linear relative to the length of the trace for a fixed formula. In the right-hand side we see that as the size of the formula, or its number of constants increases, the execution time appears to grow only slightly faster than linearly. This is expected over traces with bounded variability. More realistic benchmarks would be needed in order to fully assess the practical behavior of our algorithm relative to formula dimensions. Its asymptotic behavior in that respect is only of relative interest, given that beyond a handful of temporal operators or time constants, formulas quickly become less intelligible.

7 Conclusion

We demonstrated how the offline monitoring of temporal logic with real-valued clock variables can be made to scale with the trace length. In the future, we would like to investigate the monitoring problem for logics with other forms of quantification such as first-order [7,18], or *freeze* quantification over signal values [11]. Efficient monitoring of such logics would be of practical interest.

Acknowledgements This research was supported in part by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award).

References

1. Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.
2. Rajeev Alur and Thomas A Henzinger. Logics and models of real time: A survey. In *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*, pages 74–106. Springer, 1991.
3. Rajeev Alur and Thomas A Henzinger. A really temporal logic. *Journal of the ACM (JACM)*, 41(1):181–203, 1994.
4. Roy Armoni, Dana Fisman, and Naiyong Jin. SVA and PSL local variables–a practical approach. In *International Conference on Computer Aided Verification*, pages 197–212. Springer, 2013.
5. Eugene Asarin, Paul Caspi, and Oded Maler. Timed regular expressions. *Journal of the ACM*, 49(2):172–206, 2002.
6. Eugene Asarin, Dejan Nickovic, Oded Maler, and Dogan Ulus. Combining the temporal and epistemic dimensions for MTL monitoring. In *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2017.
7. David Basin, Felix Klaedtke, Samuel Müller, and Eugen Zălinescu. Monitoring metric first-order temporal properties. *Journal of the ACM (JACM)*, 62(2):15, 2015.
8. David Basin, Srđan Krstić, and Dmitriy Traytel. Almost event-rate independent monitoring of metric dynamic logic. In *International Conference on Runtime Verification*, pages 85–102. Springer, 2017.
9. Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 432–443. Springer, 2005.
10. Marius Bozga, Jean-Claude Fernandez, Lucian Ghirvu, Susanne Graf, Jean-Pierre Krimm, and Laurent Mounier. IF: A validation environment for timed asynchronous systems. In *International Conference on Computer Aided Verification*, pages 543–547. Springer, 2000.
11. Lubos Brim, P Dluhoš, D Šafránek, and Tomas Vejpustek. STL*: Extending signal temporal logic with signal-value freezing operator. *Information and Computation*, 236:52–67, 2014.
12. Ming Chai and Holger Schlingloff. A rewriting based monitoring algorithm for TPTL. In *International Workshop on Concurrency, Specification and Programming (CS&P)*, pages 61–72, 2013.
13. Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martı-Oliet, José Meseguer, and José F Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 285(2):187–243, 2002.
14. D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems*, pages 197–212, New York, NY, USA, 1990. Springer-Verlag New York, Inc.
15. Adel Dokhanchi, Bardh Hoxha, Cumhur Erkan Tuncali, and Georgios Fainekos. An efficient algorithm for monitoring practical TPTL specifications. In *International Conference on Formal Methods and Models for System Design (MEMOCODE)*, pages 184–193. IEEE, 2016.
16. Shiguang Feng, Markus Lohrey, and Karin Quaas. Path checking for MTL and TPTL over data words. In *International Conference on Developments in Language Theory*, pages 326–339. Springer, 2015.

17. Harry Foster. Assertion-based verification: Industry myths to realities (invited tutorial). In *Computer Aided Verification*, pages 5–10. Springer, 2008.
18. Klaus Havelund, Doron Peled, and Dogan Ulus. First order temporal logic monitoring with BDDs. *Formal Methods in Computer-Aided Design FMCAD 2017*, page 116, 2017.
19. Klaus Havelund and Grigore Roşu. Monitoring java programs with java pathexplorer. *Electronic Notes in Theoretical Computer Science*, 55(2):200–217, 2001.
20. Klaus Havelund and Grigore Roşu. Synthesizing monitors for safety properties. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 342–356. Springer, 2002.
21. Yoram Hirshfeld and Alexander Rabinovich. Expressiveness of metric modalities for continuous time. In *Computer Science–Theory and Applications*, pages 211–220. Springer, 2006.
22. Paul Hunter, Joël Ouaknine, and James Worrell. Expressive completeness for metric temporal logic. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 349–357. IEEE Computer Society, 2013.
23. Moonzoo Kim, Mahesh Viswanathan, Sampath Kannan, Insup Lee, and Oleg Sokolsky. Java-mac: A run-time assurance approach for java programs. *Formal methods in system design*, 24(2):129–155, 2004.
24. Manolis Koubarakis. Complexity results for first-order theories of temporal constraints. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 379–390, 1994.
25. Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299, 1990.
26. Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Joint International Conferences on Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems (FORMATS/FTRTFT)*, pages 152–166, 2004.
27. Oded Maler and Dejan Nickovic. Monitoring properties of analog and mixed-signal circuits. *STTT*, 15(3):247–268, 2013.
28. Nicolas Markey and Jean-François Raskin. Model checking restricted sets of timed paths. *Theoretical Computer Science*, 358(2-3):273–292, 2006.
29. Dejan Nickovic, Olivier Lebeltel, Oded Maler, Thomas Ferrère, and Dogan Ulus. AMT 2.0: Qualitative and quantitative trace analysis with extended signal temporal logic. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 303–319. Springer, 2018.
30. Amir Pnueli. The temporal logic of programs. In *Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.
31. Jean-François Raskin. *Logics, Automata and Classical Theories for Deciding Real Time*. PhD thesis, Université de Namur, 1999.
32. Volker Stolz and Eric Bodden. Temporal assertions using AspectJ. *Electronic Notes in Theoretical Computer Science*, 144(4):109–124, 2006.
33. Dogan Ulus, Thomas Ferrère, Eugene Asarin, and Oded Maler. Timed pattern matching. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 222–236. Springer, 2014.
34. Srikanth Vijayaraghavan and Meyyappan Ramanathan. *A practical guide for SystemVerilog assertions*. Springer Science & Business Media, 2005.