# Model Checking Restricted Sets of Timed Paths

Nicolas Markey and Jean-François Raskin

Département d'Informatique
Université Libre de Bruxelles
Bld du Triomphe – CP 212
1050 Brussels – Belgium
email: {nmarkey,jraskin}@ulb.ac.be

**Abstract.** In this paper, we study the complexity of model-checking formulas of three important real-time logics (MTL, MITL, and TCTL) over restricted sets of timed paths. The classes of restricted sets of timed paths that we consider are (*i*) a single finite (or ultimately periodic) timed path, (*ii*) a infinite set of finite (or infinite) timed paths defined by a finite (or ultimately periodic) path in a region graph, (*iii*) a infinite set of finite (or infinite) timed paths defined by a finite (or ultimately periodic) path in a zone graph.

## Introduction

*Timed automata* have been introduced in [2] as a formal notation to model behaviors of real-time systems. Requirements for real-time systems modeled as timed automata are conveniently expressed using *real-time logics*. Real-time logics are *quantitative extensions* of temporal logics. Three main logics have been defined to express real-time requirements: TCTL [1] is a real-time extension of the CTL logic, while MTL and MITL [3] are extensions of the LTL logic. The model checking problems for those logics over *sets of timed paths* defined by timed automata have been studied. The results are as follows: For the logic TCTL, the model checking problem has been shown PSPACE-complete in [1]. For the logic MTL, the problem has been shown undecidable in [4]. For the logic MITL, the problem has been shown EXPSPACE-complete in [3].

In this paper, we study the model checking problems for those real-time logics on *several classes* of *restricted* sets of timed paths. We consider the model checking problems related to TCTL, MTL, and MITL when the set of timed paths is (*i*) a single finite (or ultimately periodic) timed path, (*ii*) a set of finite (or infinite) timed paths defined by a finite (or ultimately periodic) path in a region graph, (*iii*) a set of finite (or infinite) timed paths defined by a finite (or ultimately periodic) path in a zone graph. Note that in cases (*ii*) and (*iii*), the sets contain *uncountably many* timed paths. Note also that finite or ultimately periodic region paths as well as zone paths can be seen as *simple* form of timed automata.

Beside the theoretical interest to study the complexity of the model checking problems for those subcases, there are important practical reasons to study them.

First, verification algorithms for timed automata have to manipulate symbolically infinite state spaces. This is done either through the region graph or through the zone graph. When the verification of a *safety* or a *linear-time property* fails, those symbolic algorithms identify a finite or an infinite ultimately periodic path in the region graph or in the zone graph [6]. This path is the symbolic representation of an infinite set of timed paths that are counter-examples to the property. Usually, the information that is given back to the user is a single timed path extracted from this symbolic path. Nevertheless, it may be much more interesting to give to the user not only a single counter-example but the entire infinite set of counter-examples actually computed by the symbolic algorithm. As this counter example is symbolic, the possibility to analyze this counter-example using model checking should be given to the user. In fact, in order to better understand this infinite set of counter examples, the user may want to formulate model checking questions about this set. We should then look if we can specialize our verification algorithms for those (possibly) simpler problems.

Second, a real-time system that is executing constructs naturally a timed path that represent the evolution of its state along time. Correctness requirements about this single execution path can be expressed using a linear real-time logic, like MTL. Can we efficiently verify properties expressed by MTL formulas on this single timed path? In the dense-time framework, we know from the undecidability result for MTL that we cannot construct, as in the finite state case for LTL, a *monitor* (in the form of a timed automaton for example) that will enter a bad state in the case the formula is not satisfied. It is clear again that we need to look at specific techniques.

Third, if a timed automaton is too complex to be completely verified, we may be interested to *test* it instead. Testing a timed automaton against a set of real-time formulas consists in: (*i*) extracting a set of timed paths out of the timed automaton by executing it, and (*ii*) verifying that the set of extracted path verify some given real-time formulas. The set of timed paths can be either extracted by explicit execution of the timed automaton (in this case it is a finite set) or, more interestingly, extracted through symbolic execution of the timed automaton which amounts to visit a subset of its region or zone graph. In the two latter cases, we must check real-time formulas on infinite sets of timed paths defined by a finite set of region or zone paths. Again, for those subcases, we should look at the existence of specialized techniques.

The results that we have obtained for the model checking problems of the three real-time logics over the six classes of restricted sets of timed paths are given in Table 1. To the best of our knowledge, only the three results from the first line were known, all the other results are new. The undecidability and EXPSPACE-hard result for the model checking of MTL and MITL over ultimately periodic region paths were *unexpected* and their proofs have required new encoding techniques for Turing Machine computations using timed paths. Those hardness results imply hardness results for ultimately periodic zone paths. In those proofs, all the information about the tape of the Turing Machine (TM for short) is encoded into the timing information of the path, the sequence of

| | MTL | MITL | TCTL |
|---|---|---|---|
| Timed Automata | Undecidable [4] | EXPSPACE-c. [3] | PSPACE-c. [1] |
| Ult. Per. Zone Paths | Undecidable | EXPSPACE-c. | PSPACE-c. |
| Ult. Per. Region Paths | Undecidable | EXPSPACE-c. | PTIME |
| Finite Zone Paths | co-NP-c. | co-NP-c. | PSPACE-c. |
| Finite Region Paths | co-NP-c. | co-NP-c. | PTIME |
| Ult. Per. Timed Paths | PTIME | PTIME | PTIME |
| Finite Timed Paths | PTIME | PTIME | PTIME |

Table 1: Complexity of path model checking

propositional assignments being fixed by the ultimately periodic region path. This situation is rather different from the classical proofs in [4] and in [3]. Note also that the complexity of TCTL model checking goes from PSPACE-complete to PTIME in the case of region paths which is not the case for zone paths for which the problem stays PSPACE-complete. On the other hand, when we consider finite region or zone paths, the model checking problem for MTL becomes *decidable* and is co-NP-complete. The proofs for those results are based on $(i)$ a *polynomial time* algorithm for checking the truth value of a MTL formula over a single finite timed path and on $(ii)$ the proof that transitions between regions (respectively zones) in a region (respectively zone) path can be chosen nondeterministically in a finite subset of the rationals with *polynomially many* elements to establish if a region (respectively zone) path has at least one concretization that satisfies the MTL formula.

*Related work. Path model checking* has been introduced in [10], where efficient algorithms are proposed for several untimed temporal logics (LTL, LTL+Past, ...). The basic remark there is that a CTL algorithm can be applied in order to verify LTL specifications, since path quantifier would refer to the only possible run of the structure. This does not hold here when we deal with region or zone paths.

Runtime verification and monitoring is another related issue. In that case, properties are verified on the fly during the run, as the events occur. Recently, monitoring algorithms have been proposed in the discrete-time framework for MTL [11].

In our work, we verify properties expressed in three important timed temporal logics. The case where the property is expressed as a timed automaton is treated in [5], where the authors show that deciding if a timed trace corresponds to an execution of a given timed automaton is PSPACE-complete. This result can easily be extended to region paths.

*Structure of the paper.* The rest of the paper is structured as follows. In a first section, we define the classes of restricted sets of timed paths for which we study the complexity of the model checking problems. We also recall in this section the syntax and semantics of MTL, MITL, and TCTL. In a second section, we present complexity results that can be interpreted as negative: we show that for some classes of restricted sets of timed models, some model checking problems are not easier than in the general case (when the set of timed paths is defined by a timed automaton). In a third section, we present complexity results that can be interpreted as positive: we show that for some interesting classes of restricted set of timed paths, some model checking problems are easier than in the general case.

# 1 Preliminaries

## 1.1 Timed automata and paths

We write $\mathbb{R}^+$ for the set of positive real numbers. In the sequel, all intervals we consider are convex subsets of $\mathbb{R}^+$ with rational greatest lower and least upper bounds. An interval is said to be *singular* if it contains only one value. Given two intervals $I$ and $J$, and a positive rational number $t$, we write $I-t$ for the set $\{x \in \mathbb{R}^+ \mid x+t \in I\}$ and $I-J$ for the set $\{x \in \mathbb{R}^+ \mid \exists y \in J.\ x+y \in I\}$. Given an interval $I$, we note $l(I)$ the greatest lower bound of $I$ and $r(I)$ least upper bound of $I$. An interval $J$ follows an interval $I$ if $I \cup J$ is convex, $I \cap J = \emptyset$ and $r(I) = l(J)$. A finite (resp., infinite) sequence of intervals $(I_i)$, with $0 \leq i \leq n$ (resp. $0 \leq i$) partitions a set $D \subseteq \mathbb{R}^+$ if for any $0 < i \leq n$ (resp. $0 < i$), interval $I_i$ follows $I_{i-1}$ and $\cup_{i=0}^{i=n} I_i = D$ (resp. $\cup_{i=0}^{i=+\infty} I_i = D$).

Let $H$ be a set of variables. We define the set $\mathbb{C}(H)$ of *clock difference constraints* inductively as follows:

$$\mathbb{C}(H) \ni \delta, \delta' \ ::= \ x \sim c \mid x - y \sim c \mid \delta \wedge \delta'$$

for any two variables $x$ and $y$ in $H$, for $\sim$ in $\{<, \leq, =, \geq, >\}$, and for any integer $c$.

Given a valuation $v$ for the variables in $H$, the boolean value of a difference constraint $\delta$ is defined in the obvious way. Moreover, for any real $t$, we define the valuation $v+t$ as being the valuation $x \mapsto v(x)+t$, and for any subset $C$ of $H$, the valuation $v_C$ as being the valuation $x \in C \mapsto 0$, $x \notin C \mapsto v(x)$.

**Definition 1.1.** *Given a set of states $Q$, and a set of clocks $H$, a* timed path[1] *$\tau = (q_i, v_i, I_i)$ is a (finite or infinite) sequence s.t.:*

- *$(q_i)$ is a sequence of states in $Q$;*
- *$(I_i)$ is a sequence of intervals forming a partition of $\mathbb{R}^+$ (or possibly of an interval $[0,p]$ or $[0;p)$ in the case of a finite path);*
- *$v_i \colon H \to \mathbb{R}^+$ is the valuation of clocks in $H$ when entering location $q_i$, at date $l(I_i)$. We require that, for each $i$ and each clock $x$, either $v_{i+1}(x) = 0$ or $v_{i+1}(x) = v_i(x) + r(I_i) - l(I_i)$;*

[1] For the sake of brevity, we only consider dense time in the sequel. However, our results still hold when considering super-dense time [9].

– *for each $i$, either $q_{i+1} \neq q_i$, or there exists a clock $x$ s.t. $v_{i+1}(x) \neq v_i(x) + r(I_i) - l(I_i)$. This ensures that, at each step along that sequence, either we change location or we reset at least one variable*[2].

A *position* along a timed path $\tau = (q_i, v_i, I_i)$ is a triple $(q, v, t) \in Q \times \mathbb{R}^H \times \mathbb{R}$ for which there exists an integer $j$ s.t. $q = q_j$ and $v = v_j + t - l(I_j)$ and $t \in I_j$. For each $t \in \cup_i I_i$, there exists exactly one position $(q, v, t)$ along $\tau$, which we denote by $\tau(t)$. Given a timed path $\tau = (q_i, v_i, I_i)$ and a position $(q_j, v, t)$ along $\rho$, the suffix of $\rho$ starting at position $(q_j, v, t)$, denoted by $\rho^{\geq t}$, is the timed path $(q'_i, v'_i, I'_i)$ where (1) $q'_i = q_{i+j}$ for all $i$, (2) $v'_i = v_{i+j}$ for $i > 0$, and $v'_0 = v$, (3) $I'_i = I_{i+j} - t$ for $i > 0$, and $I'_0 = ([t, +\infty) \cap I_j) - t$.

**Definition 1.2.** *A* timed automaton *(TA) is a* 6*-tuple* $\mathcal{A} = (Q, Q_0, H, l, \mathrm{Inv}, T, F)$ *where: $Q$ is a (finite) set of states; $Q_0$ is a subset of $Q$ containing the set of initial states; $H$ is a finite set of real-valued clocks; $l$ is a function $Q \to 2^{AP}$ labeling each state with atomic propositions of $AP$;* Inv *is a function $Q \to \mathbb{C}(H)$ labeling each state with a set of timing constraints (called "invariants"); $T \subseteq Q \times \mathbb{C}(H) \times 2^H \times Q$ is a set of transitions; $F \subseteq Q$ is a subset of $Q$ containing the set of accepting states.*

In the sequel, we generally identify a location $q \in Q$ with its labeling $l(q)$, if no ambiguity may arise from this notation. A *position* in a TA is a couple $(q, v)$ where $q$ is a state and $v$ is a valuation of clocks in $H$ satisfying $\mathrm{Inv}(q)$.

**Definition 1.3.** *Given a set of states $Q$ and a set of clocks $H$, a timed path $(q_i, v_i, I_i)$ is a* concretization *of a TA $(Q, Q_0, H, l, \mathrm{Inv}, T)$ if*

– *$q_0 \in Q_0$;*
– *For each $j$, and for each $t \in I_j$, valuation $v_j + t - l(I_j)$ satisfies* $\mathrm{Inv}(q_j)$*;*
– *For each $j$, there exists a transition $(q_j, \varphi, C, q_{j+1}) \in E$ s.t. valuation $v_j + r(I_j) - l(I_j)$ satisfies $\varphi$, and for all $x \in C$, $v_{j+1}(x) = 0$, and for all $x \in H \smallsetminus C$, $v_{j+1}(x) = v_j(x) + r(I_j) - l(I_j)$.*
– *either the timed path is infinite or its last state $q_n$ is accepting, that is $q_n \in F$.*

**Definition 1.4.** *Two clock valuations $v$ and $v'$ are said to be* equivalent *w.r.t. a family $(c_x)_{x \in H}$ of constants, if the following conditions hold:*

– *for all clocks $x \in H$, either both $v(x)$ and $v'(x)$ are greater than $c_x$, or both have the same integer part;*
– *for all clocks $x \in H$, if $v(x) \leq c_x$, then $v(x) \in \mathbb{N}$ iff $v'(x) \in \mathbb{N}$;*
– *for all $x, y \in H$ with $v(x) \leq c_x$ and $v(y) \leq c_y$, if $\mathrm{fract}(v(x)) \leq \mathrm{fract}(v(y))$, then $\mathrm{fract}(v'(x)) \leq \mathrm{fract}(v'(y))$, where* fract *stands for the fractional part.*

This obviously defines an equivalence relation. A *clock region* is an equivalence class for the equivalence relation between clocks. [2] proves that there are finitely many clock regions, more precisely at most $|H|! \cdot 4^{|H|} \cdot \prod_{x \in H} (c_x + 1)$.

[2] This conditions rules out "stuttering" paths. This is not restrictive as our logics, as you'll see later, cannot distinguish between timed traces with or without stuterring.

A clock region $\alpha$ is a *time-successor* of a clock region $\beta$ if for each valuation $v \in \beta$, there exists a positive $t \in \mathbb{R}$ s.t. valuation $v + t$ is in $\alpha$, and for each $t'$ s.t. $0 \leq t' \leq t$, valuation $v + t'$ is in $\alpha \cup \beta$. It can be proved that, each clock region $\alpha$ has exactly one time-successor, which we will denote by $\mathrm{succ}(\alpha)$ in the sequel. A clock region $\alpha$ is a *boundary class* if for any valuation $v \in \alpha$ and for any positive real $t$, valuation $v + t$ is not in $\alpha$.

**Definition 1.5.** *Given a TA $\mathcal{A} = (Q, Q_0, H, l, \mathrm{Inv}, T, F)$, and the family $(c_x)$ of maximal constants to which each clock $x$ is compared in $\mathcal{A}$, the* region graph $\mathcal{R}_{\mathcal{A}}$ *of $\mathcal{A}$ is the labeled graph $(V, l', E)$ defined as follows:*

- *$V$ is the product of the set of states of $\mathcal{A}$ and the set of clock regions;*
- *$l' : V \to 2^{AP}$ is defined by $l'(q, \alpha) = l(q)$;*
- *$E$ is the set of edges, containing two type of edges: Edges representing the elapse of time: for each vertex $(q, \alpha)$ in $V$, there is an edge to $(q, \mathrm{succ}(\alpha))$, if* $\mathrm{succ}(\alpha)$ *exists and contains a valuation satisfying the invariant* $\mathrm{Inv}(q)$*; Edges corresponding to transitions in $\mathcal{A}$: for each vertex $(q, \alpha)$ in $V$, for each edge $(q, \varphi, C, q')$ in $T$, if there exists a valuation $v \in \alpha$ satisfying $\varphi$ and s.t. $v_C$ satisfies* $\mathrm{Inv}(q')$*, then there is an edge from $(q, \alpha)$ to $(q', \beta)$ where $\beta$ is the region containing valuation $v_C$.*

**Definition 1.6.** *A* region path *is a (finite or infinite) sequence $\rho = (q_i, \alpha_i)$ where $q_i$ are locations and $\alpha_i$ are regions s.t. for all $i$ either $\alpha_{i+1} = \mathrm{succ}(\alpha_i)$, and $q_{i+1} = q_i$, or there exists a valuation $v \in \alpha_i$ and a set of clocks $C$ s.t. $v_C \in \alpha_{i+1}$.*

**Definition 1.7.** *A* zone *is a convex union of regions. It can equivalently be defined as the set of clock valuations satisfying a difference constaint in $\mathbb{C}(H)$. A* zone path *is a (finite or infinite) sequence $\rho = (q_i, Z_i, C_i)$ where $q_i$ are locations, $Z_i$ are zones and $C_i$ are the sets of clocks that are reset when entering $Z_i$.*

A region (resp. zone) path $\pi$ is said to be *ultimately periodic* (u.p. for short) if it can be written under the form $u \cdot v^\omega$, where $u$ and $v$ are finite region (resp. zone) paths. In both cases, finite paths are special cases of u.p. paths. A timed path is *ultimately periodic* if it is finite or if there exist two integers $m$ and $p > 0$, and a real $t$, s.t. for any $i \geq m$, $q_{i+p} = q_i$, $v_{i+p} = v_i$, and $I_{i+p} = I_i + t$.

Note that a finite (or u.p.) region path is a special case of a TA, where states are pairs $(q_i, \alpha_i)$, the set of initial states is the singleton $\{(q_0, \alpha_0)\}$, invariants are region constraints, clocks that are reset are clocks whose value is 0 when entering the target region, and the set of final states $F$ is the last state pair $(q_n, \alpha_n)$ if the path is finite and is empty otherwise. A *concretization* of a region path is a concretization of the corresponding TA. The following proposition provides a simplified characterization.

**Proposition 1.8.** *Let $\rho = (p_i, \alpha_i)_i$ be a region path. We say that a timed path $\pi = (q_j, v_j, I_j)_j$ is* compatible with *$\rho$, or is a* concretization of *$\rho$, iff* (1) *$\rho$ and $\pi$ are either both finite or both infinite, and for all $k$, $p_k = q_k$,* (2) *for all $j$, for all $t \in I_j$, valuation $v_j + t - l(I_j)$ belongs to region $\alpha_j$.*

Similarly, finite or u.p. zone paths form another subclass of the class of TA. We have the following simplified characterization of a *concretization* for a zone path:

**Proposition 1.9.** *Let $\rho = (p_i, Z_i, C_i)_i$ be a zone path. We say that a timed path $\pi = (q_j, v_j, I_j)_j$ is* compatible with *$\rho$, or is a* concretization of *$\rho$, iff (1) $\rho$ and $\pi$ are either both finite or both infinite, and for all $k$, $p_k = q_k$, (2) for all $k$, for all $t \in I_k$, valuation $v_k + t - l(I_k)$ belongs to zone $Z_k$, (3) for all $k$, for all $x \in C_k$, $v_k(x) = 0$.*

Note that a concretization of an u.p. region (or zone) path is generally not u.p. However, verifying that an u.p. timed path is a concretization of a region (or zone) path may be done in polynomial time [5].

### 1.2 Timed Temporal Logics

**Definition 1.10.** *Let $AP$ be a set of atomic propositions. The logic* $\mathsf{MTL}$ *is defined as follows:*

$$\mathsf{MTL} \ni \varphi, \psi \ ::= \ \neg\varphi \mid \varphi \vee \psi \mid \varphi \, \mathbf{U}_I \, \psi \mid p \mid q \mid ...$$

*where $I$ is an interval with integer greatest lower and least upper bounds and $p, q, ...$ belong to $AP$. The logic* $\mathsf{MITL}$ *is the sub-logic of* $\mathsf{MTL}$ *where intervals may not be singular.*

$\mathsf{MTL}$ (and $\mathsf{MITL}$) formulas are interpreted along timed paths[3]. Given a timed path $\tau = (q_i, v_i, I_i)$ and an $\mathsf{MTL}$ formula $\varphi$, we say that $\tau$ *satisfies* $\varphi$ (written $\tau \models \varphi$) when:

$$\begin{array}{rll}
\text{if } \ \varphi = p & \text{then} & p \in l(q_0) \\
\text{if } \ \varphi = \neg\xi & \text{then} & \tau \not\models \xi \\
\text{if } \ \varphi = \xi \vee \zeta & \text{then} & \tau \models \xi \text{ or } \tau \models \zeta \\
\text{if } \ \varphi = \xi \, \mathbf{U}_I \zeta & \text{then} & \text{there exists a position } (q, v, t) \text{ along } \tau \text{ s.t. } t \in I, \tau^{\geq t} \models \zeta \\
 & & \text{and, for all } t' \in (0; t), \tau^{\geq t'} \models \xi.
\end{array}$$

Standard unary modalities $\mathbf{F}_I$ and $\mathbf{G}_I$ are defined with the following semantics: $\mathbf{F}_I \xi \stackrel{\text{def}}{=} \top \, \mathbf{U}_I \xi$ and $\mathbf{G}_I \xi \stackrel{\text{def}}{=} \neg \mathbf{F}_I \neg \xi$, where $\top$ is always true. We simply write $\mathbf{F}$ and $\mathbf{G}$ for $\mathbf{F}_{\mathbb{R}^+}$ and $\mathbf{G}_{\mathbb{R}^+}$, respectively.

**Definition 1.11.** *Let $\mathcal{A}$ be a TA, and $\varphi$ be an* $\mathsf{MTL}$ *formula. The model checking problem defined by $\mathcal{A}$ and $\varphi$ consists in determining if, for any concretization $\tau$ of $\mathcal{A}$ starting in an initial state, we have that $\tau \models \varphi$.*

**Definition 1.12.** *Let $AP$ be a set of atomic propositions. The logic* $\mathsf{TCTL}$ *is defined as follows:*

$$\mathsf{TCTL} \ni \varphi, \psi \ ::= \ \varphi \vee \psi \mid \neg\varphi \mid \mathbf{E}\,(\varphi \, \mathbf{U}_I \, \psi) \mid \mathbf{A}\,(\varphi \, \mathbf{U}_I \, \psi) \mid p \mid q \mid ...$$

*where $I$ is an interval with integer greatest lower and least upper bounds and $p, q, ...$ belong to $AP$.*

[3] For the sake of simplicity, we interpret $\mathsf{MTL}$ (and $\mathsf{MITL}$) formulas directly on timed paths instead of defining a notion of timed model where states and clocks are hidden.

TCTL formulas are interpreted at a position in a TA. Given a TA $\mathcal{A}$, a position $(q, v)$ and a TCTL formula $\varphi$, we say that *position* $(q, v)$ *in* $\mathcal{A}$ *satisfies* $\varphi$, written $\mathcal{A}, (q, v) \models \varphi$, when:

| | | |
|---|---|---|
| if $\varphi = p$ | then | $p \in l(q_0)$ |
| if $\varphi = \neg\xi$ | then | $\mathcal{A}, (q, v) \not\models \xi$ |
| if $\varphi = \xi \vee \zeta$ | then | $\mathcal{A}, (q, v) \models \xi$ or $\mathcal{A}, (q, v) \models \zeta$ |
| if $\varphi = \mathbf{E}(\xi\, \mathbf{U}_I \zeta)$ | then | there exists a concretization $\tau = (q_i, v_i, I_i)$ of $\mathcal{A}$ s.t. $q_0 = q$ and $v_0 = v$, and a position $(q', v', t')$ along $\tau$, s.t. $t' \in I$, $\mathcal{A}, (q', v') \models \zeta$ and all intermediate position $\tau(t'') = (q'', v'', t'')$ with $0 < t'' < t'$, $\mathcal{A}, (q'', v'') \models \xi$ |
| if $\varphi = \mathbf{A}(\xi\, \mathbf{U}_I \zeta)$ | then | for any concretization $\tau = (q_i, v_i, I_i)$ of $\mathcal{A}$ with $q_0 = q$ and $v_0 = v$, there exists a position $(q', v', t')$ along $\tau$, s.t. $t' \in I$, $\mathcal{A}, (q', v') \models \zeta$ and all intermediate position $\tau(t'') = (q'', v'', t'')$ with $0 < t'' < t'$, $\mathcal{A}, (q'', v'') \models \xi$ |

We also define standard unary abbreviations $\mathbf{E}\,\mathbf{F}_I \xi$, $\mathbf{A}\,\mathbf{F}_I \xi$ and $\mathbf{E}\,\mathbf{G}_I \xi$, $\mathbf{A}\,\mathbf{G}_I \xi$ respectively as $\mathbf{E}(\top\, \mathbf{U}_I \xi)$, $\mathbf{E}(\top\, \mathbf{U}_I \xi)$ and $\neg\mathbf{A}\,\mathbf{F}_I \neg\xi$, $\neg\mathbf{E}\mathbf{F}_I \neg\xi$. We omit the subscript $I$ when it equals $\mathbb{R}^+$.

Since region and zone paths can be seen as TA, satisfaction of a TCTL formula at a position along a region or zone path is defined in the obvious way. Note that contrary to the untimed case [10], TCTL is not equivalent to MTL along a region or zone path, since such a path contains (infinitely) many timed paths.

**Definition 1.13.** *Let $\mathcal{A}$ be a TA, $(q, v)$ be a position of $\mathcal{A}$, and $\varphi$ be a* TCTL *formula. The model-checking problem defined by $\mathcal{A}$, $(q, v)$ and $\varphi$ consists in determining if $\mathcal{A}, (q, v) \models \varphi$.*

In the sequel, for the two problems defined above, we consider the subcases where $\mathcal{A}$ is *(i)* a single finite (or u.p.) timed path, *(ii)* a finite (or u.p.) region path, *(iii)* a finite (or u.p.) zone path.

## 2 Negative results

The main goal of restricting to subclasses of TA is to obtain feasible algorithms for problems that are hard in the general case. This section presents cases where our restrictions are not sufficient and do not reduce complexity.

### 2.1 Linear time logics along ultimately periodic region paths

What we expected most was that model checking MTL would become decidable along an u.p. region path. This is not the case, as shown in Theorem 2.1. The proof of this theorem requires an encoding of a TM computation by timing information only. Remember that the proof for the general model checking problem (for sets of models defined by TA) is simply a reduction from the satisfiability problem of MTL. The technique needed here is different: We encode the tape of an unbounded TM on a unit-length path by an atomic proposition being true for a strictly positive (but as small as we want) amount of time. MTL can distinguish between those two cases, and allows us to ensure that the path really encodes a computation of the TM. See Fig. 1 for an example.

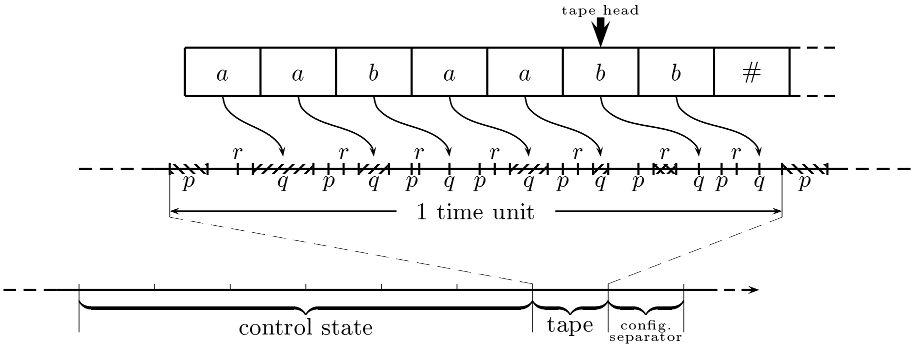



Fig. 1: Encoding of the tape of a Turing Machine

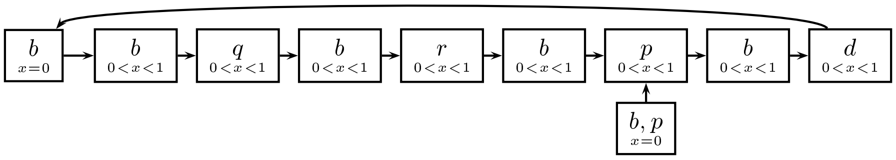



Fig. 2: The region path $\rho$

**Theorem 2.1.** *Model checking a* MTL *formula along an u.p. region path is undecidable.*

*Proof.* This is done by encoding the acceptance problem for a TM (does $\mathcal{M}$ accept $w$?) to the problem of verifying a MTL formula along a region path. Wlog, we assume that the alphabet has only two letters $\{a, b\}$, and a special symbol # for empty cells. Since the ordering of atomic propositions along the path is fixed, the contents of the tape has to be encoded through timing informations only. Since we have no bound on the total length needed for the computation, encoding of one letter must be arbitrarily compressible. Encoding of an $a$ is done by atomic proposition $q$ being true at only one precise moment (with duration 0), while $b$ is encoded by $q$ being true for a positive amount of time. An atomic proposition $p$ is used in the same way for indicating the beginning and end of the encoding of the tape. See top of Fig. 1 for an example. For any atomic proposition $x$, we write $x^+ = x\,\mathbf{U}_{>0}\,\top$ and $x^0 = x \wedge \neg x^+$. Then $a$ is encoded with $p^+$ and $b$ with $p^0$.

A third letter, $r$, is used for encoding the position of the control head: $r^+$ is true (between $p$ and $q$) at the position where the control head stands, and $r^+$ is false everywhere else. Encoding the control state ($s_k$, for some $k$ between 0 and $n-1$) is done through $n$ 1-time-unit-long slices of the path. Along each slice, $q^+$ and $r^+$ will never be satisfied; $p^+$ will be true only in the $k+1$-th slice, meaning that the current control state is $s_k$, and false everywhere else. Fig. 1 shows a complete encoding of one configuration. The configuration separator will be the only slice where $d^+$ will hold, for a fourth atomic proposition $d$. There is one last atomic proposition, $b$, used for filling up all the gaps. The region path generating such an encoding is shown on Fig. 2.

With this encoding, it is possible to write MTL formulas ensuring the correct behavior of the TM. $\square$

In the same way, MITL model checking problems are not easier with u.p. region paths than in the general case. Again, the proof for the general model checking problem is a reduction from the satisfiability problem for MITL. Here, we cannot proceed that way and must encode the computation of an exponential space TM using a single region path and an MITL formula.

**Theorem 2.2.** *Model checking an* MITL *formula along an u.p. region path is EXPSPACE-complete.*

### 2.2 TCTL along finite or ultimately periodic zone paths

Since zones are more general than regions, hardness results for region paths extend to zone paths. Thus model checking MITL and MTL along a zone path is respectively EXPSPACE-complete and undecidable.

Regarding TCTL, the algorithm we propose for region paths (see Section 3.3) could be extended to zone paths, but would result in an exponential explosion in the number of states (since a zone may contain an exponential number of regions). In fact, this explosion cannot be avoided (unless PTIME=PSPACE), since we have the following result:

**Theorem 2.3.** *Model checking* TCTL *along an ultimately periodic zone path is PSPACE-complete.*

## 3 Positive results

Restricting to paths sometimes allows for more efficient algorithms. This happens for MTL and MITL along single timed paths as well as along finite region or zone paths, and for TCTL along u.p. region paths.

### 3.1 Linear time logics and timed paths

Along a timed path, all quantitative information is precisely known, and model checking MTL can be performed quite efficiently.

**Theorem 3.1.** *Model checking* MTL *along a u.p. timed path is in PTIME.*

*Proof.* Consider a finite[4] timed path $\tau = (q_i, v_i, I_i)_{i=0..p}$. The idea is to compute, for each subformula $\psi$ of the MTL formula $\varphi$ under study, the set of reals $t$ s.t. $\tau^{\geq t} \models \psi$. We represent this set $S_\psi$ as a union (which we prove is finite) of intervals whose interiors are disjoint.

The sets $S_\psi = \{J_i^\psi\}$ are computed recursively as follows:

- For atomic propositions, the intervals are trivially computed by "reading" the input path;
- For boolean combinations of subformulas, they are obtained by applying the corresponding set operations, and then possibly merging some of them in order to get disjoint intervals. Obviously the union of two families $\cup_{i=1}^m I_i$ and $\cup_{j=1}^n J_j$ of intervals contains at most $m+n$ intervals, and the complement of $\cup_{i=1}^m I_i$ contains at most $m+1$ intervals. Thus the intersection of $\cup_{i=1}^m I_i$ and $\cup_{j=1}^n J_j$ contains at most $m+n+3$ intervals;

[4] We describe our algorithm only for finite paths, but it can easily be extended to infinite u.p. paths, by reasoning symbolicaly about the periodic part.

– For subformulas of the form $\varphi \mathbf{U}_I \psi$, the idea is to consider, for each interval $J_i^\varphi \in S_\varphi$ and each interval $J_j^\psi \in S_\psi$, the interval $((\overline{J_i^\varphi} \cap J_j^\psi) - I) \cap J_i^\varphi$. It precisely contains all points in $J_i^\varphi$ satisfying $\varphi \mathbf{U}_I \psi$ with a witness for $\psi$ in $J_j^\psi$.
  This construction seems to create $|S_\varphi| \cdot |S_\psi|$ intervals, but a more careful enumeration shows that it only creates at most $|S_\varphi| + |S_\psi| + 3$: indeed, the procedure only creates at most one interval for each *non-empty* interval $\overline{J_i^\varphi} \cap J_j^\psi$, and the intersection of $\cup_{i=1}^m \overline{I_i}$ and $\cup_{j=1}^n J_j$ contains at most $m+n+3$ intervals.

At the end of this procedure, $S_\varphi$ contains $O(p \cdot |\varphi|)$ intervals, and $\tau \models \varphi$ iff 0 is in one of these intervals. Our algorithm thus runs in time $O(|\tau| \cdot |\varphi|)$. ☐

Timed paths could be seen as timed automata if rational difference constraints were allowed in guards and invariants. In that case, the semantics of TCTL along a timed path would have been equivalent to the semantics of MTL, since timed automaton representing a timed path would be completely deterministic.

### 3.2 MTL and MITL along finite region and zone paths

The difficulty for model checking MTL along infinite u.p. region or zone paths was that we had to remember precise timing information about the (infinite, not periodic) concretization against which we verify the MTL formula. In the finite case, we prove we only have to guess and remember a finite (in fact, polynomial) amount of information, making the problem decidable:

**Lemma 3.2.** *Model checking* MTL *along a finite zone path is in co-NP.*

*Proof.* We prove that the *existential* model checking problem is in NP, which is equivalent. The basic idea is to non-deterministically guess the dates $t_i$ at which each of the $n$ transitions is fired. Once these dates are known, we have a timed path and we can check in polynomial time that this path is a concretization of the initial zone path and that it satisfies the MTL formula (see Theorem 3.1).

What remains to be proved is that $t_i$'s can be chosen in polynomial time, *i.e.* the number of non-deterministic steps is polynomial. To that purpose, we consider an MTL formula $\varphi$, and prove that if $\varphi$ is true along the region path, *i.e.* if there exist timestamps *s.t.* the corresponding timed path satisfies $\varphi$, then there exists timestamps in the set $\{p/(n+1) \mid 0 \leq p \leq (n+1) \cdot (c_Z + c_\varphi)\}$ where $n$ is the number of states in the zone path, $c_Z$ is the sum of the constants appearing in the zone path and $c_\varphi$ is the sum of the constants appearing in $\varphi$.

The proof of this last statement is as follows: the set of (in)equalities $t_i$'s must satisfy are: (In)equalities related to the zone path: when $t_i$'s are "fixed", we can compute all valuations of clocks along the zone path. The constraints those valuations must satisfy give constraints that $t_i$'s must satisfy. These constraints have the form $a \leq t_i - t_j \leq b$ or $a \leq t_i \leq b$; (In)equalities related to the formula: for each subformula, we can compute a set of *disjoint* time intervals (depending on $t_i$'s) in which the subformula is true (see proof of Theorem 3.1).

This leads to a disjunction of difference constraints, which has a solution iff the formula is true along one concretization of the finite zone path. Since a difference constraints cannot distinguish between two equivalent valuations (for the equivalence of Definition 1.4), if there exists a solution, any equivalent valuation of $t_i$'s is a solution. This ensures that if there is a solution, then there is a solution in $\{p/(n+1) \mid p \in \mathbb{N}\}$. Moreover, each date can be bounded with the sum of all the constants appearing in the zone path or in the formula: Indeed, constraints between $t_i$'s only involves constants lower than this sum. Thus the dates can be guessed in polynomial time. □

This algorithm is in fact optimal, and we have the following result:

**Theorem 3.3.** *Model checking* MTL *or* MITL *along finite region (or zone) paths is co-NP-complete.*

The co-NP-hardness proof is similar to the one of Theorem 2.3, and consists in encoding 3-SAT into an (existential) model checking problem.

### 3.3 TCTL along ultimately periodic region paths

We prove that TCTL properties can be verified in polynomial time along region paths. This contrasts with the negative results we got previously for MTL and MITL, and intuitively relies on the fact that, contrary to MTL, we don't have to "remember" the precise values of the clocks when we fire a transition, since path quantifiers are applied to all modalities of the formula.

In this section, we describe our algorithm. It first requires to compute temporal relations between any two regions.

**Definition 3.4.** *Let $\rho = (\rho)_i$ be a region path. Given two integers $k$ and $l$, we say that a real $d$ is a* possible delay *between regions $\rho_k$ and $\rho_l$ if there exists a concretization $\pi = (p_j, v_j, I_j)_j$ of $\rho$, and a real $t$, s.t. $t \in I_k$ and $t + d \in I_l$. We write* $\mathrm{delay}(\rho, k, l)$ *for the set of possible delays between $\rho_k$ and $\rho_l$ along $\rho$.*

The following two lemmas prove that possible delays form an interval with integer bounds:

**Lemma 3.5.** *Given a region path $\rho$ and two integers $k$ and $l$,* $\mathrm{delay}(\rho, k, l)$ *is an interval.*

**Lemma 3.6 ([7]).** *Let $\rho$ be a region path, $k$, $l$ and $c$ be three integers. If there exists $d \in (c; c+1)$ s.t. $d \in \mathrm{delay}(\rho, k, l)$, then $(c, c+1) \subseteq \mathrm{delay}(\rho, k, l)$.*

There remains to compute both upper and lower bounds. [8] designed algorithms for computing minimum and maximum delays between valuations and regions. We could apply them in our case. However, their algorithms would compute delays between regions of a finite structure, and we need to compute delays between any two regions of the infinite, u.p. path.

It happens that possible delays in an u.p. region path are u.p., but won't necessarily have the same initial and periodic parts. Below, we compute a table containing the minimum and maximum delays between one region and any future region, by computing those delays for a finite set of regions until a periodicity is detected. Thus, we build a table containing "initial" delays of the minimal and maximal paths, plus the length and duration of their periodic parts.

**Lemma 3.7.** *Let $\rho = u \cdot v^\omega$ be an u.p. region path. We can effectively build in time $O(|u|^2 \cdot |H|)$ the table containing all the necessary information for computing $(i, j) \mapsto \text{delay}(\rho, i, j)$.*

*Proof.* We build the region graph $G$ of the product of $\rho$, seen as a timed automaton, and $T_1$ shown on Fig. 3. Graph $G$ is *not* u.p. in the general case: see Fig. 4 for an example.

Since we add one new clock which is bounded by 1, the total number of regions is at most multiplied by $2(1 + |H|)$, corresponding to the $2(1 + |H|)$ possible ways of inserting $\text{fract}(t)$ among the fractional parts of the other clocks.

In automaton $T_1$, $t$ is the fractional part of the total time elapsed since the beginning of the path, and the number of times $t$ has been reset is the integral part of that total time. Extracting the minimal and maximal delay paths is now an easy task, since in each region of $G$:

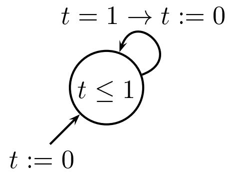
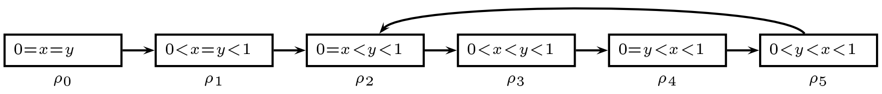



Fig. 3: Automaton $T_1$

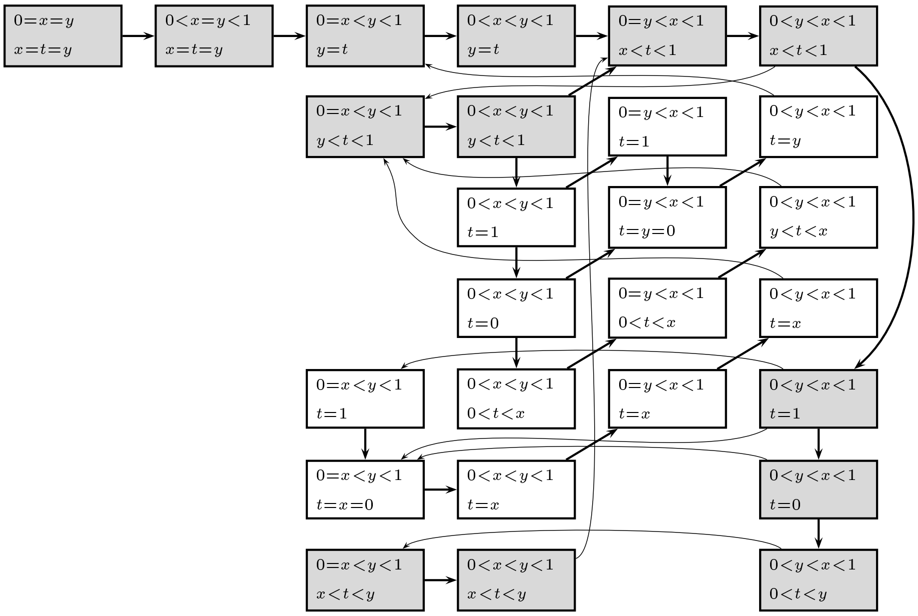
- either $\text{fract}(t) = 0$, and possibly two transitions may be firable: one corresponding to letting time elapse, going to a region where $t > 0$, and the other one corresponding to the transition in $\rho$;
- or $\text{fract}(t) > 0$, and clock $t$ can't reach value 1 in that region, because another clock will reach an integer value before; The only possible outgoing edge is the transition of the original region path;
- or $\text{fract}(t) > 0$, and clock $t$ can reach value 1 (and then be reset to 0). Two cases may arise: resetting $t$ might be the only outgoing transition, or there could be another possible transition derived from the original region path. If there are two outgoing edges, firing the transition that resets $t$ amounts to letting time elapse, and firing the other transition amounts to running as quickly as possible.

In all cases, we also have the condition that we cannot cross two successive immediate transitions, since the resulting region path would not have any concretization.

Now, the maximal delay path is obtained by considering the path where we always select the transition corresponding to time elapsing, *i.e.* resetting $t$ or switching from $t = 0$ to $0 < t < 1$, when such a transition is available; The minimal delay path is the one we get when always selecting the other transition. Moreover, those minimal and maximal delay paths are u.p., since $G$ has finitely many regions and the paths are built deterministically. They have at most $|u| +$




(a) The initial path




(b) The resulting region automaton (highlighted states are states appearing in the minimal or maximal delay path)

| | $\rho_0$ | $\rho_1$ | $\rho_2$ | $\rho_3$ | $\rho_4$ | $\rho_5$ | $\rho'_2$ | $\rho'_3$ | $\rho'_4$ | $\rho'_5$ | period |
|---|---|---|---|---|---|---|---|---|---|---|---|
| min. delay | 0 | $0^+$ | $0^+$ | $0^+$ | $0^+$ | $0^+$ | $0^+$ | $0^+$ | $0^+$ | $0^+$ | 0 t.u., 4 states |
| max. delay | 0 | $1^-$ | $1^-$ | $1^-$ | $1^-$ | $2^-$ | $2^-$ | $2^-$ | $2^-$ | $3^-$ | 1 t.u., 4 states |

(c) Delays from the initial region

Fig. 4: Computation of possible delays between regions

$2(|H|+1)\cdot|v|$ regions in their initial part and at most $2(|H|+1)\cdot|v|$ regions in their periodic part.

From these paths, we can build a table containing all relevant information for computing minimal and maximal delays between the initial region and any region along $\rho$ (see Fig. 4(c)). Any value inbetween is a possible delay thanks to lemma 3.5. Computing this table takes time $O(|u| + 2(|H|+1) \cdot |v|)$. Computing possible delays between any two states along $\rho$ can be achieved by repeating the above procedure starting from the first $|u| + |v|$ states of $\rho$ (since removing longer prefixes gives rise to the same paths), thus in total time $O((|u| + 2(|H| + 1) \cdot |v|) \cdot (|u| + |v|)) \subseteq O(|H| \cdot |\rho|^2)$. ☐

**Theorem 3.8.** *Model checking a* TCTL *formula* $\varphi$ *along an u.p. region path* $\rho$ *can be done in polynomial time (more precisely* $O(|\varphi| \cdot |\rho| \cdot |H| + |H| \cdot |\rho|^2)$*).*

*Proof.* This is achieved by a labeling algorithm. We label region $\rho_i$ of $\rho$ with subformula $\psi$ of $\varphi$ iff $\rho_i \models \psi$. This is not ambiguous as a TCTL formula cannot distinguish between two equivalent valuations [1].

The labeling procedure runs in time $O(|\varphi| \cdot |\rho| \cdot |H|)$. Since delays between regions must be computed, the global TCTL model checking problem along u.p. region paths can be performed in time $O(|\varphi| \cdot |\rho| \cdot |H| + |H| \cdot |\rho|^2)$. ☐

## References


[1] R. Alur, C. Courcoubetis, and D. L. Dill. Model-Checking in Dense Real-Time. *Information and Computation*, 104(1), pages 2–34, Academic Press, May 1993.

[2] R. Alur and D. L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2), pages 183–235, Elsevier Science, Apr. 1994.

[3] R. Alur, T. Feder, and Th. A. Henzinger. The Benefits of Relaxing Punctuality. *Journal of the ACM*, 43(1), pages 116–146, ACM Press, Jan. 1996.

[4] R. Alur and Th. A. Henzinger. A Really Temporal Logic. *Journal of the ACM*, 41(1), pages 181–203, ACM Press, Jan. 1994.

[5] R. Alur, R. P. Kurshan, and M. Viswanathan. Membership Question for Timed and Hybrid Automata. In *Proc. 19th Symp. Real-Time Systems (RTS'98)*, Dec. 1998, pages 254–263. IEEE Comp. Soc. Press, Dec. 1998.

[6] A. Bouajjani, S. Tripakis, and S. Yovine. On-the-Fly Symbolic Model Checking for Real-Time Systems. In *Proc. 18th Symp. Real-Time Systems (RTS'97)*, Dec. 1997, pages 25–35. IEEE Comp. Soc. Press, Dec. 1997.

[7] V. Bruyère, E. Dall'Olio, and J.-F. Raskin. Durations, Parametric Model Checking in Timed Automata with Presburger Arithmetic. In H. Alt and M. Habib, eds, *Proc. 20th Symp. Theoretical Aspects of Computer Science (STACS 2003)*, Feb.-Mar. 2003, vol. 2607 of *LNCS*, pages 687–698. Springer Verlag, Feb. 2003.

[8] C. Courcoubetis and M. Yannakakis. Minimum and Maximum Delay Problems in Real-Time Systems. *Formal Methods in System Design*, 1(4), pages 385–415, Kluwer Academic, Dec. 1992.

[9] Z. Manna and A. Pnueli. Verifying Hybrid Systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds, *Hybrid Systems*, vol. 736 of *LNCS*, pages 4–35. Springer Verlag, 1993.

[10] N. Markey and Ph. Schnoebelen. Model Checking a Path (Preliminary Report). In R. Amadio and D. Lugiez, eds, *Proc. 14th Intl Conf. Concurrency Theory (CONCUR 2003)*, Aug.-Sept. 2003, vol. 2761 of *LNCS*, pages 251–265. Springer Verlag, Aug. 2003.

[11] P. Thati and G. Roşu. Monitoring Algorithms for Metric Temporal Logic Specifications. In K. Havelund and G. Roşu, eds, *Proc. 4th Intl Workshop on Runtime Verification (RV 2004)*, Apr. 2004, ENTCS, pages 131–147. Elsevier Science, Apr. 2004.