Model Checking Restricted Sets of Timed Paths

Nicolas Markey [a] Jean-François Raskin [b]

[a] *Lab. Spécification et Vérification*
ENS Cachan & CNRS UMR8643
61, avenue Président Wilson
94235 Cachan Cedex – France
email: `markey@lsv.ens-cachan.fr`

[b] *Département d'Informatique*
Université Libre de Bruxelles
Bld du Triomphe – CP 212
1050 Brussels – Belgium
email: `jraskin@ulb.ac.be`

Abstract

In this paper, we study the complexity of model-checking formulas of four important real-time logics (TPTL, 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*) an infinite set of finite (or infinite) timed paths defined by a finite (or ultimately periodic) path in a region graph, (*iii*) an infinite set of finite (or infinite) timed paths defined by a finite (or ultimately periodic) path in a zone graph.

Several results are quite negative: TPTL and MTL remain undecidable along region- and zone-paths. On the other hand, we obtained PTIME algorithms for model checking TCTL along a region path, and for MTL along a single timed path.

Key words: Model checking, path checking, timed automaton, timed path, clock region, timed temporal logics.

Introduction

Timed automata have been introduced in [1] 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 [2]. Four main logics have been defined to express real-time requirements: TCTL [3] is a real-time extension of CTL [4], while TPTL [5], MTL [6] and MITL [7] are extensions of LTL [8].

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 [3]. The PSPACE algorithm relies on the *region graph* construction, which is a quotient of the timed automaton by an equivalence relation on clock valuations. In practice, *zones*, *i.e.* convex union of regions, are used, since they may gather up to an exponential number of regions. For the logics TPTL and MTL, the problem has been shown undecidable in [5]. For the logic MITL, the problem has been shown EXPSPACE-complete in [7].

In this paper, we study the model checking problems for those real-time logics over *several classes* of *restricted* sets of timed paths. We consider the model checking problems related to the four logics above 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* forms of timed automata.

Beside the theoretical interest to study the complexity of the model checking problems for those subcases, there are also 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 [9]. 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 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 it 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 examine whether 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 represents 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

<table>
<tr><th></th><th>TPTL</th><th>MTL</th><th>MITL</th><th>TCTL</th></tr>
<tr><td>Timed Automata</td><td colspan="2">Undecidable [5]</td><td>EXPSPACE-c. [7]</td><td>PSPACE-c. [3]</td></tr>
<tr><td>Ult. Per. Zone Paths</td><td colspan="2" rowspan="2">Undecidable</td><td rowspan="2">EXPSPACE-c.</td><td>PSPACE-c.</td></tr>
<tr><td>Ult. Per. Region Paths</td><td>PTIME</td></tr>
<tr><td>Finite Zone Paths</td><td rowspan="4">Decidable
PSPACE-h.</td><td colspan="2" rowspan="2">co-NP-c.</td><td>PSPACE-c.</td></tr>
<tr><td>Finite Region Paths</td><td></td></tr>
<tr><td>Ult. Per. Timed Paths</td><td colspan="3" rowspan="2">PTIME</td></tr>
<tr><td>Finite Timed Paths</td></tr>
</table>

Table 1
Complexity of path model checking

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 in *testing* 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 paths verifies 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 visiting a subset of its region or zone graph. In the two latter cases, we must check real-time formulas over 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 for the four 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-hardness 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 propositional assignments being fixed by the ultimately periodic region path. This situation is rather different from the classic proofs in [5] and in [7]. Note also that the complexity of TCTL model checking goes from

PSPACE-complete to PTIME in the case of region paths; it 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 problems for TPTL becomes *decidable*. For that logic, we prove a PSPACE lower bound but we have no upper bound. As regards MTL, we show that these problems are co-NP-complete. The proofs for these latter results are based on (i) a *polynomial time* algorithm for checking the truth value of an 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]; that paper proposes efficient algorithms for path moedl checking 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 quantifiers would refer to the only possible run of the structure. This does not hold here: region or zone paths may contain infinitely many timed paths.

Runtime verification and monitoring [11] are other related issues. 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 [12].

In our work, we verify properties expressed in four important timed temporal logics. The case where the property is expressed as a timed automaton is treated in [13], where the authors show that deciding whether a finite timed trace corresponds to an execution of a given timed automaton is already PSPACE-complete.

Structure of the paper. The rest of the paper is structured as follows. In the 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 the logics under study. In the 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 just as difficult as in the general case (when the set of timed paths is defined by a timed automaton). In the third section, we present complexity results that can be interpreted as positive: We show that for some interesting classes of restricted sets 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 nonnegative 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 denote the greatest lower bound of I with $l(I)$ and the least upper bound of I with $r(I)$. The closure of an interval I, denoted by $\overline{I}$, is the union $I \cup \{l(I), r(I)\}$. 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 δ is defined in the obvious way. Moreover, for any nonnegative 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 $v[C \leftarrow 0]$ as being the valuation that maps clocks in C to 0 and other clocks to their value according to v.

Definition 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) s.t. for all i, I_{i+1} follows I_i;*
- *$v_i \colon H \to \mathbb{R}^+$ 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 $v_{i+1}(x) = v_i(x) + r(I_i) - l(I_i)$;*
- *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*

[1] For the sake of brevity, we only consider dense time in the sequel. However, our results still hold when considering super-dense time [14].

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 τ, 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 τ, the suffix of τ starting at position (q_j, v, t), denoted by $\tau^{\geq t}$, is the timed path (q'_i, v'_i, I'_i) where

- $q'_i = q_{i+j}$ for all i,
- $v'_i = v_{i+j}$ for $i > 0$, and $v'_0 = v$,
- $I'_i = I_{i+j} - t$ for $i > 0$, and $I'_0 = ([t, +\infty) \cap I_j) - t$.

Definition 2 *Let AP be a finite, non empty set of atomic propositions. A* timed automaton *(TA) is a 6-tuple* $\mathcal{A} = (Q, Q_0, H, lab, \mathrm{Inv}, T, F)$ *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;*
- *lab 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 finite set of transitions;*
- *$F \subseteq Q$ is a subset of Q containing the set of accepting states.*

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

Definition 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, *or is* compatible with, *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 φ, and for all $x \in C$, $v_{j+1}(x) = 0$, and for all $x \in H \setminus C$, $v_{j+1}(x) = v_j(x) + r(I_j) - l(I_j)$.*
- *either the timed path is infinite or its last state q_n is accepting, that is $q_n \in F$.*

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

Definition 4 *Two clock valuations v and v' are said to be* equivalent *w.r.t. a constant M, if the following conditions hold:*

- *for all clocks $x \in H$, either both $v(x)$ and $v'(x)$ are greater than M, or both have the same integer part;*
- *for all clocks $x \in H$, if $v(x) \leq M$, then $v(x) \in \mathbb{N}$ iff $v'(x) \in \mathbb{N}$;*
- *for all $x, y \in H$ with $v(x) \leq M$ and $v(y) \leq M$, 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 of this equivalence relation over the set of clock valuations. In [1], Alur and Dill prove that there are finitely many clock regions, more precisely at most $|H|! \cdot 4^{|H|} \cdot (M+1)^{|H|}$.

A clock region α is a *time-successor* of a clock region β if for each valuation $v \in \beta$, there exists a positive $t \in \mathbb{R}$ s.t. valuation $v + t$ is in α, 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 α, except the region where all clocks are larger than M, has exactly one time-successor, which we will denote by $\mathrm{succ}(\alpha)$ in the sequel [3]. A clock region α is a *boundary class* if for any valuation $v \in \alpha$ and for any positive real t, valuation $v + t$ is not in α.

Definition 5 *Given a TA $\mathcal{A} = (Q, Q_0, H, lab, \mathrm{Inv}, T, F)$, 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, lab', E) defined as follows:*

- *V is the product of the set of states of $\mathcal{A}$ and the set of clock regions w.r.t. the constant $M = \sum_x c_x$;*
- *$lab' \colon V \to 2^{AP}$ is defined by $lab'(q, \alpha) = lab(q)$;*
- *E is the set of edges, containing two type of edges:*
  - · *Edges representing the elapse of time: For each vertex (q, α) 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, α) in V, for each edge (q, φ, C, q') in T, if there exists a valuation $v \in \alpha$ satisfying φ and s.t. $v[C \leftarrow 0]$ satisfies $\mathrm{Inv}(q')$ (or, equivalently, if all valuations in α satisfy these conditions), then there is an edge from (q, α) to (q', β) where β is the region containing valuation $v[C \leftarrow 0]$.*

Definition 6 *A* region path *is a (finite or infinite) sequence $\rho = (q_i, \alpha_i)$ where q_i are locations and α_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. $v[C \leftarrow 0] \in \alpha_{i+1}$.*

Definition 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 π is said to be *ultimately periodic* (u.p. for short) if it can be written in 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 each $i \geq m$, $q_{i+p} = q_i$, $v_{i+p} = v_i$, and $I_{i+p} = I_i + t$. The length of a region (or zone) path π, denoted by $|\pi|$, is the smallest sum $m + p$ for m and p verifying both requirements above.

Note that a finite (or u.p.) region path is a special case of a TA, where states are pairs (q_i, α_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, α_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 8 *Let $\rho = (p_i, \alpha_i)_i$ be a region path. A timed path $\pi = (q_j, v_j, I_j)_j$ is compatible with ρ iff there exists an increasing function f from the index set of $(\rho_i)_i$ onto the index set of $(\pi_j)_j$ s.t.*

- *the index sets of ρ and π are either both finite or both infinite, and for all k, $p_k = q_{f(k)}$,*
- *for all j, for all $t \in I_{f(j)}$, valuation $v_{f(j)} + t - l(I_{f(j)})$ belongs to region α_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 9 *Let $\rho = (p_i, Z_i, C_i)_i$ be a zone path. A timed path $\pi = (q_j, v_j, I_j)_j$ is compatible with ρ iff there exists an increasing function f from the index set of $(\rho_i)_i$ onto the index set of $(\pi_j)_j$ s.t.*

- *ρ and π are either both finite or both infinite, and for all k, $p_k = q_{f(k)}$,*
- *for all k, for all $t \in I_{f(k)}$, valuation $v_{f(k)} + t - l(I_{f(k)})$ belongs to zone Z_k,*
- *for all k, for all $x \in C_k$, $v_{f(k)}(x) = 0$.*

Note that a concretization of an u.p. region (or zone) path is generally not u.p. Consider for instance the u.p. region path displayed on figure 8(a): one concretization of that region path is the path where, clocks x and y are reset when they reach $1/2$, except at the 2^p-th run through the loop, for each integer p, where they are reset when they reach $1/4$. However, verifying that an u.p. timed path is a concretization of a region (or zone) path may be done in polynomial time [13].

Definition 10 *Let AP be a finite, non empty set of atomic propositions. The logic* MTL *is defined as follows:*

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

where I is an interval whose bounds are in $\mathbb{N} \cup \{\infty\}$, and p is any atomic proposition in AP. The logic MITL *is the sub-logic of* MTL *where intervals may not be singular.*

The size of an MTL (or MITL) formula is defined inductively on the structure of the formula. For instance,

$$|\varphi \, \mathbf{U}_{[a,b]} \, \psi| = |\varphi| + 1 + |a| + |b| + |\psi|$$

assuming binary notation for a and b (and $|\infty| = 1$).

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

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

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 $\top$, defined as $p \vee \neg p$ for some atomic proposition p, is always true. We simply write $\mathbf{F}$ and $\mathbf{G}$ for $\mathbf{F}_{\mathbb{R}^+}$ and $\mathbf{G}_{\mathbb{R}^+}$, respectively.

Definition 11 *Let AP be a set of atomic propositions, and V be a set of variables s.t. $V \cap H = \varnothing$. The logic* TPTL *is defined as follows:*

$$\mathsf{TPTL} \ni \varphi, \psi \ ::= \ p \mid \neg\varphi \mid \varphi \vee \psi \mid \varphi \, \mathbf{U} \, \psi \mid x.\varphi \mid \delta$$

where p is any atomic proposition in AP, $x \in V$ is a clock variable, and $\delta \in \mathbb{C}(V)$ is a difference constraint involving only variables in V.

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

As for MTL, the size of a TPTL formula is defined inductively, still using binary notation for integers appearing in difference constraints.

Here again, formulas are interpreted along timed paths, but the interpretation depends on the valuation of variables appearing in the formula: Given a timed path $\tau = (q_i, v_i, I_i)$, an TPTL formula φ and a valuation $\mathcal{E}$ of the variables in the formula, we say that $\tau, \mathcal{E}$ *satisfies* φ (written $\tau \models_{\mathcal{E}} \varphi$) when:

$$\begin{array}{rll}
\text{if } \varphi = p & \text{then} & p \in q_0 \\
\text{if } \varphi = \neg\xi & \text{then} & \tau \not\models_{\mathcal{E}} \xi \\
\text{if } \varphi = \xi \vee \zeta & \text{then} & \tau \models_{\mathcal{E}} \xi \text{ or } \tau \models_{\mathcal{E}} \zeta \\
\text{if } \varphi = \xi \,\mathbf{U}\, \zeta & \text{then} & \text{there exists a position } (q, v, t) \text{ along } \tau \text{ s.t.} \\
& & \tau^{\geq t} \models_{\mathcal{E}+t} \zeta \text{ and, for all } t' \in (0, t),\ \tau^{\geq t'} \models_{\mathcal{E}+t'} \xi. \\
\text{if } \varphi = x - y \sim c & \text{then} & \mathcal{E}(x) - \mathcal{E}(y) \sim c \\
\text{if } \varphi = x \sim c & \text{then} & \mathcal{E}(x) \sim c \\
\text{if } \varphi = x.\,\varphi & \text{then} & \tau \models_{\mathcal{E}[x \leftarrow 0]} \varphi
\end{array}$$

In this definition, $\mathcal{E}+t$ represents the function $\mathcal{E}' \colon x \mapsto \mathcal{E}(x)+t$ for each variable $x \in V$, and $\mathcal{E}[x \leftarrow 0]$ is the function $\mathcal{E}'$ s.t. $\mathcal{E}'(x) = 0$ and $\mathcal{E}'(y) = \mathcal{E}(y)$ for each $y \neq x$. A timed path τ satisfies a TPTL formula φ whenever $\tau \models_{\mathcal{E}_0} \varphi$ with $\mathcal{E}_0(x) = 0$ for each variable $x \in V$.

Obviously enough, MTL formulas can be translated into TPTL. The question whether TPTL is strictly more expressive than MTL has recently been answered (positively): the TPTL formula $x.\mathbf{F}\,(a \wedge x \leq 1 \ \wedge\ \neg b\,\mathbf{U}\,(a \wedge x \geq 1))$ cannot be expressed in MTL [15].

Definition 12 *Let $\mathcal{A}$ be a TA, and φ be an* MTL*,* MITL *or* TPTL *formula. The model checking problem defined by $\mathcal{A}$ and φ consists in determining if, for any concretization τ of $\mathcal{A}$ starting in an initial state, we have that $\tau \models \varphi$.*

Definition 13 *Let AP be a finite, non empty set of atomic propositions. The logic* TCTL *is defined as follows:*

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

where I is an interval with bounds in $\mathbb{N} \cup \{\infty\}$, and p is any atomic proposition in AP.

The size of a TCTL formula is defined in the same way as for MTL.

TCTL formulas are interpreted at a position in a TA. Given a TA $\mathcal{A}$, a position

(q, v) and a TCTL formula φ, we say that (q, v) *in* $\mathcal{A}$ *satisfies* φ, written $\mathcal{A}, (q, v) \models \varphi$, when:

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

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.

Definition 14 *Let $\mathcal{A}$ be a TA, (q, v) be a position of $\mathcal{A}$, and φ be a TCTL formula. The model-checking problem defined by $\mathcal{A}$, (q, v) and φ consists in determining if $\mathcal{A}, (q, v) \models \varphi$.*

In the sequel, for the two problems defined above (Def. 12 and 14), 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 to achieve this goal, and do not reduce complexity.

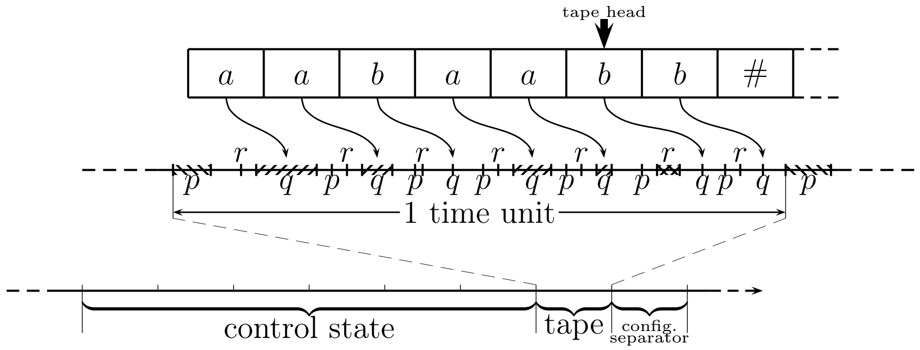


Fig. 1. Encoding of the tape of a Turing Machine

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 15. However, the proof here requires an encoding of a Turing machine, while for the general model checking problem (for sets of models defined by TA), it is simply a reduction from the satisfiability problem for MTL.

Theorem 15 *Model checking a* MTL *formula along an u.p. region path is undecidable.*

PROOF. The proof is by encoding the acceptance problem for a TM (does $\mathcal{M}$ accept w?) to the problem of verifying an MTL formula along a region path. Wlog, we assume that the alphabet of the TM 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 information only. Since we have no bound on the total length needed for the computation, we require that the encoding of one letter can be made arbitrarily small.

Encoding of letter a is done by atomic proposition q being true in a non-empty open interval (thus for a strictly positive amount of time), while letter b is encoded by q being true on a singular interval (with duration 0). To encode this, we define the following abreviations for each atomic proposition x:

$$x^+ \stackrel{\text{def}}{=} \neg x \wedge (x \,\mathbf{U}_{(0,\infty)}\, \neg x) \qquad\qquad x^0 \stackrel{\text{def}}{=} x \wedge (\neg x \,\mathbf{U}_{(0,\infty)}\, \top)$$

Thus letter a is encoded with q^+ being true "not too far away from the next configuration delimiter" (this will be made more precise later), and letter b with q^0. An atomic proposition p is used in the same way for indicating the beginning and end of the encoding of the tape. 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.

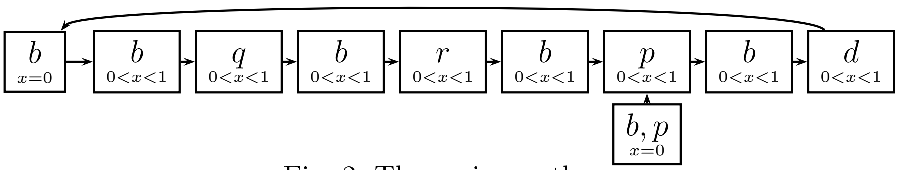


Fig. 2. The region path ρ

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 (except that we omitted d's) 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.

It now suffices to encode the behavior of the TM through an MTL formula. Below, we write a set of formulas whose conjunction Φ satisfies the following equivalence:

$$\mathcal{M} \text{ accepts } w \quad \text{iff} \quad \exists\pi \text{ compatible with } \rho \text{ s.t. } \pi \models \Phi$$

where ρ is the path shown on Fig. 2. This clearly entails undecidability of MTL model checking along a region path.

We first define the following abbreviations:

$$\begin{aligned}
\varphi_{\text{delim}} &\stackrel{\text{def}}{=} p^+ \wedge \mathbf{F}_{[0,1)}\, d^+ & \varphi_{\text{tape}} &\stackrel{\text{def}}{=} p^+ \wedge \mathbf{F}_{[1,1]}\, \varphi_{\text{delim}} \\
\varphi_a &\stackrel{\text{def}}{=} \mathbf{F}_{[0,1)}\, \varphi_{\text{delim}} \wedge q^+ & \varphi_{\text{cs}} &\stackrel{\text{def}}{=} p^+ \wedge \neg\varphi_{\text{tape}} \wedge \neg\varphi_{\text{delim}} \\
\varphi_b &\stackrel{\text{def}}{=} \mathbf{F}_{[0,1)}\, \varphi_{\text{delim}} \wedge q^0 & \varphi_{\overline{\text{cs}}} &\stackrel{\text{def}}{=} p^0 \wedge \neg\varphi_{\text{tape}} \wedge \neg\varphi_{\text{delim}} \\
\varphi_{\text{letter}} &\stackrel{\text{def}}{=} \varphi_a \vee \varphi_b & \varphi_{\text{head}} &\stackrel{\text{def}}{=} r^+ \wedge \mathbf{F}_{[0,1)}\, \varphi_{\text{delim}}
\end{aligned}$$

We split the formulas to be verified in several groups:

- Except for b, we first require that intervals where an atomic proposition holds are either singular or closed: For each atomic proposition x in $\{p, q, r, d\}$, we require the following:

  $$\neg\mathbf{F}\,(\neg x\,\mathbf{U}\,(x \wedge x\,\mathbf{U}_{>0}\,\top)) \;\wedge\; \neg\mathbf{F}\,(x\,\mathbf{U}\,(x \wedge (\neg x)\,\mathbf{U}_{>0}\,\top))$$

  This way, any interval where x holds is either a witness for x^0 or a witness for x^+. Moreover, $(x^0 \vee x^+)$ holds precisely at the left end of the closure of that interval.
- Formulas ensuring that the path is correctly sliced as explained above:

$$(\neg\varphi_{\text{delim}}\,\mathbf{U}_{=n+1}\,\varphi_{\text{delim}}) \wedge \mathbf{G}\,(\varphi_{\text{delim}} \leftrightarrow (\neg\varphi_{\text{delim}}\,\mathbf{U}_{=n+2}\,\varphi_{\text{delim}}))$$
$$(\neg\varphi_{\text{tape}}\,\mathbf{U}_{=n}\,\varphi_{\text{tape}}) \wedge \mathbf{G}\,(\varphi_{\text{tape}} \rightarrow (\neg\varphi_{\text{tape}}\,\mathbf{U}_{=n+2}\,\varphi_{\text{tape}}))$$
$$(p^0 \vee p^+) \wedge \mathbf{G}\,(((p^0 \vee p^+) \wedge \neg\varphi_{\text{tape}}) \rightarrow (\neg(p^0 \vee p^+)\,\mathbf{U}_{=1}\,(p^0 \vee p^+)))$$

The first line ensures that the delimiter occurs after $n+1$ time units, and then exactly each $n+1$ time units. The second one checks that the encoding of the tape starts exactly one t.u. before each delimiter. The last line ensures that, except during the encoding of the tape, one run through the loop of the region path takes exactly one t.u.

- Each part between two consecutive delimiters corresponds to a configuration of the TM:

$$\mathbf{G}\,(\varphi_{\text{delim}} \rightarrow ((\neg\varphi_{\text{delim}} \wedge \neg\varphi_{\text{cs}})\,\mathbf{U}\,(\varphi_{\text{cs}} \wedge \neg\varphi_{\text{delim}} \wedge (\neg\varphi_{\text{cs}}\,\mathbf{U}\,\varphi_{\text{delim}}))))$$
$$\mathbf{G}\,(\varphi_{\text{tape}} \rightarrow (\neg\varphi_t h\,\mathbf{U}_{<1}\,(\varphi_{\text{head}} \wedge (\neg\varphi_{\text{head}}\,\mathbf{U}\,\varphi_{\text{delim}}))))$$

These formulas ensure resp. that there is exactly one control state and one tape head in each configuration.

- The initial state is correct (the tape initially contains w):

$$\varphi_{\text{cs}} \wedge \mathbf{F}_{=1}\,\varphi_{\overline{\text{cs}}} \wedge \ldots \wedge \mathbf{F}_{=n-1}\,\varphi_{\overline{\text{cs}}} \wedge \mathbf{F}_{=n}\,(\varphi_{\text{tape}} \wedge$$
$$((\neg q\,\mathbf{U}\,(\varphi_{w_0} \wedge (\neg p\,\mathbf{U}\,(p \wedge \neg q\,\mathbf{U}\,(\varphi_{w_1} \wedge \ldots\ \mathbf{U}\,(\varphi_{w_k} \wedge \neg p\,\mathbf{U}\,\varphi_{\text{delim}}) \cdots))))$$

- Transitions are applied correctly between two successive configurations: There are several cases here.
  · First, we have to ensure that we always have at least as many cells on the tape from one configuration to the next one:

$$\mathbf{G}\,((\mathbf{F}_{<1}\,\varphi_{\text{delim}} \wedge p) \rightarrow \mathbf{F}_{=n+2}\,p)$$

  The following formula ensures that we will not add one cell between the other ones, by ensuring that no p is added. We express that b's that are followed by a p before the configuration delimiter may only be turned to r's or q's, or left unchanged:

$$\mathbf{G}\,((\mathbf{F}_{<1}\,\varphi_{\text{delim}} \wedge (\neg\varphi_{\text{delim}}\,\mathbf{U}\,(p \wedge \neg\varphi_{\text{delim}})) \wedge b) \rightarrow \mathbf{F}_{=n+2}\,(b \vee r \vee q))$$

  We now express that, except for the cell that is under the tape head, letters are copied verbatim from one configuration to the next one:

$$\mathbf{G}\,((\mathbf{F}_{<1}\,\varphi_{\text{delim}} \wedge p^0 \wedge (\neg r\,\mathbf{U}\,(r^0 \wedge \neg\varphi_{\text{letter}}\,\mathbf{U}\,\varphi_a)))$$
$$\rightarrow \mathbf{F}_{=n+2}\,(p^0 \wedge \neg\varphi_{\text{letter}}\,\mathbf{U}\,\varphi_a))$$

$$\mathbf{G}\,((\mathbf{F}_{<1}\,\varphi_{\text{delim}} \wedge p^0 \wedge (\neg r\,\mathbf{U}\,(r^0 \wedge \neg\varphi_{\text{letter}}\,\mathbf{U}\,\varphi_b)))$$
$$\rightarrow \mathbf{F}_{=n+2}\,(p^0 \wedge \neg\varphi_{\text{letter}}\,\mathbf{U}\,\varphi_b))$$

· For transition $(s_i, a) \to (s_j, b, \mathrm{Right})$, two cases may arise, depending on whether we must "add" a new cell at the end of the tape or not. In the first case:

$$\begin{aligned}&\mathbf{G}\,((\mathbf{F}_{=n+1}\,\varphi_{\mathrm{delim}} \wedge \mathbf{F}_{=i}\,\varphi_{\mathrm{cs}} \wedge \mathbf{F}_{<n+1}\,(\varphi_{\mathrm{head}}\wedge\\ &\quad\neg\varphi_{\mathrm{letter}}\,\mathbf{U}\,(\varphi_a \wedge \neg p\,\mathbf{U}\,\varphi_{\mathrm{delim}}))) \to (\mathbf{F}_{=n+2+j}\,\varphi_{\mathrm{cs}} \wedge \mathbf{F}_{<n+1}\,(\varphi_{\mathrm{head}}\wedge\\ &\qquad(\neg\varphi_{\mathrm{letter}}\,\mathbf{U}\,(\varphi_a \wedge \mathbf{F}_{=n+2}\,(\varphi_b \wedge (\neg\varphi_{\mathrm{delim}}\,\mathbf{U}\,(p^0 \wedge (\neg p\,\mathbf{U}\,(\varphi_{\mathrm{head}}\wedge\\ &\qquad\qquad(\neg p\,\mathbf{U}\,(\varphi_a \wedge \neg p\,\mathbf{U}\,\varphi_{\mathrm{delim}})))))))))))\end{aligned}$$

In the second case:

$$\begin{aligned}&\mathbf{G}\,((\mathbf{F}_{=n+1}\,\varphi_{\mathrm{delim}} \wedge \mathbf{F}_{=i}\,\varphi_{\mathrm{cs}} \wedge \mathbf{F}_{<n+1}\,(\varphi_{\mathrm{head}}\wedge\\ &\qquad\neg\varphi_{\mathrm{letter}}\,\mathbf{U}\,(\varphi_a \wedge \mathbf{F}_{<1}\,(\neg\varphi_{\mathrm{letter}} \wedge \mathbf{F}_{<1}\,\varphi_{\mathrm{letter}})))) \to\\ &\qquad(\mathbf{F}_{=n+2+j}\,\varphi_{\mathrm{cs}} \wedge \mathbf{F}_{<n+1}\,(\varphi_{\mathrm{head}} \wedge (\neg\varphi_{\mathrm{letter}}\,\mathbf{U}\,(\varphi_a\wedge\\ &\quad\mathbf{F}_{=n+2}\,(\varphi_b \wedge (\neg\varphi_{\mathrm{delim}}\,\mathbf{U}\,(p^0 \wedge (\neg p\,\mathbf{U}\,(\varphi_{\mathrm{head}} \wedge (\neg p\,\mathbf{U}\,\varphi_a)))))))))\wedge\\ &\qquad\mathbf{F}_{=n+2}\,(\mathbf{F}_{<n+2}\,(p^0 \wedge \neg p\,\mathbf{U}\,\varphi_{\mathrm{delim}} \wedge \mathbf{F}_{=n+2}\,(p^0 \wedge \neg p\,\mathbf{U}\,\varphi_{\mathrm{delim}})))))\end{aligned}$$

· Transitions "to the left" are handled in a similar way (except that we must handle separately the case where the tape head is on the first cell of the tape)..

- Last, we must ensure that the accepting state (s_n, say) is eventually reached. This is achieved through formula $\mathbf{F}\,(\varphi_{\mathrm{delim}} \wedge \mathbf{F}_{=n}\,\varphi_{\mathrm{cs}})$.

Now assume there exists a concretization of ρ satisfying the conjunction of those formulas. It suffices to slice that path into $n+2$-time-unit-long slices in order to get an accepting computation of the Turing machine. Conversely, if the Turing machine accepts w, it is possible to encode an accepting execution into a timed path compatible with ρ and satisfying Φ. □

In the same way, MITL model checking problems are not easier over 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 16 *Model checking an* MITL *formula along an u.p. region path is EXPSPACE-complete.*

PROOF. It is well known that the model checking problem for MITL is EXPSPACE-complete on classical TA in dense time [7]. Our problem is thus in EXPSPACE. We prove that it is EXPSPACE-hard by encoding an exponential space deterministic TM as an MITL ultimately periodic region path model checking problem.

Each cell of the tape is encoded using a slice of total duration at most $n+7$, where n is the number of control states in the TM. Along one slice, the first part (starting with atomic proposition l) is used for encoding the content of the current cell, the next one (starting with s) for the control state, the next one (starting with h) for the tape head and the last one (starting with d) for a configuration delimiter. A proposition p is used for the encoding: Letter a is encoded by the next p being true less than 1 t.u. after l, while letter b is encoded by p being true at some point between 1 and 2 t.u. after l. The presence of the tape head is encoded by p being true less than 1 t.u. after h; the end of the encoding of a configuration by p being true less than 1 t.u. after d. As regards control states: State s_k corresponds to p being true between time k and time $k+1$ after s. Note that the relevant value for the control state is the one defined in the (only) configuration where the tape head lies. Fig. 3 shows a sample path, and Fig. 4 shows the region path we consider in the rest of this proof.

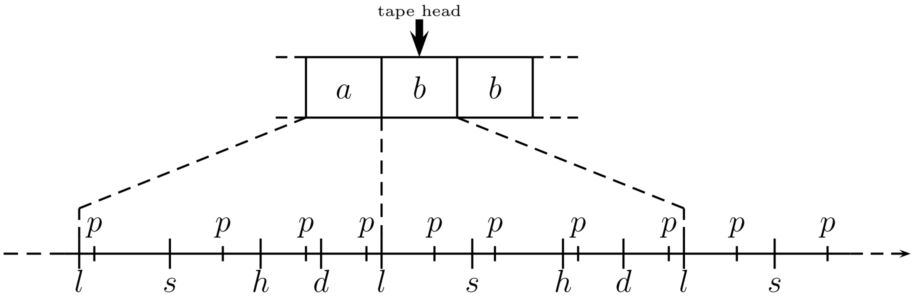


Fig. 3. Encoding a Turing Machine for MITL

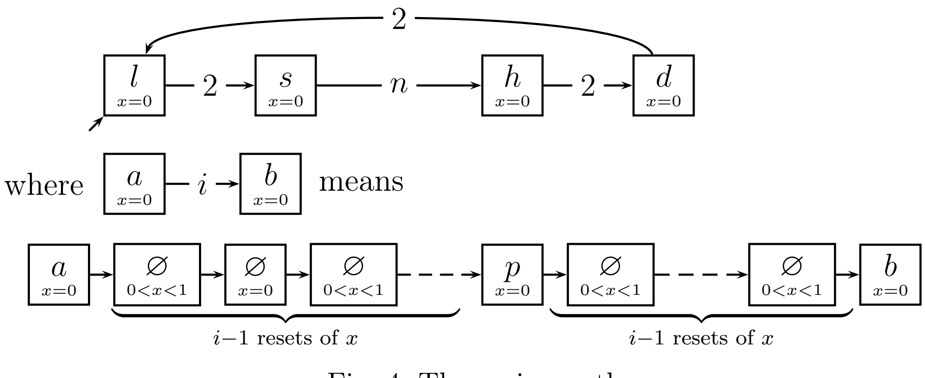


Fig. 4. The region-path

We now enumerate the conditions a path should satisfy to encode a complete computation of the TM. More precisely, for a word w and an exponential space Turing machine $\mathcal{M}$, we define a formula $\Phi \in$ MITL s.t.

$$\mathcal{M} \text{ accepts } w \quad \text{iff} \quad \exists \pi \text{ compatible with } \rho \text{ s.t. } \pi \models \Phi$$

where ρ is the region path of Fig. 4. We also require that the size of the formula Φ is polynomial in the sizes of $\mathcal{M}$ and w.

We will use the following abbreviations:

$$\varphi_{\text{delim}} \stackrel{\text{def}}{=} d \wedge \mathbf{F}_{[0,1)}\, p \qquad\qquad \varphi_{\text{head}} \stackrel{\text{def}}{=} h \wedge \mathbf{F}_{[0,1)}\, p$$

$$\varphi_a \stackrel{\text{def}}{=} l \wedge \mathbf{F}_{[0,1)}\, p \qquad \varphi_b \stackrel{\text{def}}{=} l \wedge \mathbf{F}_{[1,2]}\, p \qquad \varphi_\# \stackrel{\text{def}}{=} l \wedge \mathbf{G}_{[0,2]}\, \neg p$$

- We ensure that exactly 2 t.u. elapse between l and s, h and d, and d and l, and exactly n between s and h:

$$\mathbf{G}\,(l \to (\neg s\,\mathbf{U}_{[2,\infty)}\, s \wedge \neg s\,\mathbf{U}_{[0,2]}\, s)) \wedge \mathbf{G}\,(s \to (\neg h\,\mathbf{U}_{[n,\infty)}\, h \wedge \neg h\,\mathbf{U}_{[0,n]}\, h))\wedge$$
$$\mathbf{G}\,(h \to (\neg d\,\mathbf{U}_{[2,\infty)}\, d \wedge \neg d\,\mathbf{U}_{[0,2]}\, d)) \wedge \mathbf{G}\,(d \to (\neg l\,\mathbf{U}_{[2,\infty)}\, l \wedge \neg l\,\mathbf{U}_{[0,2]}\, l))$$

- There is a delimiter every T cells, where T is the length of the tape:

$$\mathbf{G}\,(\varphi_{\text{delim}} \to ((\neg\varphi_{\text{delim}})\,\mathbf{U}_{[((n+7)\cdot T-1),((n+7)\cdot T+1)]}\,\varphi_{\text{delim}}))\wedge$$
$$(\neg\varphi_{\text{delim}})\,\mathbf{U}_{[((n+7)\cdot T-2),((n+7)\cdot T)]}\,\varphi_{\text{delim}}$$

  Since integers are written using binary notation, the size of this formula is polynomial in the size of the TM.
- The initial configuration is correct (the first k cells contain w, where k is the length of w, and the other ones contain #):

$$\mathbf{G}_{[0,1]}\,(l \to \varphi_{w_0}) \wedge \mathbf{G}_{[(n+7),(n+7)+1]}\,(l \to \varphi_{w_1}) \wedge \ldots$$
$$\mathbf{G}_{[(k+1)(n+7),(T-1)(n+7)+1]}\,(l \to \varphi_\#)\wedge$$
$$\mathbf{F}_{[0,n+7)}\,(s \wedge \mathbf{F}_{<1}\, p) \wedge \mathbf{F}_{[0,n+7)}\,\varphi_{\text{head}}$$

- All configurations contain exactly one tape head:

$$\mathbf{G}\,(\varphi_{\text{head}} \to (\neg\varphi_{\text{head}}\,\mathbf{U}\,\varphi_{\text{delim}})) \wedge \mathbf{G}\,(\varphi_{\text{delim}} \to (\neg\varphi_{\text{delim}}\,\mathbf{U}\,\varphi_{\text{head}}))$$

- Transitions are applied correctly. We write this formula for a transition $(s_k, a) \to (s_m, b, \text{RIGHT})$ and leave the other cases since they are similar:

$$\mathbf{G}\Big((\varphi_a \wedge \mathbf{F}_{[0,n+7)}\,(s \wedge \mathbf{F}_{[k,k+1]}\, p \wedge \mathbf{F}_{[k-1,k]}\, p) \wedge \mathbf{F}_{[0,n+7)}\,\varphi_{\text{head}}) \to$$
$$\mathbf{F}_{[(n+7)\cdot T-1,(n+7)\cdot T]}\,(\varphi_b \wedge \mathbf{F}_{[0,n+7)}\,(s \wedge \mathbf{F}_{[m,m+1]}\, p \wedge \mathbf{F}_{[m-1,m]}\, p)\wedge$$
$$\mathbf{F}_{[n+6,n+7]}\,(l \wedge \mathbf{F}_{[0,n+7)}\,\varphi_{\text{head}}))\Big)$$

- Except around the tape head, the content of the tape is not modified between two consecutive configurations: For each letter x in $\{a, b, \#\}$,

$$\mathbf{G}\,((\varphi_x \wedge \mathbf{G}_{[0,n+7)}\,\neg\varphi_{\text{head}}) \to \mathbf{F}_{[(n+7)\cdot T-1,(n+7)\cdot T]}\,\varphi_x)$$

- Finally, we must reach the accepting state (assuming the only accepting state of the TM is s_n):

$$\mathbf{F}\,(s \wedge (\neg p\,\mathbf{U}_{[n-1,n]}\,p))$$

□

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 increase in the number of states (since a zone may contain an exponential number of regions). In fact, this increase cannot be avoided (unless PTIME=PSPACE), since we have the following result:

Theorem 17 *Model checking* TCTL *along an ultimately periodic zone path is PSPACE-complete.*

PROOF. Membership in PSPACE already occurs for general TA. PSPACE-hardness is proved by reduction from QBF. We prove PSPACE-hardness by reducing QBF to our model checking problem. To this aim, consider an instance $\varphi = Q_1 p_1.\ Q_2 p_2. \ldots Q_n p_n.\ \psi$ of QBF, where $\psi = \bigwedge_{j=1}^{m}(l_{j,1} \vee l_{j,2} \vee l_{j,3})$ and $Q_i \in \{\forall, \exists\}$ for any i s.t. $1 \leq i \leq n$.

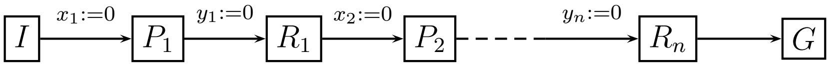
The zone path we consider is made of two parts: In the initial part, we "encode" values of the boolean variables, and in the second part, we check if ψ is satisfied. See Fig. 5 for the details.

We encode the fact that p_i is true by staying exactly one time unit in state P_i, *i.e.* by moving to R_i when $x_i = 1$. In that case, when we enter in_i, $y_i = x_i - 1$ and we can stay in in_i for one time unit, making $\widehat{l_{j,i}}$ true. On the other hand, if we moved to R_i with $x_i < 1$, then when entering in_i, $y_i > 1 - x_i$ and we cannot stay in in_i for one time unit.

We are left to write the TCTL formula to be evaluated along that path. The formula has the following form:

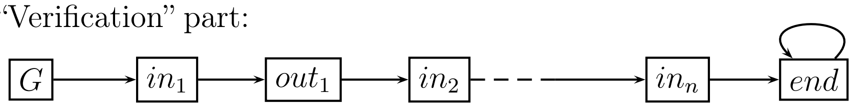
$$\widehat{\varphi} = Q'_1\mathbf{F}\,(P_1 \wedge Q'_2\mathbf{F}\,(P_2 \ldots Q'_n\mathbf{F}\,(P_n \wedge \mathbf{E}\,\mathbf{F}\,(G \wedge \widehat{\psi}))\ldots))$$

“Encoding” part:



with $\mathrm{Inv}(I) = 0 \leq x_1 = y_1 = \cdots = x_n = y_n \leq 1$
$\mathrm{Inv}(P_i) = 0 \leq x_i \leq 1, \quad \mathrm{Inv}(R_i) = 0 \leq y_i \leq 1$

“Verification” part:



with $\mathrm{Inv}(G) = x_1 \leq 2n \qquad \mathrm{Inv}(end) = 2n \leq y_n$
$\mathrm{Inv}(in_i) = y_i \leq 2n \leq x_i \quad \mathrm{Inv}(out_i) = x_i \leq 2n \leq y_{i-1}$

Fig. 5. Encoding QBF for TCTL model checking along a zone path

where $Q'_i = \mathbf{A}$ (resp. $\mathbf{E}$) if $Q_i = \forall$ (resp. $\exists$), and $\widehat{\psi} = \bigwedge_{j=1}^{m} \widehat{l_{j,1}} \vee \widehat{l_{j,2}} \vee \widehat{l_{j,3}}$ with

$$\text{if } l_{j,i} = p_k \text{ then } \widehat{l_{j,i}} = \mathbf{E}\,\mathbf{F}\,(in_k \wedge \mathbf{E}\,\mathbf{F}_{=1}\, in_k)$$
$$\text{if } l_{j,i} = \neg p_k \text{ then } \widehat{l_{j,i}} = \mathbf{E}\,(\neg in_k\, \mathbf{U}\,(in_k \wedge \neg \mathbf{E}\,\mathbf{F}_{=1}\, in_k)$$

□

2.3 TPTL *along finite paths*

It is well-known that the logic TPTL subsumes the logic MTL [5]. As a consequence, we know that the model-checking of TPTL over timed automata, ultimately periodic zone paths and ultimately periodic zone path are undecidable.

Here, we show that model-checking the logic TPTL is harder than MTL. In fact, we show in this subsection that model-checking a TPTL formula against a finite timed path is already PSPACE-hard. In our opinion, this means that TPTL is hardly usable as a specification language within an automatic verification method.

Theorem 18 *Model checking a* TPTL *formula along a finite timed path is PSPACE-hard.*

PROOF. The proof consists in encoding a QBF instance into a TPTL formula that will be evaluated along a simple timed path $\tau = (q, 0, [0, n+1])$ where n is the number of variables involved in the QBF instance.

Let φ be a QBF instance. Wlog, we may assume that it has the form

$$\forall x_1.\, \exists x_2.\, \forall x_3.\, \ldots\; \psi(x_1, x_2, x_3, \ldots).$$

and that each variable x_i is quantified exactly once.

Now consider the TPTL formula obtained by recursively applying the following transformations to φ:

$$\begin{aligned}
\forall x_i.\psi \;\; & \text{translated to } \mathbf{G}\,[(X_i = 0 \vee X_i = 1) \rightarrow X_{i+1}.\,\psi] \\
\exists x_i.\psi \;\; & \text{translated to } \mathbf{F}\,[(X_i = 0 \vee X_i = 1) \wedge X_{i+1}.\,\psi] \\
\psi(x_1, x_2, \ldots) \;\; & \text{translated to } \psi(X_1 - X_2 = 1, X_2 - X_3 = 1, \ldots, X_n - X_{n+1} = 1)
\end{aligned}$$

This TPTL formula involves $n+1$ real variables X_i, and the truth value of each boolean variable x_i of the orignial QBF instance is encoded by the difference $X_i - X_{i+1}$ being compared to 1. This difference is constant once X_{i+1} has been "reset", and it equals 0 or 1 depending on the value of X_i at the time it is reset. Obviously enough, the TPTL formula holds along τ iff the initial QBF instance is true. □

As a corollary, we have that:

Corollary 19 *Model checking a* TPTL *formula along an ultimately periodic timed path is PSPACE-hard.*

The proof above can be trivially adapted to establish the following theorem:

Theorem 20 *Model checking a* TPTL *formula along a finite zone path or a finite region path is PSPACE-hard.*

So, TPTL model-checking is already difficult for our simpliest problems. For the sake of completeness, we show here that those four restricted model-checking problem even if hard are solvable algorithmically. We first show that the model-checking problem of a TPTL formula against a finite zone path or a finite region path is reducible to the validity checking of sentences in the additive theory of the reals.

Theorem 21 *Model checking a* TPTL *formula along a finite zone path or a finite region path is decidable.*

PROOF. We prove the result for finite zone path as it is more general. Let ρ be a finite zone path $(q_0, Z_0, C_0)(q_1, Z_1, C_1) \ldots (q_n, Z_n, C_n)$. Any concretization π of ρ is completely caracterized by the following two items of information:

- the sequence of time stamps $t_0, t_1, \ldots, t_{n-1}, t_n$ where each t_i, for $0 \leq i \leq n-1$, is the time at which π jumps from zone Z_i to zone Z_{i+1}, and t_n is the time of the end of π if π is a finite timed path or $+\infty$ if π is infinite.
- the sequence of bits $b_0, b_1, \ldots, b_n$ that says if at time t_i, π is still in zone Z_i, in that case $b_i = 1$, or already in zone Z_{i+1}, in that case $b_i = 0$. The set of all those sequences is denoted by $\mathcal{B}$.

When $B \in \mathcal{B}$ is fixed, the sequence of time stamps $t_0, t_1, \ldots, t_n$ must satisfy the following set of constraints that can be defined in the additive logic of the reals:

(1) the sequence of time stamps is increasing. So, we have the constraint: $t_1 \leq t_2 \leq \cdots \leq t_n$;

(2) the sequence of time stamps $t_0, t_1, \ldots, t_n$ gives the times at which clocks are reset along the concretization π of ρ. This sequence must be consistent with the constraints on clocks contained in the zones along ρ. As a consequence, for each i, $0 \leq i \leq n$, we have that:
   - for each clock constraint of the form $x \sim c$ in Z_i, let $k \leq i$ be the greatest index such that $x \in C_k$. We construct the following constraint:

$$\forall t : t_{i-1} \sim_1 t \sim_2 t_i : t - t_k \sim c, \text{ if } i \geq 1$$
$$\forall t : 0 \leq t \sim_2 t_0 : t - t_k \sim c, \text{ if } i = 0.$$

   where $\sim_1$ is $<$ if b_{i-1} is 1 and $\leq$ otherwise, and conversely, $\sim_2$ is $\leq$ if b_i is 1, and $<$ otherwise.
   - for each $i \geq 1$ and each clock constraint of the form $x - y \sim c$ in Z_i, let $k_1 \leq i$ (respectively $k_2 \leq i$) be the greatest index such that $x \in C_{k_1}$ (respectively $y \in C_{k_2}$). The associated constraint is:

$$t_{k_2-1} - t_{k_1-1} \sim c$$

   We denote by $\Psi^{\rho}_{B}(t_0, t_1, \ldots, t_n)$ the conjunction of all the constraints above.

   Given a TPTL formula φ, our finite zone path ρ, and a sequence of bits $B = b_0, b_1, \ldots, b_n$, we now explain how to construct a formula of the additive theory of the reals over the free variables $t_0, t_1, \ldots, t_n$ that formalizes the semantics of φ over any concretization π of ρ that respects B.

   Given a proposition p, let $\mathcal{I}(\rho, B, p)$ denote the set of symbolic intervals I_i for each $0 \leq i \leq n$ such that $p \in q_i$. Each such interval I_i is as follows: $l(I_i) = t_{i-1}$, if $i \geq 1$, and $l(I_i) = 0$ if $i = 0$, $r(I_i) = t_i$, and, furthermore, I_i is left-open iff $b_{i-1} = 1$, and I_i is right closed iff $b_i = 1$. This set of intervals will allow us to construct constraints for atomic propositional formulas.

   As TPTL formulas contain clock variables, we need to know at which time each clock variable has last been reset. This is formalized in the following by the function r that maps each clock to the last time at which it has been reset.

We are now fully equipped to define the function Tr that given a TPTL formula φ, a finite zone path ρ, a sequence of bits B, a time variable t and a reset function r returns a formula of the additive logic of the reals that formalizes the semantics of φ at time t along any concretization π of ρ that respect B. This function Tr is defined on the structure of TPTL formulas as follows:

- $Tr(p, \rho, B, t, r) = \bigvee_{I \in \mathcal{I}(\rho,B,p)} t \in I$
- $Tr(x \sim c, \rho, B, t, r) = t - r(x) \sim c$
- $Tr(x.\varphi, \rho, B, t, r) = Tr(\varphi, \rho, B, t, r[x \mapsto t])$
- $Tr(\varphi_1 \wedge \varphi_2, \rho, B, t, r) = Tr(\varphi_1, \rho, B, t, r) \wedge Tr(\varphi_1, \rho, B, t, r)$
- $Tr(\neg\varphi, \rho, B, t, r) = \neg\, Tr(\varphi, \rho, B, t, r)$
- $Tr(\varphi_1 \mathbf{U} \varphi_2, \rho, B, t, r) = \exists t_1 \geq t : Tr(\varphi_2, \rho, B, t_1, r) \wedge \forall t_2 : t < t_2 < t_1 : Tr(\varphi_1, \rho, B, t_2, r)$.

We are now ready to define the reduction of the finite zone path model-checking problem of TPTL to the validity problem of a formula of the additive theory of the reals. In fact, by construction, we have that:

$$\rho \models \varphi \quad \text{iff} \quad \bigwedge_{B \in \{0,1\}^{n+1}} \forall t_0, t_1, \ldots, t_n : (\Psi^\rho_B \rightarrow Tr(\varphi, \rho, B, 0, r_0))$$

where n is the length of ρ and $r_0(x) = 0$ for any clock x that appears in φ. □

An easy adaptation of the last proof allow us to establish that:

Theorem 22 *Model checking a* TPTL *formula along a finite timed path is decidable.*

For utimately periodic path, we need to use the additive theory of the reals extended with the integer predicate. This extended logic allows us to characterize the periodic part to the timed path:

Theorem 23 *Model checking a* TPTL *formula along a ultimately periodic timed path is decidable.*

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.

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

Theorem 24 *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 ψ of the MTL formula φ under study, the set of reals t s.t. $\tau^{\geq t} \models \psi$. We represent this set S_ψ 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;
- 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^φ satisfying $\varphi \, \mathbf{U}_I \, \psi$ with a witness for ψ in J_j^ψ. If $0 \in I$, we must also add intervals J_j^ψ, since those points also satisfy $\varphi \, \mathbf{U}_I \, \psi$ but might not belong to any J_i^φ.

  The construction $((\overline{J_i^\varphi} \cap J_j^\psi) - I) \cap J_i^\varphi$ 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 each step of this procedure, S_φ contains $O(|\tau| \cdot |\varphi|)$ intervals, and $\tau \models \varphi$ iff 0 is in one of these intervals. Our algorithm thus runs in time $O(|\tau| \cdot |\varphi|^2)$. □

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

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

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 25 *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 24).

What remains to be proved is that the 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 φ, and prove that if φ is true along the zone path, *i.e.* if there exist timestamps *s.t.* the corresponding timed path satisfies φ, 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_φ is the sum of the constants appearing in φ.

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 the 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 the 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 the t_i's) in which the subformula is true (see proof of Theorem 24).

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 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 by the sum of all the constants appearing in the zone path or in the formula. Indeed, constraints between the t_i's only involve 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:

Lemma 26 *Model checking* MITL *along a finite region path is co-NP-hard.*

PROOF. Here again, we prove that the dual problem is NP-hard by providing a reduction from 3-SAT. Let $\varphi = \bigwedge_{i=1}^{m} (l_{i,1} \vee l_{i,2} \vee l_{i,3})$ be an instance of 3-SAT, built using a set of atomic propositions $\{p_1, \ldots, p_n\}$. We consider the finite region path ρ shown on Fig. 6. It uses only one clock x, and atomic propositions $\{q_1, \ldots, q_n\}$. The encoding is as follows: an atomic proposition p_i in the 3-SAT instance is evaluated to true iff the corresponding atomic proposition q_i (in the region path) holds for more than 1 t.u. along the concretization.

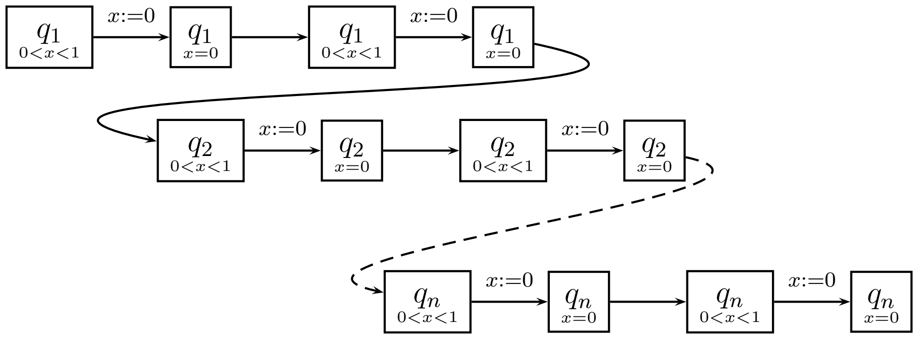


Fig. 6. The path associated to an instance of 3-SAT

Formula φ is then easily transformed into the MITL formula $\widehat{\varphi} = \bigwedge_{i=1}^{m} (\widehat{l_{i,1}} \vee \widehat{l_{i,2}} \vee \widehat{l_{i,3}})$ where $\widehat{l_{i,j}} = \mathbf{F}\,(q_k \wedge \mathbf{F}_{(1,2)}\, q_k)$ if $l_{i,j} = p_k$, and $\widehat{l_{i,j}} = \neg\mathbf{F}\,(q_k \wedge \mathbf{F}_{(1,2)}\, q_k)$ if $l_{i,j} = \neg p_k$. There exists a concretization of ρ satisfying $\widehat{\varphi}$ iff φ is satisfiable. □

The previous two Lemmas entail the following Theorem:

Theorem 27 *Model checking* MTL *or* MITL *along finite region (or zone) paths is co-NP-complete.*

We prove that TCTL properties can be verified in polynomial time along region paths. This contrasts with the negative results we obtained 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 28 *Let $\rho = (\rho_i)_i$ be a region path. Given two integers k and l, we say that a real d is a* possible delay *between regions ρ_k and ρ_l if there exists a concretization $\pi = (p_j, v_j, I_j)_j$ of ρ, and a real t, called the offset, s.t. $t \in I_k$ and $t + d \in I_l$. We write* $\text{delay}(\rho, k, l)$ *for the set of possible delays between ρ_k and ρ_l along ρ.*

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

Lemma 29 *Given a region path ρ and two integers k and l,* $\text{delay}(\rho, k, l)$ *is an interval.*

PROOF. Let d and d' be two possible delays between ρ_k and ρ_l along ρ. According to the definition, let $\tau = (p_j, v_j, I_j)_j$, $\tau' = (p_j, v'_j, I'_j)_j$ be the respective concretizations for d and d', and t and t' be their respective offsets.

Pick some $d'' \in [d, d']$. Since $[d, d']$ is convex, there exists a real $\lambda \in [0, 1]$ s.t. $d'' = \lambda d + (1 - \lambda)d'$. We consider the path $\tau'' = (p_j, v''_j, I''_j)_j$ with $v''_j = \lambda \cdot v_j + (1 - \lambda) \cdot v'_j$ and I''_j is obtained from I_j by replacing $l(I_j)$ and $r(I_j)$ with $\lambda l(I_j) + (1 - \lambda) l(I'_j)$ and $\lambda r(I_j) + (1 - \lambda) r(I'_j)$, respectively. We also define $t'' = \lambda t + (1 - \lambda)t'$.

We claim that τ'' and t'' witness the fact that $d'' \in \text{delay}(\rho, k, l)$:

- τ'' is a timed path: indeed, (I''_k) satisfies the requirements of Definition 1, since I_k and I'_k do. Valuation v''_j fulfills the "continuity" condition for clocks that are not reset: If a clock x is reset when entering a given region ρ_n, then $v_n(x)$ and $v'_n(x)$ both equal 0, and so does $v''_n(x)$.
- τ'' is compatible with ρ, by convexity of regions, and since τ and τ' are;
- since $t \in I_k$ and $t' \in I'_k$, then $t'' \in I''_k$, and since $t + d \in I_l$ and $t' + d' \in I'_l$, then $t'' + d'' \in I''_l$.

□

Lemma 30 ([16]) *Let ρ 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 $(c, c+1) \subseteq \mathrm{delay}(\rho, k, l)$.*

PROOF. We consider the product of the suffix ρ^k (seen as a TA) of ρ starting at state k, with the TA T_1 shown on Fig. 7 (assuming that clock t of T_1 does not occur in ρ). By construction of T_1, clock t represents the fractional part of the total time elapsed since the beginning of the run, and the integer part is the number of times t has been reset.

Consider the region graph corresponding to the resulting TA: By hypothesis, there exists a timed path starting in the initial region (namely $(\rho_k, t = 0)$) and reaching the region $(\rho_l, 0 < t < 1)$, namely with $t = \mathrm{fract}(d)$. Since the transition relation of the region graph is *back-stable* [17], it follows that, for any clock valuation v in $(\rho_l, 0 < t < 1)$, there exists a timed path starting in the initial region $(\rho_k, t = 0)$, crossing exactly the same regions, and ending with valuation v. That is, for any $0 < \delta' < 1$, there exists a timed path starting in region $(\rho_k, t = 0)$, along which t is reset exactly c times, and ending with $t = \delta'$. The same argument can be applied for finding the first part of the path, *i.e.* the offset t'. This proves that $c + \delta'$ is a possible delay between ρ_k and ρ_l along ρ. □

There remains to compute both upper and lower bounds. In [18], Courcoubetis and Yannakakis 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. This table contains all the necessary information for computing the delays between any two regions along the region path.

Lemma 31 *Let H be a set of clocks, and $\rho = u \cdot v^\omega$ be an u.p. region path involving clocks in H. We can effectively compute, in time $O(|\rho|^2 \cdot |H|)$, the function $(i, j) \mapsto \mathrm{delay}(\rho, i, j)$.*

PROOF. We build the region graph G of the product of ρ, seen as a timed automaton, and T_1 shown on Fig. 7. Graph G is *not* u.p. in the general case: see Fig. 8 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 $\mathrm{fract}(t)$ among the fractional parts of the other clocks.

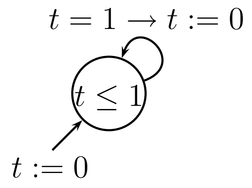


Fig. 7. Automaton T_1

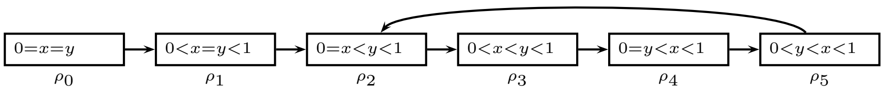
In automaton T_1, t is the fractional part of the total time elapsed since the beginning of the path, and the number of times t has been reset is the integral part of that total time. Extracting the minimal and maximal delay paths is now an easy task, since in each region of G:

- either $\mathrm{fract}(t) = 0$, 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 ρ;
- or $\mathrm{fract}(t) > 0$, 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 $\mathrm{fract}(t) > 0$, 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.

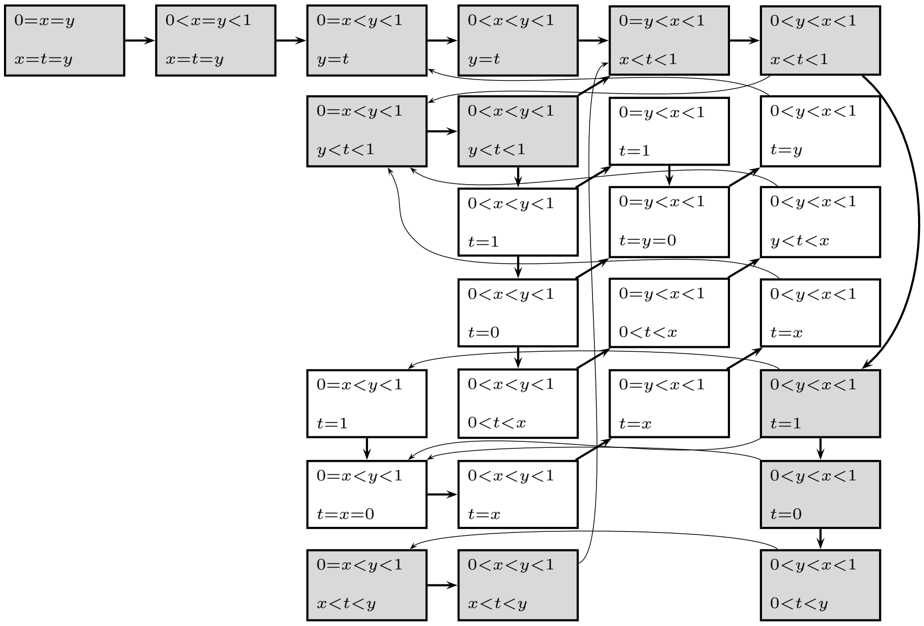
In all cases, we also have the condition that we cannot cross two successive immediate transitions, since the resulting region path would not have any concretization.

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

This construction gives us:



(a) The initial path



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

	ρ_0	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ'_2	ρ'_3	ρ'_4	ρ'_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. 8. Computation of possible delays between regions

- A minimal delay path, which is an ultimately periodic region path with at most $|u| + 2(|H| + 1) \cdot |v|$ regions in its initial part and $2(|H| + 1) \cdot |v|$ in its periodic part;
- A maximal delay path, which is an ultimately periodic region path with at most $|u| + 2(|H| + 1) \cdot |v|$ regions in its initial part and $2(|H| + 1) \cdot |v|$ in its 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 ρ (see Fig. 8(c)). Any intermediate value is a possible delay thanks to Lemma 29. Computing this table takes time $O(|u|+2(|H|+1)\cdot|v|)$. Computing possible delays between any two states along ρ can be achieved by repeating the above procedure starting from the first $|u| + |v|$ states of ρ

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

PROOF. This is achieved by a labeling algorithm. We label region ρ_i of ρ with subformula ψ of φ iff $\rho_i \models \psi$. This is not ambiguous as a TCTL formula cannot distinguish between two equivalent valuations [3].

The algorithm consists in recursively applying the following rules:

- Obvious rules when ψ is an atomic proposition or a boolean combination of already labeled subformulas;
- if $\psi = \mathbf{E}\,\zeta\,\mathbf{U}_I\,\xi$ where I is a bounded interval, label ρ_i with ψ iff there exists a region ρ_j s.t. $\mathrm{delay}(\rho, i, j) \cap I \neq \varnothing$, and ρ_j is labeled with ξ and intermediate regions ρ_m satisfy $\rho_m \models \zeta$. This can be done in time $O(|\rho| \cdot |H|)$, assuming that all possible delays have been computed earlier.
  If I is unbounded, for instance $I = [k, +\infty)$, we consider the earliest (if any) region labeled with ξ and whose possible delays contain a value larger than or equal to k. We label ρ_i with ψ iff such a region exists and all intermediate regions are labeled with ζ.
- if $\psi = \mathbf{E}\,\mathbf{G}_I\,\zeta$, where I is bounded and its lower bound is 0: We consider the example where $I = (0, p)$, the other cases being similar. We look for the earliest region ρ_j whose possible delays from ρ_i contains p. It is necessary and sufficient that all regions between ρ_{i+1} and ρ_{j-1} satisfy ζ for ρ_i to satisfy $\mathbf{E}\,\mathbf{G}_{(0,p)}\,\zeta$.
  If I is bounded but the lower bound is not 0, for instance with $I = (q, p)$, we replace $\mathbf{E}\,\mathbf{G}_I\,\zeta$ with $\mathbf{E}\,\mathbf{F}_{[q,q]}\,\mathbf{E}\,\mathbf{G}_{(0,p-q)}\,\zeta$.
  If I is not bounded, for instance $I = (k, +\infty)$, then $\mathbf{E}\,\mathbf{G}_I\,\zeta$ is equivalent to $\mathbf{E}\,\mathbf{F}_{[k,k]}\,\mathbf{E}\,\mathbf{G}_{(0,\infty)}\,\zeta$. Verifying $\mathbf{E}\,\mathbf{G}_{(0,\infty)}\,\zeta$ is easy, since it is equivalent to verifying that all future states, except possibly the current one, satisfy ζ, and that there exists a non-zeno path.

These modalities are sufficient for handling other TCTL modalities. For instance, $\mathbf{A}\,(\zeta\,\mathbf{U}_{(p,q]}\,\xi)$ is equivalent to

$$\mathbf{A}\,\mathbf{F}_{(p,q]}\,\xi \;\wedge\; \mathbf{A}\,\mathbf{G}_{[0,p]}\,(\zeta \wedge \neg\mathbf{E}\,(\neg\xi\,\mathbf{U}_{(0,\infty)}\,(\neg\zeta \wedge \neg\xi))).$$

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, D. L. Dill, A theory of timed automata, Theoretical Computer Science 126 (2) (1994) 183–235.

[2] A. Pnueli, The temporal logic of programs, in: Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS'77), IEEE Comp. Soc. Press, 1977, pp. 46–57.

[3] R. Alur, C. Courcoubetis, D. L. Dill, Model-checking in dense real-time, Information and Computation 104 (1) (1993) 2–34.

[4] E. M. Clarke, E. A. Emerson, Design and synthesis of synchronous skeletons using branching-time temporal logic, in: D. Kozen (Ed.), Proceedings of the 3rd Workshop on Logics of Programs (LOP'81), Vol. 131 of Lecture Notes in Computer Science, Springer-Verlag, 1981, pp. 52–71.

[5] R. Alur, T. A. Henzinger, A really temporal logic, Journal of the ACM 41 (1) (1994) 181–203.

[6] R. Koymans, Specifying real-time properties with metric temporal logic, Real-Time Systems 2 (4) (1990) 255–299.

[7] R. Alur, T. Feder, T. A. Henzinger, The benefits of relaxing punctuality, Journal of the ACM 43 (1) (1996) 116–146.

[8] D. M. Gabbay, A. Pnueli, S. Shelah, J. Stavi, On the temporal analysis of fairness, in: Conference Record of the 7th ACM Symposium on Principles of Programming Languages (POPL'80), ACM Press, 1980, pp. 163–173.

[9] A. Bouajjani, S. Tripakis, S. Yovine, On-the-fly symbolic model checking for real-time systems, in: Proceedings of the 18th Symposium on Real-Time Systems (RTS'97), IEEE Comp. Soc. Press, 1997, pp. 25–35.

[10] N. Markey, Ph. Schnoebelen, Model checking a path (preliminary report), in: R. Amadio, D. Lugiez (Eds.), Proceedings of the 14th International Conference on Concurrency Theory (CONCUR 2003), Vol. 2761 of Lecture Notes in Computer Science, Springer-Verlag, 2003, pp. 251–265.

[11] K. Havelund, G. Roşu (Eds.), Proceedings of the 1st International Workshop on Runtime Verification (RV 2001), Vol. 55 of Electronic Notes in Theoretical Computer Science, Elsevier Science, 2001.

[12] P. Thati, G. Roşu, Monitoring algorithms for metric temporal logic specifications, in: K. Havelund, G. Roşu (Eds.), Proceedings of the 4th International Workshop on Runtime Verification (RV 2004), Electronic Notes in Theoretical Computer Science, Elsevier Science, 2004, pp. 131–147.

[13] R. Alur, R. P. Kurshan, M. Viswanathan, Membership question for timed and hybrid automata, in: Proceedings of the 19th Symposium on Real-Time Systems (RTS'98), IEEE Comp. Soc. Press, 1998, pp. 254–263.

[14] Z. Manna, A. Pnueli, Verifying hybrid systems, in: R. L. Grossman, A. Nerode, A. P. Ravn, H. Rischel (Eds.), Hybrid Systems, Vol. 736 of Lecture Notes in Computer Science, Springer-Verlag, 1993, pp. 4–35.

[15] P. Bouyer, F. Chevalier, N. Markey, TPTL is strictly more expressive than MTL, to be submitted.

[16] V. Bruyère, E. Dall'Olio, J.-F. Raskin, Durations, parametric model checking in timed automata with Presburger arithmetic, in: H. Alt, M. Habib (Eds.), Proceedings of the 20th Symposium on Theoretical Aspects of Computer Science (STACS 2003), Vol. 2607 of Lecture Notes in Computer Science, Springer-Verlag, 2003, pp. 687–698.

[17] R. Alur, C. Courcoubetis, T. A. Henzinger, Computing accumulated delays in real time systems, Formal Methods in System Design 11 (2) (1997) 137–155.

[18] C. Courcoubetis, M. Yannakakis, Minimum and maximum delay problems in real-time systems, Formal Methods in System Design 1 (4) (1992) 385–415.