Logical Methods in Computer Science
Vol. 8(3:18)2012, pp. 1–27
www.lmcs-online.org

Submitted Jul. 19, 2006
Published Sep. 19, 2012

# WEAK ALTERNATING TIMED AUTOMATA

PAWEŁ PARYS [a] AND IGOR WALUKIEWICZ [b]

[a] Warsaw University, Poland

[b] CNRS and Bordeaux University, France

Abstract. Alternating timed automata on infinite words are considered. The main result is a characterization of acceptance conditions for which the emptiness problem for these automata is decidable. This result implies new decidability results for fragments of timed temporal logics. It is also shown that, unlike for MITL, the characterisation remains the same even if no punctual constraints are allowed.

## 1. Introduction

Timed automata [5] are widely used models of real-time systems. They are obtained from finite automata by adding clocks that can be reset and whose values can be compared with constants. The crucial property of timed automata is that their emptiness is decidable. Some other properties, like universality, are undecidable though. Alternating timed automata have been introduced in [16, 22] following a sequence of results [1, 2, 21] indicating that a restriction to one clock can influence decidability. Indeed, the emptiness and universality problems for one clock alternating timed automata are decidable over finite words. On the contrary, over infinite words both problems remain undecidable even for automata with one clock [25, 17]. All undecidability arguments rely on the ability to express "infinitely often" properties. Our main result shows that once these kind of properties are forbidden the emptiness problem is decidable.

To say formally what are "infinitely often" properties we look at the theory of infinite sequences. We borrow from that theory the notion of an index of a language. It is known that the index hierarchy is infinite with "infinitely often" properties almost at its bottom. From this point of view, the undecidability result mentioned above leaves open the possibility that safety properties and "almost always" properties can be decidable. This is indeed what we prove here.

The automata theoretic approach to temporal logics [27] is by now a standard way of understanding these formalisms. For example, we know that the modal $\mu$-calculus corresponds to all automata, and LTL to very weak alternating automata, or equivalently, to

*1998 ACM Subject Classification:* F.1.1, F.4.3.

*Key words and phrases:* verification, timed systems, alternating timed automata.

[a] Author supported by Polish government grant no. N206 008 32/0810.

[b] Author supported by project DOTS (ANR-06-SETI-003).



DOI:10.2168/LMCS-8(3:18)2012

counter-free nondeterministic automata [30]. By translating a logic to automata we can clearly see combinatorial challenges posed by the formalism. We can also abstract from irrelevant details, such as a choice of operators for a logic. This approach was very beneficial for the development of logical formalisms over sequences.

An automata approach has been missing in timed models for an obvious reason: no standard model of timed automata is closed under boolean operations. Event-clock automata [7] may be considered as an exception, but the price to pay is a restriction on the use of clocks. Alternating timed automata seem to be a good model, although the undecidability result over infinite words shows that the situation is more difficult than for finite sequences.

The idea of restricting to one clock automata dates back at least to [15]. Alternating timed automata were studied in a number of papers [17, 25, 4, 3]. Our main result is that the emptiness problem for alternating timed automata with one clock and "almost always" conditions is decidable. A particular case of such automata is when all the states are accepting. This case was considered by Ouaknine and Worrell [24] who have shown decidability of the emptiness problem under some additional restriction on the form of transitions.

The above mentioned result of Ouaknine and Worrell allowed them to identify a decidable fragment of MTL called Safety MTL. In the present paper we show that our main theorem allows to get a decidable fragment of TPTL [8] with one variable, that we call Constrained TPTL. This fragment contains Safety MTL, allows all "eventually" formulas, and more liberal use of clock constraints. Its syntax has also some similarities with another recently introduced logic: FlatMTL [12, 13]. We give some elements of comparison between the logics later in the paper. In brief, the reason why Constrained TPTL is not strictly more expressive than FlatMTL is that the later includes MITL [6]. This is a sub-logic of MTL where punctualilty constraints are not allowed.

The case of MITL makes it natural to ask what happens to alternating timed automata when we disallow punctual constraints. This is an interesting question also because all known undecidability proofs have used punctual constraints in an essential way. Our second main result (Theorem 5.2), says that the decidability frontier does not change even if we only allow to test if the value of a clock is bigger than 1. Put it differently, it is not only the lack of punctual constraints, but also the very weak syntax of the logic that makes MITL decidable.

We should also discuss the distinction between continuous and pointwise semantics. In the latter, the additional restriction is that formulas are evaluated only in positions when an action happens. So the meaning of $F_{(x=1)}\alpha$ in the continuous semantics is that in one time unit from now formula $\alpha$ holds, while in the pointwise semantics we additionally require that there is an action one time unit from now. Pointwise semantics is less natural if one thinks of encoding properties of monadic predicates over reals. Yet, it seems sufficient for descriptions of behaviors of devices, like timed automata, over time [26]. Here we consider the pointwise semantics simply because the emptiness of alternating timed automata in continuous semantics is undecidable even over finite words. At present it seems that an approach through compositional methods [14] is more suitable to deal with continuous semantics.

Our work inserts itself also into the line of research using well-quasi-orders to solve decidability questions. Particularly close are models of lossy counter machines and their duals: machines with incremental errors. Ouaknine and Worrell have shown the undecidability of

the emptiness problem for ATA over infinite words by reduction to the repeated reachability problem for incremental machines with occurrence testing (ICMOT) [23]. While paper [11] gives a finer analysis of the complexity of several problems for ICMOT, using well-quasi orders it is easy to show that the existence of a computation satisfying "almost always" property is decidable for ICMOT. Nevertheless this observation does not imply decidability of the same problem for ATA, whose structure is more complicated. It is worth to mention that checking the existence of a run satisfying "almost always" property is in general more difficult than checking reachability. Recall for example that the former problem is not decidable for lossy counter machines [18], while reachability is decidable for this model.

The depth of nesting of positive and negative conditions of type "infinitely often" is reflected in the concept of the index of an automaton. Wagner [28], as early as in 1977, established the strictness of the hierarchy of indices for deterministic automata on infinite words. Weak conditions were first considered by Staiger and Wagner [29]. There are several results testifying their relevance. For example Mostowski [19] has shown a direct correspondence between the index of weak conditions and the alternation depth of weak second-order quantifiers. For recent results on weak conditions see [20] and references therein.

The next preliminary section is followed by a presentation of our main decidability result (Theorem 3.1). Section 4 introduces Constrained TPTL, gives a translation of the logic into a decidable class of alternating timed automata, and discusses relations with FlatMTL. The last section presents the accompanying undecidability result (Theorem 5.2).

## 2. Preliminaries

A *timed word* over a finite alphabet $\Sigma$ is a sequence

$$w = (a_1, t_1)(a_2, t_2)\dots$$

of pairs from $\Sigma \times \mathbb{R}_+$. We require that the sequence $\{t_i\}_{i=1,2,\dots}$ is strictly increasing and unbounded. If $t_i$ describes the time when event $a_i$ has occurred then these restrictions say that there cannot be two actions at the same time instance and that there cannot be infinitely many actions in a finite time interval (non Zeno behavior).

We will consider alternating timed automata (ATA) with one clock [17]. Let $x$ be this clock and let $\Phi$ denote the set of all comparisons of $x$ with constants, eg. $(x < 1 \wedge x \geq 0)$.

A one-clock ATA over an alphabet $\Sigma$ is a tuple

$$\mathcal{A} = \langle Q, \Sigma, q_o, \delta, \Omega : Q \to \mathbb{N} \rangle,$$

where $Q$ is a finite set of states and $\Omega$ determines the parity acceptance condition. The transition function of the automaton $\delta$ is a finite partial function

$$\delta : Q \times \Sigma \times \Phi \dot{\to} \mathcal{B}^+(Q \times \{\texttt{nop}, \texttt{reset}\}),$$

where $\mathcal{B}^+(Q \times \{\texttt{nop}, \texttt{reset}\})$ is the set of positive boolean formulas over atomic propositions of the form $\top$, $\bot$, and $(q, f)$ with $q \in Q$ and $f \in \{\texttt{nop}, \texttt{reset}\}$.

Intuitively, automaton being in a state $q$, reading a letter $a$, and having a clock valuation satisfying $\theta$ can proceed according to the positive boolean formula $\delta(q, a, \theta)$. It means that if a formula is a disjunction then it chooses one of the disjuncts to follow, if it is a conjunction then it makes two copies of itself each following one conjunct. If a formula is "atomic", i.e., of the form $(q, \texttt{reset})$ or $(q, \texttt{nop})$ then the automaton changes the state to $q$ and either sets

the value of the clock to 0 or leaves it unchanged, respectively. To simplify the definition of acceptance there is also one more restriction on the transition function:

> *(Partition)* For every $q \in Q$, $a \in \Sigma$ and $v \in \mathbb{R}_+$, there is at most one $\theta$ s.t. $\delta(q, a, \theta)$ is defined, and $v$ satisfies $\theta$.

It is easy to transform an automaton to this form.

The *acceptance condition* of the automaton determines which infinite sequences of states (runs of the automaton) are accepting. A sequence $q_1, q_2, \dots$ satisfies:

- *weak parity condition* if $\min\{\Omega(q_i) : i = 1, 2, \dots\}$ is even,
- *strong parity condition* if $\liminf_{i=1,2,\dots} \Omega(q_i)$ is even.

Observe that the difference between weak and strong conditions is that in the weak case we consider all occurrences of states and in the strong case only those that occur infinitely often. In this paper we will mostly consider automata with weak conditions. Whenever we will be considering strong conditions we will say it explicitly.

For an alternating timed automaton $\mathcal{A}$ and a timed word $w = (a_1, t_1)(a_2, t_2)\dots$ we define the *acceptance game* $G_{\mathcal{A},w}$ between two players: Adam and Eve. Intuitively, the objective of Eve is to accept $w$, while the aim of Adam is the opposite. A *play* starts at the initial configuration $(q_0, 0)$. It consists of potentially infinitely many phases. The $(k+1)$-th phase starts in $(q_k, v_k)$, ends in some configuration $(q_{k+1}, v_{k+1})$ and proceeds as follows. Let $v' := v + t_{k+1} - t_k$. Let $\theta$ be a unique (by the partition condition) constraint such that $v'$ satisfies $\theta$ and $\delta(q_k, a_{k+1}, \theta)$ is defined; if there is no such $\theta$ then Eve is blocked. Now the outcome of the phase is determined by the formula $b = \delta(q_k, a_{k+1}, \theta)$. There are four cases:

- $b = b_1 \wedge b_2$: Adam chooses one of subformulas $b_1$, $b_2$ and the play continues with $b$ replaced by the chosen subformula;
- $b = b_1 \vee b_2$: dually, Eve chooses one of subformulas;
- $b = (q, f) \in Q \times \{\texttt{nop}, \texttt{reset}\}$: the phase ends with the result $(q_{k+1}, v_{k+1}) := (q, f(v'))$ and a new phase starts from this configuration;
- $b = \top, \bot$: the play ends.

The winner of such a play is Eve if she is not blocked, and the sequence ends in $\top$, or it is infinite and the states appearing in the sequence satisfy the acceptance condition of the automaton.

Formally, a play is a finite sequence of consecutive game positions of the form $\langle k, q, v\rangle$ or $\langle k, q, v, b\rangle$, where $k$ is the phase number, $b$ a boolean formula, $q$ a location and $v$ a valuation. A *strategy* of Eve is a mapping which assigns to each such sequence ending in Eve's position a next move of Eve. A strategy is *winning* if all the plays respecting the strategy are winning.

**Definition 2.1** (Acceptance)**.** An automaton $\mathcal{A}$ *accepts* $w$ iff Eve has a winning strategy in the game $G_{\mathcal{A},w}$. By $L(\mathcal{A})$ we denote the language of all timed words $w$ accepted by $\mathcal{A}$.

The *Mostowski index* of an automaton with the, strong or weak, acceptance condition given by $\Omega$ is the pair consisting of the minimal and the maximal value of $\Omega$: $(\min(\Omega(Q)), \max(\Omega(Q)))$. We may assume without a loss of generality that $\min(\Omega(Q)) \in \{0, 1\}$. (Otherwise we can scale down the rank by $\Omega(q) := \Omega(q) - 2$.) Automata with strong conditions of index $(0, 1)$ are traditionally called Büchi automata and their acceptance condition is given by a set of accepting states $Q_+ \subseteq Q$; in our presentation these are states with rank 0.

## 3. Decidability for one-clock timed automata

We are interested in the emptiness problem for one clock ATA. As it was mentioned in the introduction, the problem is undecidable for automata with strong Büchi conditions (strong $(0,1)$ conditions). Here we will show a decidability result for automata with weak acceptance conditions of index $(0,1)$.

**Theorem 3.1.** *It is decidable whether a given one-clock alternating timed automaton with weak* $(0,1)$ *condition accepts some non Zeno timed word. The complexity of the problem is non-primitive recursive.*

The lower bound for the complexity holds already for automata over finite words [17]. So in the rest of this section we give a decidability proof.

Before we start, it will be useful to make a couple of remarks that allow to restrict the form of automata. A weak $(0,1)$ automaton can be also presented as an automaton with a strong $(0,1)$ condition where all transitions from an accepting state, state of rank 0, go only to accepting states. Indeed, once the automaton sees a state of priority 0 then any infinite run is accepting (but there may be runs that get blocked). In the following we will write $Q_+$ for accepting states and $Q_-$ for the other states. For automata presented in this way the strong $(0,1)$ condition says simply: there are only finitely many states from $Q_-$ in the run. So the automaton accepts if Eve has a strategy to reach $\top$, or to satisfy this condition.

We can also make some restrictions on a form of the transition function. We can require that every boolean formula that appears as a value of the function is in a disjunctive normal form. Moreover, we can eliminate the $\bot$ and $\top$ propositions. Proposition $\bot$ can be simulated by a state $q_\bot$ from which there is no transition, and $\top$ by an accepting state $q_\top$ on which the automaton loops on all letters. Observe that this is fine as we have put no restriction on transitions going to accepting states. Finally, we can assume that every disjunct of every transition of $\mathcal{A}$ has some pair with `reset` and some pair with `nop`. This can be guaranteed by adding conjuncts $(q_\top, \texttt{nop})$ and $(q_\top, \texttt{reset})$.

To fix the notation we take a one clock ATA in a form as described above:

$$\mathcal{A} = \langle Q, \Sigma, q_o, \delta, Q_+ \subseteq Q \rangle.$$

This means that for every $q$, $a$, and $\theta$, the formula $\delta(q,a,\theta)$ is in a disjunctive normal form; every disjunct contains a pair with `nop` and a pair with `reset`; there are no $\top$ or $\bot$; if $q \in Q_+$ then only states from $Q_+$ appear in the formula;

Our first step will be to construct some infinite transition system $\mathcal{H}(\mathcal{A})$, so that the existence of an accepting run of $\mathcal{A}$ is equivalent to the existence of some good path in $\mathcal{H}(\mathcal{A})$. In the second step we will use some structural properties of this transition system to show decidability of the problem stated in the theorem.

### 3.1. An abstract transition system.

The goal of this subsection is to define a transition system $\mathcal{H}(\mathcal{A})$ such that existence of an accepting computation of $\mathcal{A}$ is reduced to existence of some special infinite path in $\mathcal{H}(\mathcal{A})$ (Corollary 3.9). This system will be some abstraction of the transition system of configurations of $\mathcal{A}$. While $\mathcal{H}(\mathcal{A})$ will be infinite, it will have some well-order structure and other additional properties that will permit to analyze it.

First, consider an auxiliary labeled transition system $\mathcal{S}(\mathcal{A})$ whose states are finite sets of configurations, i.e., finite sets of pairs $(q,v)$, where $q \in Q$ and $v \in \mathbb{R}_+$. The initial

position in $\mathcal{S}(\mathcal{A})$ is $P_0 = \{(q_0, 0)\}$ and there are transitions of two types $P \overset{t}{\hookrightarrow} P'$ and $P \overset{a}{\hookrightarrow} P'$. Transition $P \overset{t}{\hookrightarrow} P'$ is in $\mathcal{S}(\mathcal{A})$ iff $P'$ can be obtained from $P$ by changing every configuration $(q, v) \in P$ to $(q, v + t)$. Transition $P \overset{a}{\hookrightarrow} P'$ is in $\mathcal{S}(\mathcal{A})$ iff $P'$ can be obtained from $P$ by the following nondeterministic process:

- First, for each $(q, v) \in P$, do the following:
  - let $b = \delta(q, a, \theta)$ for the uniquely determined $\theta$ satisfied in $v$,
  - choose one of disjuncts of $b$, say
$$(q_1, r_1) \wedge \ \cdots \ \wedge (q_k, r_k) \qquad (k > 0),$$
  - let $\mathrm{Next}(q, v) = \{(q_i, r_i(v)) : i = 1 \dots k\}$.
- Then, let $P' := \bigcup_{(q,v)\in P} \mathrm{Next}(q, v)$.

Observe that there may be no $P'$ such that $P \overset{a}{\hookrightarrow} P'$ because for some $(q, v) \in P$ the value $\delta(q, a, \theta)$ required above is not defined.

**Definition 3.2.** We will call a sequence $P_0, P_1, \dots$ of the states of $\mathcal{S}(\mathcal{A})$ *accepting* if the states from $Q_-$ appear only in a finite number of $P_i$.

**Lemma 3.3.** *$\mathcal{A}$ accepts an infinite timed word $(a_0, t_0)(a_1, t_1) \dots$ iff there is an accepting sequence in $\mathcal{S}(\mathcal{A})$:*
$$P_0 \overset{t_0}{\hookrightarrow} P_1 \overset{a_0}{\hookrightarrow} P_2 \overset{t_1}{\hookrightarrow} P_3 \overset{a_1}{\hookrightarrow} P_4 \dots$$

*Proof.* The right to left implication is obvious. For the left to right implication, recall that acceptance of a word by an automaton is defined as existence of a winning strategy for Eve in the acceptance game. This is a game with Büchi conditions, so if Eve has a winning strategy, then she has a memoryless winning strategy. This strategy gives a run of the form required by the lemma. □

Our next goal is to remove time labels on transitions. But we cannot just erase them, as then we will not be able to say if a word is Zeno or not. We start by introducing regions.

Let $d_{max}$ denote the biggest constant appearing in $\delta$, i.e., the transition function of the automaton. Let set $\mathtt{reg}$ of *regions* be a partition of $\mathbb{R}_+$ into $2 \cdot (d_{max} + 1)$ sets as follows:
$$\mathtt{reg} := \{\{0\}, (0, 1), \{1\}, (1, 2), \dots, (d_{max} - 1, d_{max}), \{d_{max}\}, (d_{max}, +\infty)\}.$$
There are three kinds of regions: bounded intervals (denoted $\mathtt{reg}_I$), one-point regions (denoted $\mathtt{reg}_P$), and one unbounded interval $(d_{max}, +\infty)$. We will use the notation $\mathcal{I}_i$ for the region $(i - 1, i)$. In a similar way, $\mathcal{I}_\infty$ will stand for $(d_{max}, +\infty)$. For $v \in \mathbb{R}_+$, let $\mathtt{reg}(v)$ denote the region $v$ belongs to; and let $\mathtt{fract}(v)$ denote the fractional part of $v$.

Let us try to give an intuition behind the way time information will be eliminated. Recall that a state $P$ is a finite set of pairs $(q, v)$. If $v \in I_\infty$ then the precise value of $v$ does not matter from the point of view of the automaton. For other values it is important to look at their fractional parts. Among all $v \notin \mathcal{I}_\infty$ appearing in $P$ take the one with the biggest fractional part. Then, by making the time pass we can get $v$ to a new region without changing the regions of valuations with smaller, but positive, fractional parts. Intuitively this is the smallest delay that makes a visible change to $P$. We will introduce a special label to signal when time progresses in this way. As integer valuation would force us to introduce a cumbersome case distinction we will set things so that they can be avoided.

These remarks lead us to consider a new alphabet:
$$\overline{\Sigma} = \Sigma \cup \{(\mathtt{delay}, \varepsilon)\} \cup (\{\mathtt{delay}\} \times \Sigma),$$

and three new kinds of transitions.

Transition on $a$ will do the action and make some time pass without any valuation changing the region.

> $P \xrightarrow{a} P'$ if $P \overset{a}{\hookrightarrow} P_1 \overset{t_1}{\hookrightarrow} P'$ for some $P_1$, and $t_1 > 0$ such that for every $(q, v) \in P$, the value $v + t_1$ is in the same region as $v$.

For a transition on a letter $(\texttt{delay}, \varepsilon)$, pick a valuation $v$ among these with $\texttt{reg}(v) \neq \mathcal{I}_\infty$ with a maximal $\texttt{fract}(v)$. The transition will make the time pass so that $v$ goes to the next interval region but all valuations with smaller fractional parts do not change their regions:

> $P \xrightarrow{(\texttt{delay},\varepsilon)} P'$ if $P \overset{t_1}{\hookrightarrow} P'_1 \overset{t_2}{\hookrightarrow} P'$ for some $P'_1$ and $t_1, t_2 > 0$ such that there is $(q, v) \in P$, with $v + t_1$ being an integer and $v + t_1 + t_2$ in the following interval region. Moreover, for all $(q', v') \in P$ if $\texttt{fract}(v) \neq \texttt{fract}(v')$ then the value $v' + t_1 + t_2$ is in the same region as $v'$.

Finally, we come to the most complex $(\texttt{delay}, a)$ transition. Even though we did not allow transitions $(\texttt{delay}, \varepsilon)$ to reach one-point regions, it is still important to be able to execute actions in those regions. A transition on $(\texttt{delay}, a)$ permits to reach a one-point region, execute the action, and leave the region.

> $P \xrightarrow{(\texttt{delay},a)} P'$ if $P \overset{t_1}{\hookrightarrow} P_1 \overset{a}{\hookrightarrow} P_2 \overset{t_2}{\hookrightarrow} P'$ for some $P_1, P_2$ and $t_1, t_2 > 0$ such that there is $(q, v) \in P$, with $v + t_1$ being an integer and $v + t_1 + t_2$ in the following interval region. Moreover for all $(q', v') \in P$ if $\texttt{fract}(v) \neq \texttt{fract}(v')$ then the value $v' + t_1 + t_2$ is in the same region as $v'$.

The following lemma shows that with a new alphabet we can replace non Zeno condition by a simple infinitary condition.

**Lemma 3.4.** *There is a non Zeno accepting sequence in $\mathcal{S}(\mathcal{A})$:*

$$P_0 \overset{t_0}{\hookrightarrow} P_1 \overset{a_0}{\hookrightarrow} P_2 \overset{t_1}{\hookrightarrow} P_3 \overset{a_1}{\hookrightarrow} P_4 \dots$$

*iff there is an accepting sequence*

$$P_0 \xrightarrow{\sigma_0} P'_1 \xrightarrow{\sigma_1} P'_2 \dots,$$

*where $\sigma_0, \sigma_1, \dots \in \overline{\Sigma}$ and $(\texttt{delay}, \cdot)$ letters appear infinitely often in the sequence.* □

The next step in the construction is to abstract from valuations in the states of the transition system. Intuitively, we will replace every valuation by its region. To compensate for erasing fractional parts, we will also keep information about the relative order between them. With the construction described in the definition below the states become words from

$$\Lambda_I^* \cdot \Lambda_\infty,$$
$$\text{where } \Lambda_I = \mathcal{P}(Q \times \texttt{reg}_I) \text{ and } \Lambda_\infty = \mathcal{P}(Q \times \{\infty\}).$$

**Definition 3.5.** For a state $P$ of $\mathcal{S}(\mathcal{A})$ we define a word $H(P)$ from $\Lambda_I^* \cdot \Lambda_\infty$ as the one obtained by the following procedure:

- replace each $(q, v) \in P$ by a triple $\langle q, \texttt{reg}(v), \texttt{fract}(v)\rangle$ if $v \leq d_{max}$ (this yields a finite set of triples)
- sort all these triples w.r.t. $\texttt{fract}(v)$ (this yields a finite sequence of triples)
- group together triples having the same value of $\texttt{fract}(v)$ (this yields a finite sequence of finite sets of triples)

- forget $\texttt{fract}(v)$, that is, change every triple $\langle q, \texttt{reg}(v), \texttt{fract}(v)\rangle$ into a pair $(q, \texttt{reg}(v))$ (this yields a finite sequence of finite sets of pairs, a word in $\Lambda_I^*$).
- Add at the end the letter $(\{q : (q, v) \in P, v > d_{max}\}, \mathcal{I}_\infty) \in \Lambda_\infty$.

Finally, we can define $\mathcal{H}(\mathcal{A})$.

**Definition 3.6.** $\mathcal{H}(\mathcal{A})$ is a transition system which has $\Lambda_I^* \times \Lambda_\infty$ as a set of configurations, and for every letter $\sigma \in \overline{\Sigma}$ there is a transition $c \xrightarrow{\sigma} c'$ if there are states $P, P'$ of $\mathcal{S}(\mathcal{A})$ such that $P \xrightarrow{a} P'$ and $H(P) = c$, $H(P') = c'$.

Direct examination of the definition gives us the following.

**Lemma 3.7.** *If $H(P_1) = H(P_2)$ and $P_1 \xrightarrow{\sigma} P_1'$ then $P_2 \xrightarrow{\sigma} P_2'$ with $H(P_1') = H(P_2')$.* □

**Definition 3.8.** We say that a path (equivalently: run, computation) in $\mathcal{H}(\mathcal{A})$ is *good*, if it passes through infinitely many transitions labeled by letters $(\texttt{delay}, \cdot)$. We say that a path (equivalently: run, computation) in $\mathcal{H}(\mathcal{A})$ is *accepting*, if it is good and passes through only finitely many configurations containing states from $Q_-$.

**Corollary 3.9.** *$\mathcal{A}$ accepts an infinite non Zeno timed word iff there is an accepting path in $\mathcal{H}(\mathcal{A})$ starting in configuration $(\{(q_0, \mathcal{I}_1)\}, \{\emptyset, I_\infty\})$.*

*Proof.* $\mathcal{A}$ accepts a non Zeno word iff there is a path in $\mathcal{S}(\mathcal{A})$ satisfying the acceptance condition. By Lemma 3.4 it is equivalent to having a good path in $\mathcal{S}(\mathcal{A})$ with transitions from the alphabet $\overline{\Sigma}$ satisfying the acceptance conditions. Lemma 3.7 implies that this is equivalent to having an accepting path in $\mathcal{H}(\mathcal{A})$. □

We finish the section with a more explicit characterization of transitions in $\mathcal{H}(\mathcal{A})$ that will be used extensively in the decidability proof. The characterisation is spelled out in the next three lemmas whose proofs are obtained directly from the definitions.

**Lemma 3.10.** *Consider a state $(\lambda_1 \dots \lambda_k, \lambda_\infty)$ of $\mathcal{H}(\mathcal{A})$. If $k = 0$ then there is no $(\texttt{delay}, \cdot)$ transition from this state. Otherwise let $\lambda_k' = \{(q, \mathcal{I}_{d+1}) : d < d_{max}, (q, \mathcal{I}_d) \in \lambda_k\}$ and $\lambda_\infty' = \lambda_\infty \cup \{(q, \mathcal{I}_\infty) : (q, \mathcal{I}_{d_{max}}) \in \lambda_k\}$. In $\mathcal{H}(\mathcal{A})$ there is exactly one transition on $(\texttt{delay}, \epsilon)$:*

$$(\lambda_1 \dots \lambda_k, \lambda_\infty) \xrightarrow{(\texttt{delay},\epsilon)} (\lambda_k' \lambda_1 \dots \lambda_{k-1}, \lambda_\infty') \qquad \text{if } \lambda_k' \neq \emptyset,$$
$$(\lambda_1 \dots \lambda_k, \lambda_\infty) \xrightarrow{(\texttt{delay},\epsilon)} (\lambda_1 \dots \lambda_{k-1}, \lambda_\infty') \qquad \text{otherwise.}$$ □

In order to describe transitions of $\mathcal{H}(\mathcal{A})$ on an action $a$, we define an auxiliary notion of a transition from $\lambda \in \mathcal{P}(Q \times \texttt{reg})$. By the partition condition, for every $(q, r) \in \lambda$ there is at most one constraint $\theta$ such that every valuation in $r$ satisfies this constraint and $\delta(q, a, \theta)$ is defined. We choose a conjunct from $\delta(q, a, \theta)$:

$$(q_1, \texttt{nop}) \wedge \dots \wedge (q_l, \texttt{nop}) \wedge (q_1', \texttt{reset}) \wedge \dots \wedge (q_m', \texttt{reset}).$$

From this choice we can obtain two sets: $\mathrm{Next}(q, r) = \{(q_1, r), \dots, (q_l, r)\}$ and $\mathrm{Next}_0(q, r) = \{(q_1', \mathcal{I}_1), \dots, (q_m', \mathcal{I}_1)\}$. We put

$$\lambda \xrightarrow{a}_{\mathcal{A}} (\lambda', \gamma'), \quad \text{where}$$
$$\lambda' = \bigcup_{(q,r)\in\lambda} Next(q, r) \quad \text{and} \quad \gamma' = \bigcup_{(q,r)\in\lambda} Next_0(q, r).$$

Observe that there are as many transitions $\xrightarrow{a}$ from $\lambda$ as there are choices of different conjuncts for each pair $(q, r)$ in $\lambda$. In particular there is no transition if for some pair

the transition function of the automaton is not defined. Notice also that the clock after reseting, described by elements of $\gamma'$, is in interval $I_1$, not in $\{0\}$; this is because we describe a transition of the original automaton followed by a small time elapse.

**Lemma 3.11.** *In $\mathcal{H}(\mathcal{A})$ transitions on an action $a$ have the form*

$$(\lambda_1 \dots \lambda_k, \lambda_\infty) \xrightarrow{a} (\gamma' \lambda'_1 \dots \lambda'_k, \lambda'_\infty),$$

*where $\lambda_i \xrightarrow{a}_{\mathcal{A}} (\lambda'_i, \gamma'_i)$ and $\gamma' = \bigcup \gamma'_i$ (for $i = 1, \dots, k, \infty$).* □

Note that neither $\gamma'$ nor any of $\lambda'_i$ may be empty.

Finally, we have the most complicated case of $(\texttt{delay}, a)$ action.

**Lemma 3.12.** *In $\mathcal{H}(\mathcal{A})$ the transitions on an action $(\texttt{delay}, a)$ have the form*

$$(\lambda_1 \dots \lambda_k, \lambda_\infty) \xrightarrow{(\texttt{delay},a)} (\gamma' \lambda'_1 \dots \lambda'_{k-1}, \lambda''_\infty),$$

*where the elements on the right are obtained by preforming the following steps:*

- *First, we change regions in $\lambda_k$. Every pair $(q, \mathcal{I}_d) \in \lambda_k$ becomes $(q, \{d\})$. Let us denote the result by $\lambda^1_k$.*
- *For $i = 1, \dots, k, \infty$ we take $\lambda'_i, \gamma'_i$ such that: $\lambda^1_k \xrightarrow{a}_{\mathcal{A}} (\lambda'_k, \gamma'_k)$ and $\lambda_i \xrightarrow{a}_{\mathcal{A}} (\lambda'_i, \gamma'_i)$ for $i \neq k$.*
- *We again increase regions in $\lambda'_k$: from $\{d\}$ they become $\mathcal{I}_{d+1}$, or $\mathcal{I}_\infty$ if $d = d_{max}$.*
- *We put $\gamma' = \bigcup \gamma'_i \cup \{(q, \mathcal{I}_d) : (q, \{d\}) \in \lambda'_k, d < d_{max}\}$ and $\lambda''_\infty = \lambda'_\infty \cup \{(q, \mathcal{I}_\infty) : (q, \{d_{max}\}) \in \lambda'_k\}$.* □

We write $c \to c'$, $c \xrightarrow{(\texttt{delay},\cdot)} c'$, $c \twoheadrightarrow c'$, $c \overset{\Sigma^*}{\twoheadrightarrow} c'$ to denote that we may go from a configuration $c$ to $c'$ using one transition, one transition reading a letter of the form $(\texttt{delay}, \cdot)$, any number of transitions or any number of transitions reading only letters from $\Sigma$, respectively.

3.2. **Finding an accepting path in $\mathcal{H}(\mathcal{A})$.** Here we overview the decision procedure, which is described in details in the next subsections. By Corollary 3.9, our problem reduces to deciding if in $\mathcal{H}(\mathcal{A})$ there is a good path with only finitely many appearances of states from $Q_-$. The decision procedure works in two steps. In the first step we compute the set $\widehat{G}$ of all configurations of $\mathcal{H}(\mathcal{A})$ from which there exists a good path. Observe that if a configuration from $\widehat{G}$ has only states from $Q_+$ then there exists an accepting run from this configuration. So, in the second step it remains to consider configurations that have states from both $Q_-$ and $Q_+$. This is relatively easy as an accepting run from such a configuration consists of a finite prefix ending in a configuration without states from $Q_-$ and a good run from that configuration. Hence, there is an accepting run from a configuration iff it is possible to reach from it a configuration from $\widehat{G}$ that has only $Q_+$ states. Once we know $\widehat{G}$, the later problem can be solved using the standard reachability tree technique.

3.3. **Computing accepting configurations.** We start with the second step of our procedure as it is much easier than the first one. We need to decide if from an initial state one can reach a configuration from $\widehat{G}$ having only $Q_+$ states. We can assume that we are given $\widehat{G}$ but we need to discuss a little how it is represented. It turns out that there are useful well-quasi-orders on configurations that allow to represent $\widehat{G}$ in a finitary way (Corollary 3.15)

A *well-quasi-order* is a relation with a property that for every infinite sequence $c_1, c_2, \ldots$ there exist indexes $i < j$ such that the pair $(c_i, c_j)$ is in the relation.

The order we need is the relation, denoted $\preceq$, over configurations of $\mathcal{H}(\mathcal{A})$: we put $(\lambda_1 \ldots \lambda_k, \lambda_\infty) \preceq (\lambda'_1 \ldots \lambda'_{k'}, \lambda'_\infty)$ if $\lambda_\infty \subseteq \lambda'_\infty$ and there exists a strictly increasing function $f\colon \{1, \ldots, k\} \to \{1, \ldots, k'\}$ such that $\lambda_i \subseteq \lambda'_{f(i)}$ for each $i$. Observe that here we use the fact that each $\lambda_i$ is a set so we can compare them by inclusion. This relation is somehow similar to the relation of being a subsequence, but we do not require that the corresponding letters are equal, only that the one from the smaller word is included in the one from the greater word. The first property of this order is proved by a standard application of Higman's lemma.

**Lemma 3.13.** *The relation $\preceq$ is a well-quasi-order.* □

The following shows an important interplay between $\preceq$ relation and transitions of $\mathcal{H}(\mathcal{A})$.

**Lemma 3.14.** *Let $c_1, c'_1, c_2$ be configurations of $\mathcal{H}(\mathcal{A})$ such that $c'_1 \preceq c_1$. Whenever $c_1 \twoheadrightarrow c_2$, then there exist $c'_2 \preceq c_2$ such that $c'_1 \twoheadrightarrow c'_2$ and the second computation has the length not greater than the first one. Similarly, when from $c_1$ there exists a good computation, then from $c'_1$ such a computation exists.*

*Proof.* For the first statement of the lemma we will simulate one transition from $c_1$ by at most one transition from $c'_1$. If $c_1 \xrightarrow{a} c_2$ then directly from Lemma 3.11 it follows that there is $c'_2 \preceq c_2$ such that $c_2 \xrightarrow{a} c'_2$. When $c_1 \xrightarrow{(\texttt{delay},\varepsilon)} c_2$ we have two cases depending on the relation between one before last element of the two configurations. To be more precise, suppose that $c_1 = (\lambda_1 \ldots \lambda_k, \lambda_\infty)$ and $c'_1 = (\lambda'_1 \ldots \lambda'_{k'}, \lambda'_\infty)$. If $\lambda'_{k'} \subseteq \lambda_k$ then we may do $(\texttt{delay}, \epsilon)$ from $c'_1$ and we get $c'_2 \preceq c_2$. Otherwise already $c'_1 \preceq c_2$, we do not do any action and take $c'_2 = c'_1$. Similarly for $(\texttt{delay}, a)$: either we match it with $(\texttt{delay}, a)$ or just with $a$. An obvious induction gives a proof of the first statement.

For the second statement we need to show that the computation from $c'_1$ obtained by matching steps as described above is good (if the one from $c_1$ has been good). This is not immediate as we remove some $(\texttt{delay}, \cdot)$ letters in the matching computation.

Fix a good computation from $c_1$. Let $c_2$ be a configuration in a computation starting from $c_1$, and let $c'_2$ be the corresponding configuration in the matching computation from $c'_1$. To arrive at a contradiction assume that there are no delays after $c'_2$. Let us denote $c'_2 = (\lambda'_1 \ldots \lambda'_{k'}, \lambda'_\infty)$ and $c_2 = (\lambda_1 \ldots \lambda_k, \lambda_\infty)$. Because $c'_2 \preceq c_2$, we know that $\lambda'_{k'}$ is covered by some $\lambda_i$, i.e., $\lambda'_{k'} \subseteq \lambda_i$. Let us take the biggest possible $i$. If some $a$-action is done from $c_2$ then it is matched by an $a$-action from $c'_2$, and for the resulting configurations the inclusion is preserved. This can happen only finitely many times though, as there are infinitely many $(\texttt{delay}, \cdot)$ actions after $c_2$. If a $(\texttt{delay}, \cdot)$ action is done from $c_2$ and $i = k$ then it is matched by a $(\texttt{delay}, \cdot)$ action from $c'_2$, a contradiction with the choice of $c_i$. If $i < k$ then the element $\lambda'_{k'}$ is left on its position in $c'_2$, while in $c_2$ we remove $\lambda_k$, hence $\lambda_i$ covering $\lambda'_{k'}$ gets closer to the end of the sequence. Repeating this argument, we get that the covering $\lambda_i$ finally becomes the last element and the previous case applies. □

**Corollary 3.15.** *The set $\widehat{G}$ is downward closed, so it can be described by the finite set of minimal elements that do not belong to it.* □

As we have mentioned before, there is a good accepting computation from a configuration iff it is possible to reach from it a configuration from $\widehat{G}$ that has only $Q_+$ states. The following lemma says that this property is decidable.

**Lemma 3.16.** *Let $X$ be a downward closed set of configurations of $\mathcal{H}(\mathcal{A})$, represented by the (finite) set of all its minimal elements. It is decidable whether from a given configuration one can reach a configuration in a given set $X$ that has only states from $Q_+$.*

*Proof.* We will use a standard reachability tree argument. The reachability tree is a tree in which the initial configuration is in the root, and every configuration has as children all configurations that may be reached by reading one letter. The algorithm constructs a portion $t$ of the tree according to the following rule: do not add a node $c'$ to $t$ in a situation when among its ancestors there is some $c \preceq c'$. Each path of $t$ is finite because $\preceq$ is a well-quasi-order. Furthermore, since the degree of every node is finite, $t$ is a finite tree. Then we check $t$ for a configuration from $X$ without states from $Q_-$.

We only need to prove that, if in the whole reachability tree there is a configuration as above (which means that $\mathcal{H}(\mathcal{A})$ may accept), then there is also some in $t$. Let $c$ be such a configuration reachable from the initial configuration of $\mathcal{H}(\mathcal{A})$ by a path $\pi$ of the shortest length. Assume that $c$ is not in $t$, i.e. there are two nodes on $\pi$, say $c_1$ and $c_2$, such that $c_1$ is an ancestor of $c_2$ and $c_1 \preceq c_2$ (i.e. $c_2$ was not added to $t$). Then from Lemma 3.14, there exists $c' \preceq c$ that may be reached from $c_1$ and the path from $c_1$ to $c'$ will be not longer than that from $c_2$ to $c$. So the path leading to $c'$ from the initial configuration is strictly shorter than $\pi$. Moreover, as $c' \preceq c$ and $X$ is downward closed, we immediately deduce that $c' \in X$, and $c'$ does not contain states from $Q_-$ which is a contradiction. □

3.4. **Computing $\widehat{G}$.** In this subsection we deal with the main technical problem of the proof that is computing the set $\widehat{G}$ of all configurations from which there exist a good computation. We will actually compute the complement of $\widehat{G}$. While we will use well-orderings in the proof, standard termination arguments do not work in this case. We will need to examine more closely the definition of $\mathcal{H}(\mathcal{A})$ an in particular the mechanics of its transition as described in Lemmas 3.10, 3.11, and 3.12.

We write $X{\uparrow}$ for an upward closure of set $X$,

$$X{\uparrow} = \{c : \exists_{c' \in X} c' \preceq c\}.$$

Observe that by Corollary 3.15 the complement of $\widehat{G}$ is upward closed.

Let set $pre^{\forall}_{\texttt{delay}}$ (respectively $pre^{\forall}_{\Sigma^*}$) contain all configurations, from which after reading any letter $(\texttt{delay}, \cdot)$ (any number of letters from $\Sigma$), we have to reach a configuration from $X$,

$$pre^{\forall}_{\texttt{delay}}(X) = \{c : \forall_{c'}(c \overset{(\texttt{delay},\cdot)}{\longrightarrow} c' \Rightarrow c' \in X)\},$$
$$pre^{\forall}_{\Sigma^*}(X) = \{c : \forall_{c'}(c \overset{\Sigma^*}{\twoheadrightarrow} c' \Rightarrow c' \in X)\}.$$

Now we can use these $pre$ operations to compute a sequence of sets of configurations

$$Z_{-1} = \emptyset, \qquad Z_i = pre^{\forall}_{\Sigma^*}(pre^{\forall}_{\texttt{delay}}(Z_{i-1}{\uparrow})).$$

It is important that we may effectively represent and compare all the sets $Z_i{\uparrow}$. Because the relation $\preceq$ is a well-quasi-order, any upward closed set $X{\uparrow}$ may be represented by finitely many elements $c_1, \ldots, c_k$ (called *generators*) such that $X{\uparrow} = \{c_1, \ldots, c_k\}{\uparrow}$. Moreover, an easy induction shows that $Z_{i-1}{\uparrow} \subseteq Z_i{\uparrow}$ for every $i$ (because both $pre^{\forall}$ operations preserve inclusion). Once again, because relation $\preceq$ is a well-quasi-order, there has to be $i$ such that $Z_{i-1}{\uparrow} = Z_i{\uparrow}$. Let us write $Z_\infty$ for this $Z_i$.

First, we show that $Z_\infty$ is indeed the complement of $\widehat{G}$.

**Lemma 3.17.** *There is a good computation from a configuration $c$ iff $c \notin Z_\infty\uparrow$.*

*Proof.* ($\Rightarrow$) We show by induction that $c \notin Z_i$ for $i = -1, 0, 1 \dots$. For $i = -1$ it is obvious. Assume for contradiction that there exists a good computation from $c$, but $c \in Z_i\uparrow$. Then there exists $c' \preceq c$ with $c' \in Z_i$. From Lemma 3.14 we know that a good infinite computation exists also from $c'$. This computation may first read some letters from $\Sigma$, but finally it has to read a letter $(\texttt{delay}, \cdot)$, that results in a configuration $c_2$. Definition of $Z_i$ tells us that $c_2 \in Z_{i-1}\uparrow$. But from $c_2$ there is also a good infinite computation, a contradiction.

($\Leftarrow$) Assume that every computation (finite or infinite) from $c$ reads at most $k$ letters $(\texttt{delay}, \cdot)$. An easy induction on $k$ shows that $c \in Z_k$. □

To compute $Z_\infty$ it is enough to show how to compute $Z_i\uparrow$ from $Z_{i-1}\uparrow$. This is the most difficult part of the proof that will occupy the rest of the subsection. Once this is done we will calculate all the sets $Z_i\uparrow$, starting with $Z_{-1} = \emptyset$ and ending when $Z_{i-1}\uparrow = Z_i\uparrow$.

The main idea in calculating $pre^\forall_{\Sigma^*}(pre^\forall_{\texttt{delay}}(X))$ is that the length of its generators may be bounded by some function in the length of generators of $X$. This is expressed by the following lemma.

**Lemma 3.18.** *Given an upward closed set $X$ we can compute a constant $D(X)$ (which depends also on our fixed automaton $\mathcal{A}$) such that the size of every minimal element of $pre^\forall_{\Sigma^*}(pre^\forall_{\texttt{delay}}(X))$ is bounded by $D(X)$*

Once we know the bound on the size of generators, we can try all potential candidates. The following lemma shows that it is possible.

**Lemma 3.19.** *For every upper-closed set $X$, the membership in $pre^\forall_{\Sigma^*}(pre^\forall_{\texttt{delay}}(X))$ is decidable.*

Together Lemmas 3.18 and 3.19 allow us to compute the sequence $Z_0, Z_1, \dots, Z_\infty$ and hence also $\widehat{G}$.

To finish the proof of the theorem, it remains to give proofs of the two lemmas. The first is substantially more complicated, and will occupy most of the space, while the second we will get as a rather simple corollary. In the first proof, we will calculate separately bounds for $pre^\forall_{\texttt{delay}}(X)$ and for $pre^\forall_{\Sigma^*}(X)$. In the sequel we will need to use some special representation for sets of configurations.

**Definition 3.20.** A *compressed configuration* has a form

$$\widehat{c} = (\lambda_1 \dots \lambda_l, f, \lambda_\infty),$$

where $\lambda_i \in \Lambda_I$, $\lambda_\infty \in \Lambda_\infty$ and $f : \Lambda_I \to \mathcal{P}(\Lambda_I)$ (values of $f$ are subsets of $\Lambda_I$).

On compressed configurations we introduce an expansion operation parametrized by words from $\Lambda_I^*$.

**Definition 3.21.** A compressed configuration $\widehat{c} = (\lambda_1 \dots \lambda_l, f, \lambda_\infty)$ may be expanded in a context of some word $\lambda_1^0 \dots \lambda_k^0 \in \Lambda_I^*$, giving as a result the set of configurations $(\lambda_1 \dots \lambda_l \lambda'_{l+1} \dots \lambda'_{l+k}, \lambda_\infty)$ such that $\lambda'_{l+i} \in f(\lambda_i^0)$ for $1 \le i \le k$. We will use $exp(\widehat{c}, \lambda_1^0 \dots \lambda_k^0)$ to denote the set of obtained configurations. Similarly, if $\widehat{C}$ is a set of compressed configurations we write $Exp(\widehat{C}, \lambda_1^0 \dots \lambda_k^0)$ for $\bigcup\{exp(\widehat{c}, \lambda_1^0 \dots \lambda_k^0) : \widehat{c} \in \widehat{C}\}$.

Observe that the value $f(\lambda)$ for $\lambda$ not appearing in $\lambda_1^0 \dots \lambda_k^0$ does not matter; moreover if some $f(\lambda_i^0) = \emptyset$ then the result of expanding is the empty set.

We use compressed configurations, because the set of successors of a configuration may be described by a bounded number of compressed configurations. This is not true for ordinary configurations due to nondeterminism. For example, when there is more than one choice of a transition on action $a$ form a letter $\lambda$ then every occurrence of $\lambda$ in a configuration may make a choice independently, so the number of successor configurations grows with the number of occurrences of $\lambda$ in a configuration.

Let us see how to calculate $pre^{\forall}_{\texttt{delay}}(X)$. Some care is needed as this set is not upward closed with respect to the $\preceq$ relation. This is because the a $(\texttt{delay}, \cdot)$ action treats the one before the last element of a configuration in a special way. So if something is inserted after $\lambda_k$ in $(\lambda_1 \dots \lambda_k, \lambda_\infty)$ then the `delay` operation uses this inserted element instead of $\lambda_k$. As a side remark let us mention that using the upward closure of $pre^{\forall}_{\texttt{delay}}(Z_{i-1}\uparrow)$ in the definition of $Z_i$ would be incorrect (Lemma 3.17 would not be true).

To remedy this problem we use a refined relation $\preceq_r$. Given two configurations $c' = (\lambda'_1 \dots \lambda'_{k'}, \lambda'_\infty)$ and $c = (\lambda_1 \dots \lambda_k, \lambda_\infty)$ we set

$$c' \preceq_r c \quad \text{iff} \quad k' > 0,\ c' \preceq c \text{ and } \lambda'_{k'} \subseteq \lambda_k$$

Note that the set $pre^{\forall}_{\texttt{delay}}(X)$ is upward closed with respect to relation $\preceq_r$, when $X$ is upward closed with respect to $\preceq$. This is because if $c'_1 \preceq_r c_1$ and $c_1 \xrightarrow{(\texttt{delay},\cdot)} c_2$ then also $c'_1 \xrightarrow{(\texttt{delay},\cdot)} c'_2$ with some $c'_2 \preceq c_2$. Hence, if $c_1 \notin pre^{\forall}_{\texttt{delay}}(X)$ then $c'_1 \notin pre^{\forall}_{\texttt{delay}}(X)$.

The following lemma tells us that successors of a configuration may be described using compressed configurations and that there are not too many of them.

**Lemma 3.22.** *For every configuration $c_0 = (\lambda_1 \dots \lambda_k, \lambda_\infty)$, $k > 0$ there exists a finite set of compressed configurations $\widehat{C}(\lambda_k, \lambda_\infty)$ (depending only on $\lambda_k$ and $\lambda_\infty$) such that:*

- *if $c_0 \xrightarrow{(\texttt{delay},\cdot)} c$ then $c \in Exp(\widehat{C}(\lambda_k, \lambda_\infty), \lambda_1 \dots \lambda_{k-1})$;*
- *if $c \in Exp(\widehat{C}(\lambda_k, \lambda_\infty), \lambda_1 \dots \lambda_{k-1})$ then $c_0 \xrightarrow{(\texttt{delay},\cdot)} c'$ for some $c' \preceq c$.*

*Proof.* The transition on $(\texttt{delay}, \epsilon)$ is deterministic. If $c_0 \xrightarrow{(\texttt{delay},\epsilon)} c'$ then we either have $c' = (\lambda'_k \lambda_1 \dots \lambda_{k-1}, \lambda'_\infty)$ or $c' = (\lambda_1 \dots \lambda_{k-1}, \lambda'_\infty)$ depending on $\lambda_k$. In the first case we add $\widehat{c} = (\lambda'_k, \mathbf{sgl}, \lambda'_\infty)$ to $\widehat{C}(\lambda_k, \lambda_\infty)$, in the second case $\widehat{c} = (\epsilon, \mathbf{sgl}, \lambda'_\infty)$, where $\mathbf{sgl}(\lambda) = \{\lambda\}$. In both cases $exp(\widehat{c}, \lambda_1 \dots \lambda_{k-1}) = \{c'\}$.

Now consider transitions reading $(\texttt{delay}, a)$. A result of this transition is not unique and depends on the choice of a transition for each element of the configuration. We fix a set $\mathcal{T}$ of transitions $\lambda \xrightarrow{a}_{\mathcal{A}} (\lambda', \gamma')$; intuitively these are allowed transitions from $\lambda_1, \dots, \lambda_{k-1}$. We also fix transitions $\lambda^1_k \xrightarrow{a}_{\mathcal{A}} (\lambda'_k, \gamma'_k)$ and $\lambda_\infty \xrightarrow{a}_{\mathcal{A}} (\lambda'_\infty, \gamma'_\infty)$ (where $\lambda^1_k$ is $\lambda_k$ with increased regions as in Lemma 3.12). This choice of transitions gives us a compressed configuration $\widehat{c} = (\gamma, f, \lambda''_\infty)$, where

$$\begin{aligned}
\gamma = {} & \gamma'_k \cup \gamma'_\infty \cup \{(q, \mathcal{I}_{d+1}) : (q, \{d\}) \in \lambda'_k, d < d_{max}\} \\
& \cup \bigcup \{\gamma' : (\lambda \xrightarrow{a}_{\mathcal{A}} (\lambda', \gamma')) \in \mathcal{T}, \lambda \in \Lambda_I\}, \\
f(\lambda) = {} & \bigcup \{\lambda' : (\lambda \xrightarrow{a}_{\mathcal{A}} (\lambda', \gamma')) \in \mathcal{T}\}, \\
\lambda''_\infty = {} & \lambda'_\infty \cup \{(q, \mathcal{I}_\infty) : (q, \{d_{max}\}) \in \lambda'_k\}.
\end{aligned}$$

We add $\widehat{c}$ into $\widehat{C}(\lambda_k, \lambda_\infty)$.

We now show that the constructed $\widehat{C}(\lambda_k, \lambda_\infty)$ has the required properties. Consider a successor $c$ of $c_0$ that is reached using the transitions we have fixed. In particular, we require that each transition from $\mathcal{T}$ is used at least once. Take $\widehat{c}$ as calculated above. Directly from the definition we get $c \in exp(\widehat{c}, \lambda_1 \dots \lambda_{k-1})$. As the choice of transitions was arbitrary, this gives the first statement of the lemma.

Now consider $c = (\gamma\lambda'_1 \dots \lambda'_{k-1}, \lambda''_\infty) \in exp(\widehat{c}, \lambda_1 \dots \lambda_{k-1})$ where $\widehat{c} = (\gamma, f, \lambda''_\infty)$ is obtained by a choice of some $\mathcal{T}$ and some transitions from $\lambda_k^1$ and $\lambda_\infty$. For every $i$ let us choose some transition $\lambda_i \xrightarrow{a}_{\mathcal{A}} (\lambda'_i, \gamma'_i)$ from $\mathcal{T}$ (there is at least one such transition in $\mathcal{T}$ because $\lambda'_i \in f(\lambda_i)$). Take $c' = (\gamma'\lambda'_1 \dots \lambda'_{k-1}, \lambda''_\infty)$ where

$$\gamma' = \gamma'_k \cup \gamma'_\infty \cup \{(q, \mathcal{I}_{d+1}) : (q, \{d\}) \in \lambda'_k, d < d_{max}\} \cup \bigcup_{1 \le i \le k-1} \gamma'_i$$

Then $\gamma' \subseteq \gamma$ so $c' \preceq c$. It is easy to check that there is a transition $c_0 \xrightarrow{(\texttt{delay},a)} c'$. □

We need to find all minimal elements of $pre^{\forall}_{\texttt{delay}}(Z_{i-1}\uparrow)$. The following lemma will allow us to get a bound on their size.

**Lemma 3.23.** *For a given $\widehat{C}$ and a set $X$ upward closed with respect to the $\preceq_r$ relation there exists a constant $B(X, \widehat{C})$ (and we may compute it) such that if for some $\lambda_1^0 \dots \lambda_k^0$*

$$Exp(\widehat{C}, \lambda_1^0 \dots \lambda_k^0) \subseteq X$$

*then there exist $1 \le i_1 < \dots < i_m \le k$, $m < B(X, \widehat{C})$ with*

$$Exp(\widehat{C}, \lambda_{i_1}^0 \dots \lambda_{i_m}^0) \subseteq X.$$

*Proof.* First suppose that $\widehat{C}$ is a singleton $\{\widehat{c}\}$; where $\widehat{c} = (\lambda_1 \dots \lambda_l, f, \lambda_\infty)$. We describe a construction of a finite automaton $\mathcal{A}^X_{\widehat{c}}$ accepting the language

$$L^X_{\widehat{c}} = \{\lambda'_1 \dots \lambda'_k : exp(\widehat{c}, \lambda'_1 \dots \lambda'_k) \subseteq X\}.$$

Recall that $X$ is an upward closed set with respect to $\preceq_r$ relation. This implies that $L^X_{\widehat{c}}$ is upward closed with respect to the standard subsequence relation $\sqsubseteq$. It is easy to check that for every letter $\lambda \in \Lambda_I$, if $L \subseteq \Lambda_I^*$ is $\sqsubseteq$-upward closed then the quotient $L/\lambda$ is also $\sqsubseteq$-upward closed. Moreover $L \subseteq L/\lambda$, as if $w \in L$ then $aw \in L$ that implies $w \in L/\lambda$. Because $\sqsubseteq$ is a well-quasi-order, this last property implies that the set of all possible quotients of $L^X_{\widehat{c}}$, i.e. the languages $L^X_{\widehat{c}}/w$ for $w \in \Lambda_I^*$, is finite. These quotients are the states of $\mathcal{A}^X_{\widehat{c}}$ we were looking for. Indeed $\mathcal{A}^X_{\widehat{c}}$ is the minimal deterministic automaton for $L^X_{\widehat{c}}$. Take $B(X, \{\widehat{c}\})$ to be the size of the automaton. From the pumping lemma it follows that if the word $\lambda_1^0 \dots \lambda_k^0$ is accepted by $\mathcal{A}^X_{\widehat{c}}$ then there is a subsequence of length $\le B(X, \{\widehat{c}\})$ accepted by $\mathcal{A}^X_{\widehat{c}}$.

Now consider a general situation. For every $\widehat{c} \in \widehat{C}$ from above we have some subsequence $\lambda_{i_1}^0 \dots \lambda_{i_m}^0$ of length $m \le B(X, \{\widehat{c}\})$, such that $exp(\widehat{c}, \lambda_{i_1}^0 \dots \lambda_{i_m}^0) \subseteq X$. We take all the elements from all these subsequences, getting a subsequence of length $\le B(X, \widehat{C}) := \sum_{\widehat{c} \in \widehat{C}} B(X, \{\widehat{c}\})$ such that all the inclusions hold. □

The above two lemmas allow to compute a bound on the size of minimal elements in $pre^{\forall}_{\texttt{delay}}(Z_{i-1}\uparrow)$.

**Lemma 3.24.** *There is an algorithm that given $X\uparrow$ computes a constant $M_{delay}(X\uparrow)$ such that the size of every minimal element of $pre^{\forall}_{\texttt{delay}}(X\uparrow)$ is bounded by $M_{delay}(X\uparrow)$.*

*Proof.* There are only finitely many different $\widehat{C}(\lambda_k, \lambda_\infty)$ as constructed in Lemma 3.22. Let $M_{delay}$ be the maximal possible value of $B(X\uparrow, \widehat{C}(\lambda_k, \lambda_\infty))$.

Suppose $c_0 = (\lambda^0_1 \dots \lambda^0_k, \lambda^0_\infty)$ is a minimal element of $pre^{\forall}_{\texttt{delay}}(X\uparrow)$. Take the set $\widehat{C}(\lambda^0_k, \lambda^0_\infty)$ as given by Lemma 3.22. We have that $Exp(\widehat{C}(\lambda^0_k, \lambda^0_\infty), \lambda^0_1 \dots \lambda^0_{k-1}) \subseteq X\uparrow$ by the second statement of this lemma. From Lemma 3.23 we get a subsequence $\lambda'_1 \dots \lambda'_l$ of $\lambda^0_1 \dots \lambda^0_{k-1}$ whose length is bounded by $B(X\uparrow, \widehat{C}(\lambda^0_k, \lambda^0_\infty)) \leq M_{delay}$ and such that $Exp(\widehat{C}(\lambda^0_k, \lambda^0_\infty), \lambda'_1 \dots \lambda'_l) \subseteq X\uparrow$. By the first statement of Lemma 3.22 we get that $(\lambda'_1 \dots \lambda'_l \lambda^0_k, \lambda^0_\infty) \in pre^{\forall}_{\texttt{delay}}(X\uparrow)$. By the minimality of $c_0$, we get that $c_0 = (\lambda'_1 \dots \lambda'_l \lambda^0_k, \lambda^0_\infty)$, so its length is bounded by $M_{delay} + 2$. □

Now we describe how to calculate $pre^{\forall}_{\Sigma^*}(Y)\uparrow$ for any set $Y$ upward closed with respect to $\preceq_r$ relation. The first lemma says that we may represent successors using compressed configurations.

**Lemma 3.25.** *For every compressed configuration $\widehat{c}_0$ there is a set of compressed configurations $\widehat{C}(\widehat{c}_0)$ (and we may compute it) such that for every $\lambda^0_1 \dots \lambda^0_k$*

- *if $c_0 \in exp(\widehat{c}_0, \lambda^0_1 \dots \lambda^0_k)$ and $c_0 \xrightarrow{a} c$ for some $a \in \Sigma$, then $c \in Exp(\widehat{C}(\widehat{c}_0), \lambda^0_1 \dots \lambda^0_k)$;*
- *if $c \in Exp(\widehat{C}(\widehat{c}_0), \lambda^0_1 \dots \lambda^0_k)$, then $c_0 \xrightarrow{a} c'$ for some $c' \preceq_r c$, $a \in \Sigma$ and some $c_0 \in exp(\widehat{c}_0, \lambda^0_1 \dots \lambda^0_k)$.*

*Proof.* Let $\widehat{c}_0 = (\lambda_1 \dots \lambda_l, f, \lambda_\infty)$. Fix a letter $a \in \Sigma$. We fix a set $\mathcal{T}$ of transitions $\lambda \xrightarrow{a}_{\mathcal{A}} (\lambda', \gamma')$; intuitively these are allowed transitions from $\lambda \in f(\lambda^0_i)$. We also fix transitions $\lambda_i \xrightarrow{a}_{\mathcal{A}} (\lambda'_i, \gamma'_i)$ for $i = 1, \dots, l, \infty$. This choice of transitions gives us a compressed configuration $\widehat{c} = (\gamma \lambda'_1 \dots \lambda'_l, f', \lambda'_\infty)$, where

$$\gamma = \bigcup_{i=1\dots,l,\infty} \gamma'_i \cup \bigcup \{\gamma' : (\lambda \xrightarrow{a}_{\mathcal{A}} (\lambda', \gamma')) \in \mathcal{T}, \lambda \in \Lambda_I\},$$

$$f'(\lambda^0) = \{\lambda' : (\lambda \xrightarrow{a}_{\mathcal{A}} (\lambda', \gamma')) \in \mathcal{T}, \lambda \in f(\lambda^0)\}.$$

We add $\widehat{c}$ into $\widehat{C}(\widehat{c}_0)$.

For the first statement of the lemma, take $c_0 \in exp(\widehat{c}_0, \lambda^0_1 \dots \lambda^0_k)$ and consider any successor $c$ of $c_0$ that is reached using the transitions we have fixed. In particular we require that each transition from $\mathcal{T}$ is used at least once. Take $\widehat{c}$ as calculated above. Then directly from the definition we get $c \in exp(\widehat{c}, \lambda^0_1 \dots \lambda^0_k)$. As the choice of transitions was arbitrary this gives the first statement of the lemma.

Now consider some $\widehat{c} \in \widehat{C}(\widehat{c}_0)$. It is of the form $(\gamma \lambda'_1 \dots \lambda'_l, f', \lambda'_\infty)$. According to the above, it was constructed from $\widehat{c}_0$ using some transitions $\lambda_i \xrightarrow{a}_{\mathcal{A}} (\lambda'_i, \gamma'_i)$ for $i = 1, \dots, l, \infty$ and some set of transitions $\mathcal{T}$. Take $c \in exp(\widehat{c}, \lambda^0_1 \dots \lambda^0_k)$. We have that $c$ is of the form $(\gamma' \lambda'_1 \dots \lambda'_l \lambda'_{l+1} \dots \lambda'_{l+k}, \lambda'_\infty)$ where $\lambda'_1 \dots \lambda'_l$ are as in $\widehat{c}$ and for $i = 1, \dots, k$ we can choose from $\mathcal{T}$ transitions $\lambda_{l+i} \xrightarrow{a}_{\mathcal{A}} (\lambda'_{l+i}, \gamma'_{l+i})$ such that $\lambda_{l+i} \in f(\lambda^0_i)$. Take $c_0 = (\lambda_1 \dots \lambda_l \lambda_{l+1}, \dots, \lambda_{l+k}, \lambda_\infty)$, i.e. a configuration whose components are predecessors of transitions we have selected. We have $c_0 \in exp(\widehat{c}_0, \lambda^0_1 \dots \lambda^0_k)$ by the definition of expansion.

Let $c' = (\gamma' \lambda'_1 \dots \lambda'_{l+k}, \lambda'_\infty)$ with $\gamma' = \bigcup_{i=1,\dots,l+k,\infty} \gamma'_i$. Observe that $\gamma'$ may be a proper subset of $\gamma$ if not all transitions from $\mathcal{T}$ have been used. Then $c' \preceq_r c$ and there is a transition $c_0 \xrightarrow{a} c'$.

□

The following lemma says that we may list a big enough portion of all configurations reachable from some $c_0$ (similarly like in step two of the decision procedure, Lemma 3.16) and moreover that size of this portion is bounded by a constant.

**Lemma 3.26.** *For every $\lambda_\infty \in \Lambda_\infty$ we can construct a set $\widehat{C}_{\Sigma^*}(\lambda_\infty)$ such that for every $\lambda_1 \dots \lambda_k \in \Lambda_I^*$*

- *if $(\lambda_1 \dots \lambda_k, \lambda_\infty) \overset{\Sigma^*}{\twoheadrightarrow} c$ for some $c$ then there is $c' \preceq_r c$ such that $c' \in Exp(\widehat{C}_{\Sigma^*}(\lambda_\infty), \lambda_1 \dots \lambda_k)$;*
- *if $c \in Exp(\widehat{C}_{\Sigma^*}(\lambda_\infty), \lambda_1 \dots \lambda_k)$ then there is $c' \preceq_r c$ with $(\lambda_1 \dots \lambda_k, \lambda_\infty) \overset{\Sigma^*}{\twoheadrightarrow} c'$.*

*Proof.* Take the compressed configuration $\widehat{c}_0 = (\epsilon, \mathbf{sgl}, \lambda_\infty)$, where, as before, $\mathbf{sgl}(\lambda) = \{\lambda\}$. We define a set $\widehat{C}$ of compressed configurations as a closure of $\{\widehat{c}_0\}$ on the operation defined in Lemma 3.25. This set may be infinite but we do not worry about it for the moment. We show first that it satisfies the requirements of the lemma.

Take some $\lambda_1 \dots \lambda_k \in \Lambda_I^*$ and $c$ such that $(\lambda_1 \dots \lambda_k, \lambda_\infty) \overset{\Sigma^*}{\twoheadrightarrow} c$. We need to show that we can find an extended configuration $\widehat{c} \in \widehat{C}$ such that $c \in exp(\widehat{c}, \lambda_1 \dots \lambda_k)$. The proof is by easy induction on the number of transitions. For the base step we have $(\lambda_1 \dots \lambda_k, \lambda_\infty) \in exp(\widehat{c}_0, \lambda_1 \dots \lambda_k)$, and the induction step is given by the first statement of Lemma 3.25.

Now, suppose that $\widehat{c} \in \widehat{C}$ and $c \in exp(\widehat{c}, \lambda_1 \dots \lambda_k)$. An induction using the second statement of Lemma 3.25 shows that there is $c' \preceq_r c$ such that $(\lambda_1 \dots \lambda_k, \lambda_\infty) \overset{\Sigma^*}{\twoheadrightarrow} c'$.

In order to reduce $\widehat{C}$ to a finite set we once again use well-quasi-orders. We define a relation $\sqsubseteq$ on compressed configurations:

$$(\lambda'_1 \dots \lambda'_{l'}, f', \lambda'_\infty) \sqsubseteq (\lambda_1 \dots \lambda_l, f, \lambda_\infty) \iff (\lambda'_1 \dots \lambda'_{l'}, \lambda'_\infty) \preceq_r (\lambda_1 \dots \lambda_l, \lambda_\infty) \text{ and } f = f'.$$

This relation is a well-quasi-order. We take $\widehat{C}_{\Sigma^*}(\lambda_\infty)$ to be the set of minimal elements in this quasi-order. It is clear that $Exp(\widehat{C}_{\Sigma^*}(\lambda_\infty), \lambda_1 \dots \lambda_k) \subseteq Exp(\widehat{C}, \lambda_1 \dots \lambda_k)$ for arbitrary $\lambda_1 \dots \lambda_k$. So, by the above observations the second property of the lemma holds. For the first property observe that whenever $\widehat{c}' \sqsubseteq \widehat{c}$ and $c \in exp(\widehat{c}, \lambda_1 \dots \lambda_k)$ then there is $c' \in exp(\widehat{c}', \lambda_1 \dots \lambda_k)$ with $c' \preceq_r c$. □

**Lemma 3.27.** *There is an algorithm that given a set $Y$ upward closed with respect to the $\preceq_r$ relation computes a constant $M_{\Sigma^*}(Y)$ such that the size of every minimal element of $pre^{\forall}_{\Sigma^*}(Y)$ is bounded by $M_{\Sigma^*}(Y)$.*

*Proof.* There are only finitely many different $\widehat{C}(\lambda_\infty)$ constructed in the above lemma. Let $M_{\Sigma^*}$ be the maximal possible value of $B(Y, \widehat{C}(\lambda_\infty))$ (cf. Lemma 3.23)

Suppose $c_0 = (\lambda_1^0 \dots \lambda_k^0, \lambda_\infty^0)$ is a minimal element of $pre^{\forall}_{\Sigma^*}(Y)$. Take the set $\widehat{C}(\lambda_\infty^0)$ as given by Lemma 3.26. We have that $Exp(\widehat{C}_{\Sigma^*}(\lambda_\infty^0), \lambda_1^0 \dots \lambda_k^0) \subseteq Y$ by the second statement of this lemma. From Lemma 3.23 we get a subsequence $\lambda'_1 \dots \lambda'_l$ of $\lambda_1^0 \dots \lambda_k^0$ whose length is bounded by $B(X, \widehat{C}(\lambda_\infty^0)) \leq M_{\Sigma^*}$ and such that $Exp(\widehat{C}_{\Sigma^*}(\lambda_\infty^0), \lambda'_1 \dots \lambda'_l) \subseteq Y$. By the first

statement of Lemma 3.26 we get that $(\lambda'_1 \dots \lambda'_l, \lambda^0_\infty) \in pre^\forall_{\Sigma^*}(Y)$. By the minimality of $c_0$, we have that $c_0 = (\lambda'_1 \dots \lambda'_l, \lambda^0_\infty)$, so its length is bounded by $M_{\Sigma^*} + 1$. □

The last step before proving Lemmas 3.18 and 3.19 consists of two simple observations.

**Lemma 3.28.** *For every set $X$ upward closed with respect to $\preceq$ relation, the membership in $Y = pre^\forall_{\texttt{delay}}(X)$ is decidable. Moreover $Y$ is a $\preceq_r$-upward closed set.*

*Proof.* The first part of the lemma is obvious, it suffices to test all possible transitions that are explicitly characterized in Lemmas 3.10 and 3.12. The second part follows from the property that we have already noticed before (page 13): if $c'_1 \preceq_r c_1$ and $c_1 \xrightarrow{(\texttt{delay},\cdot)} c_2$ then also $c'_1 \xrightarrow{(\texttt{delay},\cdot)} c'_2$ with some $c'_2 \preceq c_2$. □

**Lemma 3.29.** *For every set $Y$ upward closed with respect of $\preceq_r$ relation, the membership in $pre^\forall_{\Sigma^*}(Y)$ is decidable.*

*Proof.* Given a configuration $c$ we need to decide if $c \in pre^\forall_{\Sigma^*}(Y)$. We apply successively $\xrightarrow{a}$ transitions to $c$ constructing a part of the reachability tree. We stop the development in a node if it has an ancestor smaller with respect to $\preceq_r$-relation. As $\preceq_r$ is a well-quasi-order, and the branching at each node is finite, we get a finite tree $t$.

It remains to argue that this construction is correct. If in the above process we find a configuration that is not in $Y$ then clearly $c$ is not in $pre^\forall_{\Sigma^*}(Y)$. For the other direction, assume conversely that there is $c' \notin Y$ with $c \xrightarrow{\Sigma^*} c'$. Choose $c' \notin Y$ so that the length of a derivation $c \xrightarrow{\Sigma^*} c'$ is the smallest possible. We show that $c' \in t$. Recall that Lemma 3.11 characterizes transitions on letters. Directly from this characterization we obtain that if $c'_1 \preceq_r c_1$ and $c_1 \xrightarrow{a} c_2$ then also $c'_1 \xrightarrow{a} c'_2$ with some $c'_2 \preceq_r c_2$. Using this fact, we get that if $c'$ is not in $t$ then there is $d' \preceq c'$ such that the derivation $c \xrightarrow{\Sigma^*} d'$ is shorter than $c \xrightarrow{\Sigma^*} c'$. This is impossible by the choice of $c'$. □

**Proof** (of Lemma 3.18)
Take an upward closed set $X$. By Lemma 3.24 we can compute a constant $M_{\texttt{delay}}$ that bounds the size of minimal elements in $Y = pre^\forall_{\texttt{delay}}(X)$. Using Lemma 3.28 we can find the minimal elements of $Y$ by enumerating all configurations of size bounded by $M_{\texttt{delay}}$. Observe that $Y$ is $\preceq_r$ upward closed.

Once we have computed $Y$, Lemma 3.27 gives us a constant $M_{\Sigma^*}(Y)$ bounding the size of minimal elements in $pre^\forall_{\Sigma^*}(Y) = pre^\forall_{\Sigma^*}(pre^\forall_{\texttt{delay}}(X))$.

□

**Proof** (of Lemma 3.19)
We first compute the set $Y = pre^\forall_{\texttt{delay}}(X)$ as described above. We can then use Lemma 3.29 to test for the membership in $pre^\forall_{\Sigma^*}(Y) = pre^\forall_{\Sigma^*}(pre^\forall_{\texttt{delay}}(X))$.

□

## 4. Constrained TPTL

In this section we present a fragment of TPTL (timed propositional temporal logic) that can be translated to automata whose emptiness problem is decidable by Theorem 3.1. We

compare this fragment with other known logics for real time. We will be rather brief in presentations of different formalisms, and refer the reader to recent surveys [9, 26].

TPTL[8] is a timed extension of linear time temporal logic that allows to explicitly set and compare clock variables. We will consider the logic with only one clock variable that we denote TPTL$^1$. The syntax of the logic is:

$$p \mid x.\alpha \mid x \sim c \mid \alpha \wedge \beta \mid \alpha \vee \beta \mid \alpha \mathbb{U} \beta \mid \alpha \widetilde{\mathbb{U}} \beta,$$

where $p$ ranges over action letters, $x$ is the unique clock variable, and $x \sim c$ is a comparison of $x$ with a constant with $\sim$ being one of $=, \neq, <, \leq, >, \geq$. We do not have negation in the syntax, but from the semantics it will be clear that negation is definable.

The logic is evaluated over timed sequences $w = (a_1, t_1)(a_2, t_2) \dots$ We define a satisfaction relation $w, i, v \vDash \alpha$ saying that a formula $\alpha$ is true at a position $i$ of a timed word $w$ with a valuation $v$ of the unique clock variable:

$w, i, v \vDash p$ if $a_i = p$,
$w, i, v \vDash x \sim c$ if $t_i - v \sim c$,
$w, i, v \vDash x.\alpha$ if $w, i, t_i \vDash \alpha$,
$w, i, v \vDash \alpha \mathbb{U} \beta$ if $\exists_{j>i}$ $(w, j, v \vDash \beta$ and $\forall_{k \in (i,j)}$ $w, k, v \vDash \alpha)$,
$w, i, v \vDash \alpha \widetilde{\mathbb{U}} \beta$ if $\forall_{j>i}$ $(w, j, v \vDash \beta$ or $\exists_{k \in (i,j)} w, k, v \vDash \alpha)$.

As usual, "until" operators permit us to introduce "sometimes" and "always" operators:

$$F\alpha \equiv tt \mathbb{U} \alpha, \qquad G\alpha \equiv ff \widetilde{\mathbb{U}} \alpha.$$

For the following it will be interesting to note that the two "until" operators are interdefinable once we have "always" and "sometimes" operators:

$$\alpha \widetilde{\mathbb{U}} \beta \equiv G\beta \vee \beta \mathbb{U} \alpha, \qquad \alpha \mathbb{U} \beta \equiv F\beta \wedge \beta \widetilde{\mathbb{U}} \alpha.$$

Observe that TPTL$^1$ subsumes metric temporal logic (MTL). For example: $\alpha \mathbb{U}_{(0,j)} \beta$ of MTL is equivalent to $x.(\alpha \mathbb{U}((x < j) \wedge \beta))$. We will not present MTL here, but rather refer the reader to [10] where it is also shown that the following TPTL$^1$ formula is not expressible in MTL (when considered in the pointwise semantics):

$$x.(F(b \wedge F(c \wedge x \leq 2))). \tag{4.1}$$

The satisfiability problem over infinite timed sequences is undecidable for MTL [22], hence also for TPTL$^1$. Using our decidability result for alternating timed automata, we can nevertheless find a decidable fragment that we call Constrained TPTL. The definition of this fragment will use an auxiliary notion of positive TPTL$^1$ formulas. These formulas can be translated into alternating automata where all states are accepting. The set of *positive formulas* is given by the following grammar:

$$p \mid x.\varphi \mid x \sim c \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \varphi \widetilde{\mathbb{U}} \psi \mid F((x \leq c) \wedge \psi)$$

The set of formulas of *Constrained TPTL* is:

$$p \mid x.\alpha \mid x \sim c \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \alpha \mathbb{U} \beta \mid \varphi \qquad \varphi \text{ positive.}$$

Observe that the formula (4.1) belongs to the positive fragment if we add redundant $(x \leq 2)$ after $b$.

**Theorem 4.1.** *For a given Constrained TPTL formula $\alpha$ it is decidable whether there is a non Zeno timed word that is a model of $\alpha$. The complexity of the problem cannot be bounded by a primitive recursive function.*

*Proof.* It is enough to give a translation from formulas to automata in the class from Theorem 3.1. The translation is on the syntax of the formula.

We start with the automaton for positive formulas. The set of states of an automaton for a formula will consists of all subformulas of the formula. A state associated to a formula $\alpha$ will be denoted by $[\alpha]$. The intended semantics is that a timed word $w$ is accepted from $[\alpha]$ iff $w, 1, 0 \vDash \alpha$.

The transition relation of the automaton is given in the following table.

$$[p] \xrightarrow{p} \top \qquad [x \sim c] \xrightarrow[x \sim c]{*} \top$$

$$[\alpha \vee \beta] \xrightarrow{\varepsilon} [\alpha] \vee [\beta] \qquad [\alpha \wedge \beta] \xrightarrow{\varepsilon} [\alpha] \wedge [\beta]$$

$$[x.\alpha] \xrightarrow[x:=0]{\varepsilon} [\alpha]$$

$$[\alpha \widetilde{\mathbb{U}} \beta] \xrightarrow{*} [\alpha] \vee ([\beta] \wedge [\alpha \widetilde{\mathbb{U}} \beta])$$

$$[F\beta] \xrightarrow{*} [\beta] \vee [F\beta]$$

The transitions follow directly the semantics of formulas; state $\top$ is a special state from which every timed word is accepted. As our automaton is alternating, on the right hand side of the transition we can write a boolean expression on successor states. We should also explain labels $*$ and $\varepsilon$ over transitions. Transition $\xrightarrow{*}$ is just a shorthand for transitions on all letters of the alphabet. Transitions $\xrightarrow{\varepsilon}$ and $\xrightarrow[x:=0]{\varepsilon}$ can be seen as eager $\varepsilon$-transitions of the automaton: they are executed as soon as they are enabled. The other way is to consider them as rewrite rules where the real transition of the automaton is obtained at the end of the rewriting, i.e., reaching a transition on a letter. In this interpretation we should not forget to accumulate resets. For example, the above rules give

$$[x.(\alpha \widetilde{\mathbb{U}} \beta)] \xrightarrow[x:=0]{*} [\alpha] \vee ([\beta] \wedge [\alpha \widetilde{\mathbb{U}} \beta])$$

as a "real" transition of the automaton.

All the states are accepting. Notice that in the case of positive formulas we will have a state $[F\beta]$ only when $\beta$ is of the form $(x \leq c) \wedge \beta'$. As we consider only non Zeno words, this assures that the language accepted from this state is correct even if the state $[F\beta]$ is accepting.

For other formulas of Constrained TPTL we first assume that for every positive formula we have already an automaton constructed by the above procedure. We then use the clauses above and the clause for the $\mathbb{U}$ operator

$$[\alpha \mathbb{U} \beta] \longrightarrow [\beta] \vee ([\alpha] \wedge [\alpha \mathbb{U} \beta])$$

to construct the part of the automaton corresponding the remaining formulas. The accepting states are all those corresponding to positive formulas. All the other states are rejecting.

A standard argument based on induction on the size of the formula shows that the translation is correct. For the complexity bound announced in the statement of the theorem, it is enough to check that the proof of the same complexity bound for alternating timed automata over finite words [17] can be translated into Constrained TPTL. □

4.1. **Relation with other logics.** Safety MTL [24] can be seen as an MTL fragment of positive TPTL. Indeed, both formalisms can be translated to automata with only accepting states, but the automata obtained from MTL formulas also have the locality property (cf. [24]). This property ensures that the clock is always reset when changing state. The example (4.1) shows that this is not the case for positive TPTL. The satisfiability problem for both logics is non-elementary [25].

Using equivalences mentioned above FlatMTL[12] with pointwise non Zeno semantics can be defined as a set of formulas of the grammar:

$$p \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \alpha \mathbb{U}_J \beta \mid \chi \mathbb{U}_I \beta \mid \chi \qquad J \text{ bounded and } \chi \in MITL,$$

where MITL is a version of MTL in which we do not allow equality constraints [6]. The original definition admits more constructs, but they are redundant in the semantics we consider.

Both FlatMTL and Constrained TPTL use two different sets of formulas. The MTL part of the later logic would look like

$$p \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \alpha \mathbb{U}_I \beta \mid \varphi \qquad \varphi \text{ positive.}$$

From this presentation it can be seen that there are at least two important differences: (i) constrained TPTL does not have restrictions on the left hand side of "until", and (ii) it uses the positive fragment instead of MITL. We comment on these two aspects below.

Allowing unrestricted "until" makes the logic more expressive but also more difficult algorithmically. For example, to get the non primitive recursive bound it is enough to use the formulas generated by the later grammar without the clause for positive formulas. This should be contrasted with the Expspace-completeness result for FlatMTL [12].

The use of positive fragment instead of MITL is also important. The two formalisms are very different in expressive power. The crucial technical property of MITL is that a formula of the form $\alpha \mathbb{U}_I \beta$ can change its value at most three times in every unit interval. This is used in the proof of decidability of FlatMTL, as the MITL part can be described in a "finitary" way. The crucial property of the positive fragment is that it can express only safety properties (and all such properties). We can remark that by reusing the construction of [22] we get undecidability of the positive fragment extended with a formula expressing that some action appears infinitely often. Theorem 5.2 presented in the next section implies that this is true even if we do not use punctual constraints in the positive fragment. In conclusion, we cannot add MITL to the positive fragment without losing decidability.

## 5. Undecidability without testing for equality

Ouaknine and Worrell [22] have proved undecidability of MTL over infinite words in the case of pointwise semantics. Their construction immediately implies that the decidability result from the last section is optimal if classes of accepting conditions are concerned.

**Theorem 5.1** (Ouaknine, Worrell)**.** *It is undecidable whether a given one-clock universal timed automaton $\mathcal{A}$ with weak $(1,2)$ conditions accepts some non Zeno word.*

Recall that weak parity conditions were defined on page 4; weak $(1,2)$ condition means that each accepting run contains only accepting states, or reaches $\top$. The construction in op. cit. relies on equality constraints. Indeed, if we do not allow equality constraints in MTL then we get a fragment called MITL, and the satisfiability problem for MITL over infinite words is decidable [6].

In this section we would like to show that a similar phenomenon is very particular to MTL and does not occur in the context of automata. We show that the undecidability result holds even when automata are only allowed to test if the clock is bigger than 1.

**Theorem 5.2.** *It is undecidable if a given one-clock universal timed automaton $\mathcal{A}$ with weak $(1,2)$ conditions accepts some non Zeno word, even when $\mathcal{A}$ does not use tests for equality.*

*Remark:* The above theorems stay true if we replace "non Zeno word" by "any word". This is because we can restrict the language of an automaton to non Zeno words: the set of non Zeno words is accepted by an automaton with weak $(1,1)$ conditions.

To prove Theorem 5.2 we encode a problem of deciding whether there is a run of a counter machine with insertion errors satisfying a (strong) Büchi condition. This section is split in two parts. In the first we introduce counter machines with insertion errors, and show undecidability of the problem in question. In the second we give an encoding of this problem into the emptiness problem for automata with weak $(1,2)$ conditions.

**Machines with insertion errors.** A *$k$-counter machine with insertion errors* $\mathcal{M}^g$ has configurations $(q, c^1, \ldots, c^k)$ consisting of a control state $q \in Q$ and values of the counters $c^i \in \mathbb{N}$. There are three kinds of transitions: $(q : c^i := c^i + 1; \texttt{goto } q')$ or $(q : \texttt{if } c^i = 0 \texttt{ then goto } q')$ or $(q : \texttt{if } c^i > 0 \texttt{ then } c^i := c^i - 1; \texttt{goto } q')$. The set of transitions $\delta$ of $\mathcal{M}^g$ gives rise to a relation between configurations, describing a single step of $\mathcal{M}^g$. The machine has insertion errors, which means that before and after every step it may increase any of its counters by any value. We will denote this by $(q, c^1, \ldots, c^k) \longrightarrow (q', c'^1, \ldots, c'^k)$, to say that we may reach configuration $(q', c'^1, \ldots, c'^k)$ from $(q, c^1, \ldots, c^k)$ using some transition from $\delta$ and possibly increasing some counters before and after the transition. The initial configuration of the machine $\mathcal{M}^g$ is $(q_0, 0, \ldots, 0)$. Together with the machine there is given some subset of states $Q_{acc} \subseteq Q$. We say that a run of $\mathcal{M}^g$ *satisfies the Büchi condition* if in infinitely many of its configurations there appears a state from $Q_{acc}$.

**Theorem 5.3** (Ouaknine, Worrell [22])**.** *It is undecidable whether a given 5-counter machine with insertion errors $\mathcal{M}^g$ has a run satisfying the Büchi condition.*

For completeness, we give a short proof of Theorem 5.3 by reduction to boundedness of a lossy 4-counter machine. The principle of *lossy $k$-counter machine* is similar to that with insertion errors, with a difference that before or after every step it may decrease any of its counters by any value (instead of increasing). We say that a run of such a machine is bounded, iff there is a common bound for values of all counters in all configurations throughout the run. We will use the following result.

**Theorem 5.4** (Mayr [18])**.** *It is undecidable whether every run of a given lossy 4-counter machine $\mathcal{M}^l$ is bounded.*

*Proof of Theorem 5.3.* Coming back to insertion errors, first note that a counter machine with insertion errors is exactly the same as lossy counter machine working backward. Let $\mathcal{M}^l$ be a given lossy 4-counter machine. We construct a 5-counter machine $\mathcal{M}^g$ that can simulate in a backward fashion a computation of $\mathcal{M}^l$ on the first four counters. This machine is able to go from a configuration $(q, c^1, c^2, c^3, c^4, c^5)$ to a configuration $(q_0, 0, 0, 0, 0, c^5)$ iff $\mathcal{M}^l$ can go from $(q_0, 0, 0, 0, 0)$, that is the initial configuration, to $(q, c^1, c^2, c^3, c^4)$. Additionally to the states of $\mathcal{M}^l$, the machine has some auxiliary states, among them an accepting state

$q_{acc}$. The machine will start in the state $q_{acc}$, and this state will be reachable only from a configuration $(q_0, 0, 0, 0, 0, c^5)$. In the state $q_{acc}$, the machine increases $c^5$ by 1 and then (in a nondeterministic way) increases counters $c^1, c^2, c^3, c^4$, so that $c^1 + c^2 + c^3 + c^4 \geq c^5$. To do that it may the move value of $c^5$ simultaneously into $c^1$ and $c^2$, then move value from $c^2$ back to $c^5$ and finally while decreasing $c^1$ increase $c^2, c^3, c^4$. After that it chooses a state of $\mathcal{M}^l$ and starts computing backward (using only the first four counters). When configuration $(q_0, 0, 0, 0, 0, c^5)$ is reached we make the machine to go to $(q_{acc}, 0, 0, 0, 0, c^5)$.

Assume that $\mathcal{M}^l$ has an unbounded computation. We will show that $\mathcal{M}^g$ has a run visiting $q_{acc}$ infinitely often. Suppose that some initial fragment of this run is already constructed and we are in a configuration $(q_{acc}, 0, 0, 0, 0, c^5)$ for some value of $c^5$. As $\mathcal{M}^l$ has an unbounded computation, it can reach a configuration $(q, c^1, c^2, c^3, c^4)$ with the sum of the counters bigger than $c^5 + 1$. We increase $c^5$ by 1, distribute $c^5$ into other counters to get the values $c^1, c^2, c^3, c^4$, we choose the state $q$ and then execute the computation of $\mathcal{M}^l$ backwards, starting from $(q, c^1, c^2, c^3, c^4)$. When reaching $(q_0, 0, 0, 0, 0, c^5 + 1)$ we go to $(q_{acc}, 0, 0, 0, 0, c^5 + 1)$ and repeat this process. This gives the required infinite computation.

For the opposite direction, assume that there is a computation of $\mathcal{M}^g$ satisfying the Büchi condition. Every appearance of $q_{acc}$ is followed by some initialization, and by a backward computation of $\mathcal{M}^l$, starting in a configuration of size bigger than the value of $c^5$ and ending in $(q_0, 0, 0, 0, 0)$. However, every time this happens the value of $c^5$ increases by at least one. So we get computations of $\mathcal{M}^l$ ending in bigger and bigger configurations. By König's lemma, there exists also an unbounded computation of $\mathcal{M}^l$. □

**Encoding machines into alternating automata.** Now we return to the proof of Theorem 5.2, which occupies the rest of this section. For given 5-counter machine with insertion errors $\mathcal{M}^g$ we will construct an alternating one-clock timed automaton $\mathcal{A}$ that accepts some infinite word iff $\mathcal{M}^g$ has a run satisfying the Büchi condition. The input alphabet of $\mathcal{A}$ will consist of the instructions of $\mathcal{M}^g$ and some auxiliary letters whose use will be explained later,

$$\Sigma = \delta \cup \{\texttt{shc}, \texttt{sh\$}, \texttt{new}, \texttt{init}\}.$$

As states of $\mathcal{A}$ we take

$$Q_+ = Q_{\mathcal{M}} \cup \{1, 2, 3, 4, 5, \$, q_\infty, q_{init}\} \quad \text{and} \quad Q_- = \{q_-\}.$$

States $Q_{\mathcal{M}} \cup \{1, 2, 3, 4, 5\}$ will be used to represent configurations of $\mathcal{M}^g$: the current state and the values of the five counters. States $q_\infty$ and $q_-$ will encode the condition on successful runs. State \$ is important for technical reasons explained later. State $q_{init}$ is just the initial state that will not be reachable from other states.

In our description below we will consider the characterization of acceptance given by Lemma 3.3. In this presentation a run of $\mathcal{A}$ is a sequence

$$P_1 \overset{a_1,t_1}{\hookrightarrow} P_2 \overset{a_2,t_2}{\hookrightarrow} P_3 \dots,$$

where each $P_i \subseteq \mathcal{P}(Q_{\mathcal{A}} \times \mathbb{R}^+)$ is a set of pairs $(q, v)$ consisting of a state of $\mathcal{A}$ and a valuation of the clock. We call such a set an *extended configuration* of $\mathcal{A}$, or an *e-configuration* for short. Compared with Lemma 3.3 we have joined together a transition letting the time pass with an action transition and write just $\overset{a,t}{\hookrightarrow}$ transitions. In what follows we will use only two regions: $\mathcal{I}_1 = [0, 1]$ and $\mathcal{I}_\infty = (1, \infty)$.

**Definition 5.5.** An e-configuration $P$ of $\mathcal{A}$ is *well-formed* if:

- For every $(q, v) \in P$: if $q \in \{1, \ldots, 5, \$\}$ then $v \in \mathcal{I}_1$, and $v \in \mathcal{I}_\infty$ otherwise.
- For every $v \in \mathcal{I}_1$ there is at most one $q$ with $(q, v) \in P$.
- In $P$ there is exactly one pair with a state from $Q_{\mathcal{M}}$, exactly one pair with the state $q_\infty$, and no pairs with $q_{init}$.
- Suppose $(q, v)$ is in $P$ where $q \in \{1, \ldots, 5\}$. Then this pair is immediately preceded by some $(\$, v')$ (there is no pair $(q'', v'')$ in $P$ with $v' < v'' < v$).

Intuitively, a well-formed e-configuration is divided into two parts: the set of pairs with the clock value in $\mathcal{I}_1$ and those in $\mathcal{I}_\infty$. The first part can be seen as representing a word over $\{1, \ldots, 5, \$\}$ that is obtained by using the standard order on clock values. From the conditions above it follows that this word is of the form $\$^+ q_{i_1} \$^+ q_{i_2} \ldots \$^+ q_{i_n} \$^*$; where $q_{i_k} \in \{1, \ldots, 5\}$. Such a word represents values of the counters when the value of the counter $c^j$ is equal to the number of $j$ in the word. The clock values of pairs in $\mathcal{I}_\infty$ will not matter, so this part can be seen as a multiset of states. In this multiset there will be exactly one state from $Q_{\mathcal{M}}$ representing the state of the simulated machine. State $q_\infty$ plus some number of states $q_-$ will be there to encode a condition on a successful run.

The automaton $\mathcal{A}$ will pass also through e-configurations that are not well-formed, but in its accepting run it will have to repeatedly return to well-formed e-configurations.

**Example 5.6.** Consider an e-configuration

$$\{\$^{0.1}, 1^{0.2}, \$^{0.3}, \$^{0.4}, 2^{0.6}, \$^{0.8}, 1^{0.9}, q^5, q_-^5, q_\infty^5\}$$

where for readability we write a pair $(\$, 0.1)$ as $\$^{0.1}$; and similarly for all other elements of the set. This e-configuration is well-formed and encodes the configuration $(q, 2, 1, 0, 0, 0)$ of $\mathcal{M}^g$. Observe that there are infinitely many well-formed e-configurations encoding this configuration of $\mathcal{M}$.

Now we describe transitions of the automaton. In order to have an intuition for reading the rules below it is important to observe that if the automaton reads a letter $\sigma$ then all states in its current e-configuration have to make a transition according to some rule labeled $\sigma$. In consequence, if there is a state in the e-configuration that does not have a rule for $\sigma$ then the automaton cannot read $\sigma$.

The automaton starts in the state $q_{init}$ and waits at least one time unit to start its two copies: one in a state $q_0$ and another in $q_\infty$ (where $q_0$ is the initial state of $\mathcal{M}^g$),

$$q_{init}, \mathcal{I}_\infty \xrightarrow{\texttt{init}} q_0 \wedge q_\infty.$$

This means that the e-configuration becomes $\{(q_0, v), (q_\infty, v)\}$ with $v \in \mathcal{I}_\infty$.

States $\$$ for clock values $\leq 1$ are preserved by any transition,

$$\$, \mathcal{I}_1 \xrightarrow{\sigma} \$, \qquad \forall \sigma \in \Sigma.$$

Similarly states $1, \ldots, 5$, with the exception that a transition checking for zero should not be possible if the corresponding counter is non-zero,

$$i, \mathcal{I}_1 \xrightarrow{\sigma} i \qquad \forall i = 1, \ldots, 5 \; \forall \sigma \neq (q : \texttt{if } c^i = 0 \texttt{ then goto } q').$$

When the clock value for a pair with $\$$ or $i$ becomes greater than 1, it may be reset,

$$\begin{array}{ll} \$, \mathcal{I}_\infty \xrightarrow{\texttt{sh\$}} (\$, \texttt{reset}), & \\ i, \mathcal{I}_\infty \xrightarrow{\texttt{shc}} \$ \wedge (i, \texttt{reset}) & \forall i = 1, \ldots, 5, \\ q, \mathcal{I}_\infty \xrightarrow{\sigma} q & q \in Q_{\mathcal{M}} \cup \{q_\infty, q_-\}, \; \sigma = \texttt{sh\$} \text{ or } \sigma = \texttt{shc}. \end{array}$$

Note that the transition on \$ reads a different letter than that on $i$. In consequence, if in a e-configuration there are pairs with both \$ and $i$ having clock values in $\mathcal{I}_\infty$ then neither `sh`\$ nor `shc` are possible. As we will have no more transitions from $(\$, \mathcal{I}_\infty)$ this means that the automaton will be blocked in such e-configuration.

Now we consider moves on transitions of the machine $\mathcal{M}^g$. For $\sigma = (q : \texttt{if } c^i = 0 \texttt{ then goto } q')$ we just do

$$q, \mathcal{I}_\infty \xrightarrow{\sigma} q'.$$

Note that, thanks to earlier restriction, the transition is possible only when there are no $i$ states in the e-configuration. For $\sigma = (q : \texttt{if } c^i > 0 \texttt{ then } c^i := c^i - 1; \texttt{goto } q')$ we do

$$q, \mathcal{I}_\infty \xrightarrow{\sigma} q',$$
$$i, \mathcal{I}_\infty \xrightarrow{\sigma} \top.$$

For $\sigma = (q : c^i := c^i + 1; \texttt{goto } q')$ we do

$$q, \mathcal{I}_\infty \xrightarrow{\sigma} q' \wedge \$ \wedge (i, \texttt{reset}).$$

As the machine should allow insertion errors, we add a transition

$$q, \mathcal{I}_\infty \xrightarrow{\texttt{new}} q \wedge \$ \wedge (i, \texttt{reset}).$$

Finally, we have special states $q_\infty$ and $q_-$, that are used to ensure that states from $Q_{acc}$ appear infinitely often. The state $q_\infty$ produces repeatedly new $q_-$ states,

$$q_\infty, \mathcal{I}_\infty \xrightarrow{\sigma} q_\infty \wedge q_- \qquad \forall \sigma \in \Sigma.$$

The state $q_-$ is the only one, which is in $Q_-$, so in the accepting run every $q_-$ state has to disappear after some time. States $q_-$ disappear, when there is a transition ending in a state from $Q_{acc}$,

$$\begin{array}{ll} q_-, \mathcal{I}_\infty \xrightarrow{\sigma} \top & \forall \sigma = (\ldots \texttt{goto } q'), q' \in Q_{acc}, \\ q_-, \mathcal{I}_\infty \xrightarrow{\sigma} q_- & \text{for all other } \sigma. \end{array}$$

**Example 5.7.** Let us see how a transition $\sigma = (q : \texttt{if } c^2 > 0 \texttt{ then } c^2 := c^2 - 1; \texttt{goto } q_2)$ is simulated from the e-configuration in Example 5.6. One possibility is to immediately execute a transition reading $\sigma$. We get the e-configuration

$$\{\$^{0.1}, 1^{0.2}, \$^{0.3}, \$^{0.4}, 2^{0.6}, \$^{0.8}, 1^{0.9}, q_2^5, q_-^5, q_\infty^5\}$$

which is well-formed and encodes the configuration $(q_2, 2, 1, 0, 0, 0)$ of $\mathcal{M}^g$. The second counter has not been decreased, but this is correct as the machine is allowed to do incremental errors.

If we really want to decrease the second counter, we have to ensure that a pair with 2 is in the $\mathcal{I}_\infty$ region. If we let pass, say, 0.2 units of time we get e-configuration

$$\{\$^{0.3}, 1^{0.4}, \$^{0.5}, \$^{0.6}, 2^{0.8}, \$^1, 1^{1.1}, q^{5.2}, q_-^{5.2}, q_\infty^{5.2}\}\ .$$

Then we execute a transition `shc` and we get

$$\{1^0, \$^{0.3}, 1^{0.4}, \$^{0.5}, \$^{0.6}, 2^{0.8}, \$^1, q^{5.2}, q_-^{5.2}, q_\infty^{5.2}\}\ .$$

After time 0.1 we can execute a transition `sh`\$, getting

$$\{\$^0, 1^{0.1}, \$^{0.4}, 1^{0.5}, \$^{0.6}, \$^{0.7}, 2^{0.9}, q^{5.3}, q_-^{5.3}, q_\infty^{5.3}\}\ .$$

Then after 0.2 we execute a transition $\sigma$, getting a well-formed e-configuration corresponding to $(q_2, 2, 0, 0, 0, 0)$:

$$\{\$^{0.2}, 1^{0.3}, \$^{0.6}, 1^{0.7}, \$^{0.8}, \$^{0.9}, q_2^{5.5}, q_-^{5.5}, q_\infty^{5.5}\} .$$

Now consider the transition $\sigma' = (q_2 : c^3 := c^3 + 1; \texttt{goto } q_3)$ of $\mathcal{M}^g$. We execute a transition $\sigma'$ from the above e-configuration after time 0.1 (recall that executing two transitions at the same time is forbidden), getting

$$\{3^0, \$^{0.3}, 1^{0.4}, \$^{0.7}, 1^{0.8}, \$^{0.9}, \$^1, q_3^{5.6}, q_-^{5.6}, q_\infty^{5.6}\} .$$

After additional time 0.1 we execute the transition sh$, getting a well-formed e-configuration corresponding to $(q_3, 2, 0, 1, 0, 0)$:

$$\{\$^0, 3^{0.1}, \$^{0.4}, 1^{0.5}, \$^{0.8}, 1^{0.9}, \$^1, q_3^{5.7}, q_-^{5.7}, q_\infty^{5.7}\} .$$

**Lemma 5.8.** *There exists a run of $\mathcal{M}^g$ satisfying the Büchi condition iff $\mathcal{A}$ accepts some infinite word.*

*Proof.* Assume that $\mathcal{M}^g$ has a run satisfying the Büchi condition. From the initial state, $\mathcal{A}$ may go to a well-formed e-configuration corresponding to the initial configuration of $\mathcal{M}^g$. Then every step of $\mathcal{M}^g$ may be simulated by $\mathcal{A}$: When $\mathcal{M}^g$ increases some of its counters, we may do the same using transitions on letters new and then sh$. When $\mathcal{M}^g$ executes a transition $\sigma = (q : \texttt{if } c^i = 0 \texttt{ then goto } q')$ we may do the same in $\mathcal{A}$ reading letter $\sigma$. When $\mathcal{M}^g$ does $\sigma = (q : c^i := c^i + 1; \texttt{goto } q')$, we do the same reading letter $\sigma$ and then sh$. It is easy to check, that after each step the resulting e-configuration remains well-formed.

The only complicated transition is $\sigma = (q : \texttt{if } c^i > 0 \texttt{ then } c^i := c^i - 1; \texttt{goto } q')$. Suppose that the automaton is in a well-formed e-configuration $P$. Let us look at the biggest valuation $v \leq 1$ appearing in $P$. By the conditions of well-formedness (c.f. Definition 5.5) there is exactly one state $q \in Q_+$ such that $(q, v) \in P$. This state can be one of $1, \ldots, 5, \$$. The automaton lets the time pass so that $v$ becomes greater than 1, but all other valuations from $\mathcal{I}_1$ stay in $\mathcal{I}_1$. If $q = i$ then the automaton does $\sigma$. Otherwise, it does sh$ or shc followed by sh$ that has an effect of putting \$ or \$ followed by $q$ at the beginning of the e-configuration. After this we obtain a well-formed e-configuration where the one but the maximal valuation before became the maximal one. These operations are repeated until $q = i$. We are sure that this process ends, as there is a state $i$ in $P$.

To ensure that the obtained word is nonZeno, we have to wait some time after every transition of $\mathcal{M}^g$, doing shc and sh$ if necessary. Observe that every state $q_-$ would disappear when in the computation of $\mathcal{M}^g$ there is a transition ending in a state from $Q_{acc}$. As this computation satisfies the Büchi condition, this will happen infinitely often.

For the other direction, consider an accepting run of $\mathcal{A}$ on some word. In the first step, $\mathcal{A}$ has to reach a well-formed e-configuration corresponding to the initial configuration of $\mathcal{M}^g$. Let us see what may happen from any well-formed e-configuration. Suppose that time passes and the clock value for some states $1, \ldots, 5, \$$ becomes greater than 1. If it happens simultaneously for state \$ and some state $i$, then from the obtained e-configuration there will be no more transitions. If it happens only for state \$, then the only possible transition is the one reading sh$ after which we go back to a well-formed e-configuration corresponding to the same configuration of $\mathcal{M}^g$. If it happens just for some state $i$, then the automaton can read either shc or some $(q : \texttt{if } c^i > 0 \texttt{ then } c^i := c^i - 1; \texttt{goto } q')$. If it reads shc, then after that it has to read sh$, and we also are back in a well-formed e-configuration corresponding to the same configuration of $\mathcal{M}^g$. If it reads $\sigma = (q : \texttt{if } c^i > 0 \texttt{ then } c^i := c^i - 1; \texttt{goto } q')$, then we immediately get a well-formed e-configuration.

Transitions reading `shc` or `sh$` when no state of $1, \dots, 5, \$$ has the clock value above 1 does not change the configuration. A transition reading `new` has to be followed by a transition `sh$` and we get a well-formed e-configuration with one of the counters increased. A transition reading $\sigma = (q : \texttt{if } c^i = 0 \texttt{ then goto } q')$ is possible only when counter $c^i$ is zero. After a transition reading $\sigma = (q : c^i := c^i + 1; \texttt{goto } q')$ there has to be a transition reading `sh$` and we get a well-formed e-configuration that corresponds to a correct configuration of $\mathcal{M}^g$. A transition reading $\sigma = (q : \texttt{if } c^i > 0 \texttt{ then } c^i := c^i - 1; \texttt{goto } q')$ always gives us a well-formed e-configuration. The obtained e-configuration correctly represents the result but for the fact that the counter $i$ may not be decremented. This is not a problem as we are simulating a machine with insertion errors, so we can suppose that the incrementation error has occurred immediately after execution of this instruction.

The above argument gives some computation of $\mathcal{M}^g$ constructed from an accepting computation of $\mathcal{A}$. So every $q_-$ disappears after some time on that computation of $\mathcal{A}$. This is only possible when reading a letter of the form $(\dots \texttt{goto } q')$ with $q'$ an accepting state of $\mathcal{M}^g$. As $q_-$ needs to disappear infinitely often, the obtained computation of $\mathcal{M}^g$ is an infinite computation satisfying the Büchi condition. □

As the choice of counter machine $\mathcal{M}^g$ was arbitrary and the construction of $\mathcal{A}$ from $\mathcal{M}^g$ was effective, Lemma 5.8 implies Theorem 5.2.

## 6. Conclusions

This paper presents a study of the emptiness problem for alternating timed automata. It gives a characterization of decidable cases of this problem in terms of the complexity of acceptance conditions. The main result shows that all the classes whose decidability has been left open are indeed decidable. This result gives new decidability results for logics for real-time.

Given this characterization, in order to find other, bigger, classes of alternating timed automata with decidable emptiness problem we need to look closer at the structure of automata. In this paper one case has been studied, namely when no punctual constraints are used. This case was motivated by the phenomenon observed for metric temporal logic: while the logic is undecidable, it becomes decidable when punctual constraints are disallowed. The second main result of the paper shows that in the case of automata such a simple restriction does not work: one does not get a bigger decidable class even if one restricts to extremely simple constraints. This indicates that in order to obtain larger decidable classes, the structure of resets should be also examined more closely.

## References

[1] P. Abdulla and B. Jonsson. Veryfying networks of timed processes. In *Proc. TACAS'98*, volume 1384 of *LNCS*, pages 298–312, 1998.
[2] P. Abdulla and B. Jonsson. Timed Petri nets and BQOs. In *Proc. ICATPN'01*, pages 53–70, 2001.
[3] P. A. Abdulla, J. Deneux, J. Ouaknine, K. Quaas, and J. Worrell. Universality analysis for one-clock timed automata. *Fundam. Inform.*, 89(4):419–450, 2008.
[4] P. A. Abdulla, J. Ouaknine, K. Quaas, and J. Worrell. Zone-based universality analysis for single-clock timed automata. In *FSEN'07*, number 4767 in LNCS, pages 98–112, 2007.
[5] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
[6] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.

[7] R. Alur, L. Fix, and T. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 204, 1997.
[8] R. Alur and T. A. Henzinger. A really temporal logic. *J. ACM*, 41(1):181–204, 1994.
[9] P. Bouyer. Model-checking timed temporal logics. In *Workshop on Methods for Modalities (M4M-5)*, Electronic Notes in Theoretical Computer Science, Cachan, France, 2009. Elsevier Science Publishers. To appear.
[10] P. Bouyer, F. Chevalier, and N. Markey. On the expressiveness of TPTL and MTL. In *FSTTCS'05*, volume 3821 of *LNCS*, pages 432–443, 2005.
[11] P. Bouyer, N. Markey, J. Ouaknine, P. Schnoebelen, and J. Worrell. On termination for faulty channel machines. In *STACS'08*, volume 08001 of *Dagstuhl Seminar Proceedings*, pages 121–132, 2008.
[12] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. The cost of punctuality. In *LICS'07*, pages 109–120, 2007.
[13] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. On expressiveness and complexity in real-time model checking. In *ICALP'08*, volume 5126 of *LNCS*, pages 124–135, 2008.
[14] Y. Hirshfeld and A. M. Rabinovich. Logics for real time: Decidability and complexity. *Fundam. Inform.*, 62(1):1–28, 2004.
[15] D. V. Hung and W. Ji. On the design of hybrid control systems using automata models. In *FSTTCS'96*, number 1180 in LNCS, pages 156–167, 1996.
[16] S. Lasota and I. Walukiewicz. Alternating timed automata. In *FOSSACS'05*, number 3441 in Lecture Notes in Computer Science, pages 250–265, 2005.
[17] S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Trans. Comput. Log.*, 9(2), 2008.
[18] R. Mayr. Undecidable problems in unreliable computations. *Theoretical Computer Science*, 1-3(297):337–354, 2003.
[19] A. W. Mostowski. Hierarchies of weak automata and week monadic formulas. *Theoretical Computer Science*, 83:323–335, 1991.
[20] F. Murlak. Weak index versus borel rank. In *STACS'08*, Dagstuhl Seminar Proceedings, pages 573–584. Dagsr, 2008.
[21] J. Ouaknine and J. Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *Proc. LICS'04*, pages 54–63, 2004.
[22] J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *LICS'05*, pages 188–197, 2005.
[23] J. Ouaknine and J. Worrell. On metric temporal logic and faulty Turing machines. In *FoSSaCS*, volume 3921 of *LNCS*, pages 217–230, 2006.
[24] J. Ouaknine and J. Worrell. Safety metric temporal logic is fully decidable. In *TACAS'06*, number 3920 in LNCS, pages 411–425, 2006.
[25] J. Ouaknine and J. Worrell. On the decidability and complexity of metric temporal logic over finite words. *Logical Methods in Computer Science*, 3(1), 2007.
[26] J. Ouaknine and J. Worrell. Some recent results in metric temporal logic. In *FORMATS'08*, number 5215 in LNCS, pages 1–13, 2008.
[27] M. Y. Vardi and P.Wolper. Automata theoretic techniques for modal logics of programs. In *Sixteenth ACM Symposium on the Theoretical Computer Science*, 1984.
[28] K. Wagner. Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen. *J. Inf. Process. Cybern. EIK*, 13:473–487, 1977.
[29] K. Wagner and L. Staiger. Automatentheoretische und automatenfreie charakterisierungen topologischer klassen regularer folgenmengen. *EIK*, 10:379–392, 1974.
[30] T. Wilke. Classifying discrete temporal properties. Habilitation thesis, Kiel, Germany, 1998.