Weak Alternating Timed Automata

Pawel Parys[1]* and Igor Walukiewicz[2]**

[1] Warsaw University,Poland
[2] LaBRI, 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 the 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] is a widely used model of real-time systems. It is 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. Alternating timed automata have been introduced in [15, 20] following a sequence of results [1, 2, 19] indicating that a restriction to one clock can make some problems decidable. The emptiness of one clock alternating automata is decidable over finite words, but not over infinite words [23, 16]. Undecidability proofs 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 left open the possibility that safety properties and "almost always" properties can be decidable. This is indeed what we prove here.

Automata theoretic approach to temporal logics [26] is by now a standard way of understanding these formalisms. For example, we know that the modal μ-calculus corresponds to all automata, and LTL to very weak alternating automata, or equivalently, to counter-free nondeterministic automata [29]. By translating a logic to automata we can clearly see combinatorial challenges posed by the formalism. We can 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.

* Author supported by Polish government grant no. N206 008 32/0810.
** Author supported by project DOTS (ANR-06-SETI-003).

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 sequences. Nevertheless, Ouaknine and Worrell [22] have shown decidability of the emptiness problem provided all states are accepting, and some locality restriction on the transition function holds. Using this, they have identified a decidable fragment of MTL called Safety MTL.

In this paper we show that our main result allows to get a decidable fragment of TPTL [8] with one variable, that we call Constrained TPTL. This fragment contains Safety MTL and allows all eventually formulas. Its syntax has also some similarities with another recently introduced logic: FlatMTL [11, 12]. 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]. The later is a sub-logic of MTL where punctual 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 4), 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 very weak syntax of the logic that makes MITL decidable.

Related work The idea of restricting to one clock automata dates back at least to [14]. Alternating timed automata where studied in a number of papers [16, 23, 4, 3]. The most relevant result here is the decidability of the emptiness for the case when when all states are accepting and some locality condition holds [22]. One of technical contributions of the present paper is to remove the locality restriction, and to add a non-accepting layer of states on the top of the accepting one.

For a long time MITL [6] was the most prominent example of a decidable logic for real-time. In [23] Ouaknine and Worrell remark that MTL over finite words can be translated to alternating timed automata, and hence it is decidable. They also show that over infinite words the logic is undecidable (which is a stronger result than undecidability for the automata model in [16]). They have proposed a fragment of MTL, called Safety MTL. Decidability of this fragment was shown in [22] by reducing to the class of ATA mentioned in the previous paragraph. A fragment of MTL called FlatMTL [11, 12] represents an interesting but technically different direction of development (cf. Sect. 4).

We should also discuss the distinction between continuous and pointwise semantics. In the later, 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 α 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 en-

coding properties of monadic predicates over reals. Yet, it seems sufficient for descriptions of behaviors of devices, like timed automata, over time [24]. 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 [13] is more suitable to deal with continuous semantics.

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 [27], 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 [28]. There are several results testifying their relevance. For example Mostowski [17] 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 [18] and references therein.

Organization of the paper After a section with basic definitions we state our main decidability result (Theorem 2) and an accompanying undecidability result (Theorem 4). We give an outline of the proof of the former theorem. Sect. 4 introduces Constrained TPTL, gives a translation of the logic into a decidable class of alternating timed automata, and discusses relations with FlatMTL.

For the reasons of space, proofs are largely omitted. They can be found in the full version of the paper [25].

Acknowledgements: The second author would like to thank Abraham Riche for his cooperation in early stages of this work.

2 Preliminaries

A *timed word* over a finite alphabet Σ is a sequence: $w = (a_1, t_1)(a_2, t_2)\ldots$ of pairs from $\Sigma \times \mathbb{R}_+$. We require that the sequence $\{t_i\}_{i=1,2,\ldots}$ is strictly increasing and unbounded (non Zeno).

We will consider alternating timed automata (ATA) with one clock [16]. Let x be this clock and let Φ denote the set of all comparisons of x with constants, eg. $(x < 1 \wedge x \geq 0)$. A one-clock ATA over an alphabet Σ 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 Ω determines the parity acceptance condition. The transition function of the automaton δ 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 θ 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" of the form $(q, \texttt{nop})$ or $(q, \texttt{reset})$ then the automaton changes the state to q, and either does nothing or sets the value of the clock to 0, respectively. Formula $\top$ is unconditionally accepting, and $\bot$ unconditinally rejecting.

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 exaclty one θ s.t. $\delta(q, a, \theta)$ is defined, and v satisfies θ.

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 condition is that in the weak case we consider all occurrences of states and in the strong case only those that occur infinitely often. We will mostly use automata with weak conditions. Whenever we will be considering strong conditions we will say it explicitly.

An alternating timed automaton $\mathcal{A}$ and a timed word $w = (a_1, t_1)(a_2, t_2)\dots$ determine 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 $v' := v_k + t_{k+1} - t_k$. Let θ be the unique (by the partition condition) constraint such that v' satisfies θ and $b = \delta(q_k, a_{k+1}, \theta)$ is defined. Now the outcome of the phase is determined by the formula b. There are three 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 $(q_{k+1}, v_{k+1}) := (q, f(v'))$. A new phase is starting from this configuration.
- $b = \top, \bot$ the play ends.

The winner is Eve if 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 *partial 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 that assigns to each such sequence ending in Eve's position a next move of Eve. A strategy is winning if Eve wins whenever she applies this strategy.

Definition 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 Ω is the pair consisting of the minimal and the maximal value of Ω: $(\min(\Omega(Q)), \max(\Omega(Q)))$. 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 theses are states with rank 0.

3 Decidability for One-Clock Timed Automata

We are interested in the emptiness problem for one clock ATA. As mentioned in the introduction, the problem is undecidable for automata with strong Büchi conditions. Here we will show a decidability result for automata with weak acceptance conditions of index $(0, 1)$. A different presentation of these automata is that they are strong Büchi automata where there are no transitions from an accepting state to a non-accepting state. 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 or weak) Büchi acceptance condition says simply: there are only finitely many states from Q_-. So the automaton accepts if Eve has a strategy to reach $\top$, or to satisfy this condition.

Theorem 2. *For one-clock Büchi alternating timed automata with no transitions from states in Q_+ to states in Q_-: it is decidable whether a given automaton accepts a non Zeno timed word.*

Ouaknine and Worrell [21] have proved undecidability of MTL over infinite timed words. Their construction immediately implies undecidability for weak automata with $(1, 2)$ condition. So the above decidability result is optimal with respect to index of the accepting condition.

Theorem 3 (Ouaknine, Worell). *It is undecidable whether a given one-clock Büchi nondeterministic timed automaton $\mathcal{A}$ accepts every infinite word, even when there are no transitions in $\mathcal{A}$ from states in Q_- to states in Q_+.*

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 satisfiability problem for MITL over infinite words is decidable [6]. We show that this phenomenon does not appear in the context of automata.

Theorem 4. *It is undecidable whether a given one-clock Büchi alternating timed automaton $\mathcal{A}$ accepts an infinite word, even when there are no transitions in $\mathcal{A}$ from states in Q_- to states in Q_+, and when $\mathcal{A}$ does not test for equality.*

In the rest of the section we give an outline of the proof of Theorem 2. Due to space restrictions it is not possible to present the proof of Theorem 4. The

proof given in the full version of the paper [25] shows undecidability even when one uses only tests: $(x \geq 1)$, and its negation.

To fix the notation we take a one clock ATA:

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

We will assume that the transition function satisfies the partition condition. For simplicity, we also assume that every value of δ is a boolean formula in a disjunctive normal form. Moreover, we will require that in every disjunct of every transition of $\mathcal{A}$ there is some pair with `reset` and some pair with `nop`. Every automaton can be easily put into this form.

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.

3.1 Construction of $\mathcal{H}(\mathcal{A})$

This is surely the less interesting part of the proof, unfortunately we need to spend some time here as $\mathcal{H}(\mathcal{A})$ is the object we work with in the second step.

We need to reformulate the definition of acceptance of $\mathcal{A}$ in terms of a sequence instead of a tree. Additionally, we would like to abstract from time values in states and transitions, but still be able to tell if a run is non Zeno.

We start with the definition of regions. As we have only one clock we take:

$$\mathtt{reg} := \{\{0\}, (0,1), \{1\}, (1,2), \ldots, (d_{max}-1, d_{max}), \{d_{max}\}, (d_{max}, +\infty)\},$$

where d_{max} denotes the biggest constant appearing in δ, i.e., the transition function of the automaton. 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 its region; and let $\mathtt{fract}(v)$ denote the fractional part of v.

We will use the transition alphabet:

$$\overline{\Sigma} = \Sigma \cup \{(\mathtt{delay}, \varepsilon)\} \cup (\{\mathtt{delay}\} \times \Sigma).$$

A transition on $a \in \Sigma$ will represent an execution of an action. A transition on $(\mathtt{delay}, \varepsilon)$ will represent a passage of time with some valuation changing a region. Finally, we will have the most complicated case of $(\mathtt{delay}, a)$ transitions. Such a transition simulates a passage of time until a region becomes a one-point region, execution of the action at this moment, and letting some time pass to get into the next interval region. We will only present the transitions of the last type, the other two being simpler.

In $\mathcal{H}(\mathcal{A})$ the states will be finite words of the form $\Lambda_I^* \cdot \Lambda_\infty$, where $\Lambda_I = \mathcal{P}(Q \times \mathtt{reg}_I)$ and $\Lambda_\infty = \mathcal{P}(Q \times \{\infty\})$. The transitions on an action $(\mathtt{delay}, a)$ will have the form:

$$(\lambda_1 \ldots \lambda_k, \lambda_\infty) \xrightarrow{(\mathtt{delay}, a)} (\delta' \lambda'_1 \ldots \lambda'_{k-1}, \lambda''_\infty)$$

where the elements on the right are obtained by performing the following steps:

- (Letting the time pass to reach singleton region.) We change regions in λ_k. Every pair $(q, \mathcal{I}_d) \in \lambda_k$ becomes $(q, \{d\})$. Let us denote the result by λ_k^1.
- (Performing the action.) For $i = 1, \ldots, k, \infty$ we take λ_i', δ_i' such that: $\lambda_k^1 \xrightarrow{a}_{\mathcal{A}} (\lambda_k', \delta_k')$ and $\lambda_i \xrightarrow{a}_{\mathcal{A}} (\lambda_i', \delta_i')$ for $i \neq k$.
- (Letting the time pass again) We increase again regions in λ_k': from $\{d\}$ they become $\mathcal{I}_{d+1}$, or $\mathcal{I}_\infty$ if $d = d_{max}$.
- (Grouping the results) We put $\delta' = \bigcup \delta_i' \cup \{(q, \mathcal{I}_d) : (q, \{d\}) \in \lambda_k', d < d_{max}\}$ and $\lambda_\infty'' = \lambda_\infty' \cup \{(q, \mathcal{I}_\infty) : (q, \{d_{max}\}) \in \lambda_k'\}$.

We write $c \to c'$, $c \xrightarrow{(\texttt{delay},\cdot)} c'$, $c \twoheadrightarrow c'$, $c \xrightarrow{\Sigma^*}\!\!\!\!\!\rightarrow c'$ 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 Σ, respectively.

We say that a path is *good* if it passes through infinitely many transitions labelled by letters $(\texttt{delay}, \cdot)$. The whole point of this type of transitions is that they allow to capture non Zeno behaviours:

Lemma 5. *$\mathcal{A}$ accepts an infinite non Zeno timed word iff there is a good path in $\mathcal{H}(\mathcal{A})$ starting in the state $(\{(q_0, \mathcal{I}_1)\}, \{\emptyset, I_\infty\})$ with only finitely many appearances of states from Q_-.*

3.2 Finding a Good Path in $\mathcal{H}(\mathcal{A})$.

By Lemma 5, our problem reduces to deciding if there is a good path in $\mathcal{H}(\mathcal{A})$. The decision procedure works in two steps. In the first step we compute the set $\widehat{G}$ of all configurations from which there exists a good computation. Observe that if a configuration from $\widehat{G}$ has only states from Q_+ then this configuration is accepting. 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 a good accepting computation from a configuration iff it is possible to reach from it a configuration in $\widehat{G}$ that has only Q_+ states. Once we know $\widehat{G}$, the later problem can be solved using the standard reachability tree technique.

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 in $\widehat{G}$ 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.

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 $(\lambda_1 \dots \lambda_k, \lambda_\infty) \preceq (\lambda'_1 \dots \lambda'_{k'}, \lambda'_\infty)$ if $\lambda_\infty \subseteq \lambda'_\infty$ and there exists a strictly increasing function $f\colon \{1, \dots, k\} \to \{1, \dots, k'\}$ such that $\lambda_i \subseteq \lambda'_{f(i)}$ for each i. Observe that here we use the fact that each λ_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. A standard application of Higman's lemma proves that $\preceq$ is a well-quasi-order.

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

Lemma 6. *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.*

The proof of the lemma follows from examining the definition of transitions. Some care is needed to ensure that the matching computation is good.

Corollary 7. *The set $\widehat{G}$ is downwards closed, so it can be described by the finite set of minimal elements that do not belong to it.*

As we have mentioned, 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. A standard argument based on well-quasi-orders and examination of a finite part of the reachability tree shows that this property is decidable.

Lemma 8. *Let X be a downwards closed set in $\mathcal{H}(\mathcal{A})$. It is decidable if from a given configuration one can reach a configuration in X with all states in Q_+.*

Computing $\widehat{G}$. In the rest of the section 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 need to use in an essential way a very special form of transitions our systems have.

We write $X{\uparrow} = \{c : \exists_{c' \in X} c' \preceq c\}$ for an upward closure of set X. Observe that by Lemma 6 the complement of $\widehat{G}$ is upwards 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 Σ), we have to reach a configuration from X:

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

We 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_∞ for this Z_i.

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

Lemma 9. *There is a good computation from a configuration c iff $c \notin Z_\infty\uparrow$.*

To compute Z_∞ 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. Once this is done, we can 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 $pre^\forall_{\Sigma^*}(pre^\forall_{\mathtt{delay}}(X))$ is that the length of its generators may be bounded by some function in the length of generators of X.

Lemma 10. *Given an upwards 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 $pre^\forall_{\Sigma^*}(pre^\forall_{\mathtt{delay}}(X))$ 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 11. *For all upper-closed sets X, the membership in $pre^\forall_{\Sigma^*}(pre^\forall_{\mathtt{delay}}(X))$ is decidable.*

These two lemmas allow us to calculate Z_i from Z_{i-1}, and this is the last piece we need to complete the proof the theorem. The proofs of the lemmas are quite long, and require some additional notions. They can be found in the full version of the paper [25].

4 Constrained TPTL

We present a fragment of TPTL (timed propositional temporal logic) that can be translated to automata from our decidable class. 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, 24].

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, and we denote it by TPTL^1. The syntax of the logic is:

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

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

The logic is evaluated over timed sequences $w = (a_1, t_1)(a_2, t_2)\ldots$ We define the satisfiability relation, $w, i, v \vDash \alpha$ saying that a formula α 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 $a_i = p$
$w, i, v \vDash x.\alpha$ if $w, i, t_i \vDash \alpha$ $\quad$ $w, i, v \vDash x \sim c$ if $t_i - v \sim c$
$w, i, v \vDash \alpha \mathbb{U} \beta$ 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 $\exists_{k \in (i,j)}$ $w, k, v \vDash \alpha)$
Until operators permit us to introduce sometimes and always operators:

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

For the following, it will be interesting to note that the two until operators are inter-definable once we have always and sometimes operators:

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

Observe that TPTL^1 subsumes metric temporal logic (MTL). For example: $\alpha \mathbb{U}_{(i,j)} \beta$ of MTL is equivalent to $x.(\alpha \mathbb{U}((x > i) \wedge (x < j) \wedge \beta))$. 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:

$$x.(F(b \wedge F(c \wedge x \leq 2))). \tag{1}$$

Informally, the formula says that there is an event b followed by an event c in less than 2 units of time.

The satisfiability problem over infinite timed sequences is undecidable for MTL [21], 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*:

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

These formulas can be translated into automata where all states are accepting. Observe that the formula (1) belongs to the positive fragment if we add redundant $(x \leq 2)$ after b. The set of formulas of *Constrained TPTL* is:

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

A translation of Constrained TPTL to automata is similar in a spirit to that for Safety MTL [22]. Once again we refer the reader to the full version [25].

Theorem 12. *It is decidable if there is a non Zeno timed word that is a model of a given Constrained TPTL formula. The complexity of the problem cannot be bounded by a primitive recursive function.*

Safety MTL [22] can be seen as a MTL fragment of positive TPTL. Indeed, both formalisms can be translated to automata with only accepting states, but the automata obtained from MTL formulas have also the locality property (cf. [22]). This property ensures that the clock is always reset when changing state. The example (1) shows that this is not the case for TPTL.

Using equivalences mentioned above, FlatMTL [11] with pointwise non Zeno semantics can be presented as a set of formulas given by the grammar:

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

The original definition admits more constructs, but they are redundant in the semantics we consider. Both FlatMTL and Constrained TPTL use two 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: Constrained TPTL does not have restrictions on the left hand side of until, and it uses positive fragment instead of MITL. We comment on these two aspects below.

Unrestricted until makes the logic more expressive but also more difficult algorithmically. For example, already the logic generated by the later grammar without the clause for positive formulas has non primitive recursive complexity. This should be contrasted with EXPSPACE-completeness result for FlatMTL.

The use of positive fragment instead of MITL is also important. The two formalisms have very different expressive powers. 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 [21] we get undecidability of the positive fragment extended with a formula expressing that some action appears infinitely often. Theorem 4 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.

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*, 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*, volume 3821 of *LNCS*, pages 432–443, 2005.
11. P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. The cost of punctuality. In *LICS*, pages 109–120, 2007.
12. P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. On expressiveness and complexity in real-time model checking. In *ICALP*, volume 5126 of *LNCS*, pages 124–135, 2008.
13. Y. Hirshfeld and A. M. Rabinovich. Logics for real time: Decidability and complexity. *Fundam. Inform.*, 62(1):1–28, 2004.
14. D. V. Hung and W. Ji. On the design of hybrid control systems using automata models. In *FSTTCS*, number 1180 in LNCS, pages 156–167, 1996.
15. S. Lasota and I. Walukiewicz. Alternating timed automata. In *FOSSACS'05*, number 3441 in Lecture Notes in Computer Science, pages 250–265, 2005.
16. S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Trans. Comput. Log.*, 9(2), 2008.
17. A. W. Mostowski. Hierarchies of weak automata and week monadic formulas. *Theoretical Computer Science*, 83:323–335, 1991.
18. F. Murlak. Weak index versus borel rank. In *STACS*, Dagstuhl Seminar Proceedings, pages 573–584. Dagsr, 2008.
19. 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.
20. J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *LICS*, pages 188–197, 2005.
21. J. Ouaknine and J. Worrell. On metric temporal logic and faulty Turing machines. In *FoSSaCS*, volume 3921 of *LNCS*, pages 217–230, 2006.
22. J. Ouaknine and J. Worrell. Safety metric temporal logic is fully decidable. In *TACAS*, number 3920 in LNCS, pages 411–425, 2006.
23. 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.
24. J. Ouaknine and J. Worrell. Some recent results in metric temporal logic. In *FORMATS*, number 5215 in LNCS, pages 1–13, 2008.
25. P. Parys and I. Walukiewicz. Weak alternating timed automata. HAL, http://hal.archives-ouvertes.fr/hal-00360122/fr/, 2009.
26. M. Y. Vardi and P.Wolper. Automata theoretic techniques for modal logics of programs. In *Sixteenth ACM Symposium on the Theoretical Computer Science*, 1984.
27. K. Wagner. Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen. *J. Inf. Process. Cybern. EIK*, 13:473–487, 1977.
28. K. Wagner and L. Staiger. Automatentheoretische und automatenfreie charakterisierungen topologischer klassen regularer folgenmengen. *EIK*, 10:379–392, 1974.
29. T. Wilke. Classifying discrete temporal properties. Habilitation thesis, Kiel, Germany, 1998.