Weak Alternating Timed Automata

Pawel Parys, Igor Walukiewicz

Weak Alternating Timed Automata

Pawel Parys[1] and Igor Walukiewicz[2]
[(1)]Warsaw University, Poland
[(2)]CNRS and Bordeaux University, France

February 19, 2009

Abstract

Alternating timed automata on infinite words are considered. The main result is the 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 constrains are allowed.

Timed automata [6] 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 [17] following a sequence of results [1, 2, 22] 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 [25, 18]. Undecidability proofs relay 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 [27] 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 to counter-free nondeterministic automata [30]. By translating a logic to automata we can clearly see a 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 [8] 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 [24] 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 decidable fragment of MTL called Safety MTL.

In this paper we show that our main result allows to get a decidable fragment of TPTL [9] 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 [13, 14]. 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 [7]. This 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 32), 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 [16]. Alternating timed automata where studied in a number of papers [18, 25, 5, 4, 3]. The most relevant result here is the decidability of emptiness for the case when when all states are accepting and some locality condition holds [24]. 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 [7] was the most prominent example of a decidable logic for real-time. In [25] 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 [18]). They have proposed a fragment of MTL, called Safety MTL. Decidability of this fragment was shown in [24] by reducing to the class of ATA mentioned in the previous paragraph. A fragment of MTL called FlatMTL [13, 14] represents an interesting but technically different direction of development. We will comment more on this in Section 3.

We should also discuss the distinction between continues 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 continues 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 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 emptiness of alternating timed automata in continues semantics is undecidable even over finite words. At present it seems that an approach through compositional methods [15] is more suitable to deal with continues 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 [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 [20] 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 [21] and references therein.

Organization of the paper After a section with basic definitions we show our main decidability result (Theorem 2). Section 3 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 32).

1 Preliminaries

A *timed word* over a finite alphabet Σ is a sequence

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

of pairs form $\Sigma \times \mathbb{R}_+$. We require that the sequence $\{t_i\}_{i=1,2,\ldots}$ 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 instance and that there cannot be infinitely many actions in a finite time (non Zeno).

We will consider alternating timed automata (ATA) with one clock [18]. 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

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

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

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

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

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", i.e., of the form $(q, \mathtt{reset})$ or $(q, \mathtt{nop})$ 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 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 $q_1, q_2, \ldots$ satisfies:

- *weak parity condition* if $\min\{q_i : i = 1, 2, \ldots\}$ is even,
- *strong parity condition* if $\liminf_{i=1,2,\ldots} \Omega(q_i)$ 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. 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 $w = (a_1, t_1)(a_2, t_2)\ldots$ 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 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 θ be a unique (by the partition condition) constraint such that v' satisfies θ and $b = \delta(q_k, a_{k+1}, \theta)$ is defined; if there is no such θ then Eve is blocked. 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;
- $b = (q, f) \in Q \times \{\mathtt{nop}, \mathtt{reset}\}$: the phase ends with the result $(q_{k+1}, v_{k+1}) := (q, f(v'))$. A new phase is starting from this configuration.

The winner is Eve if she is not blocked and the sequence of states appearing in the path satisfies 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 $\langle k, q, b\rangle$, 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 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.

2 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. 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) Büchi acceptance condition says simply: a word is accepted, if there is a strategy for Eve to find an infinite path on which there are only finitely many states from Q_-.

Theorem 2 *Let $\mathcal{A}$ be one-clock Büchi alternating timed automata with no transitions from states in Q_+ to states in Q_-. It is decidable whether the automaton accepts a non Zeno timed word.*

In the rest of the section we give the proof of this theorem. To fix the notation we take a one clock ATA:

$$\mathcal{A} = \langle Q, \Sigma, q_o, \delta, F \subseteq Q \rangle$$

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 every disjunct of every transition of $\mathcal{A}$ there is some pair with `reset` and some pair with `nop`. It is easy to convert any automaton to an equivalent automaton satisfying these conditions. This form of transitions is required for the structure lemmas of this subsection.

Our first step will be to construct some infinite transition system $\mathcal{H}(\mathcal{A})$, so that existence of an accepting run of $\mathcal{A}$ is equivalent to 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 this problem.

2.1 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 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 $\mathcal{T}$ 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 $\mathcal{T}$ 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 $\mathcal{T}$ iff P' can be obtained from P by the following nondeterministic process:

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

Definition 3 We will call a sequence $P_0, P_1, \dots$ of the states of $\mathcal{S}(\mathcal{A})$ accepting if the states of Q_- appear only in a finite number of P_i.

Lemma 4 *$\mathcal{A}$ 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
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 strategy. This strategy gives a run of the form required by the lemma. □

Our next goal is to remove time labels on transitions. Still we do not want just to erase them, as then we will not to 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 δ, 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:

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

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 its region; and let $\mathtt{fract}(v)$ denote the fractional part of v

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 a smallest delay what makes a visible change to P. As integer valuation would force us to introduce a case distinction we will set things so that they can be avoided.

In order not to have precise time information on transitions we introduce a new alphabet

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

and three new kinds of transitions.

Choose a valuation v among these with $\mathtt{reg}(v) \neq \mathcal{I}_\infty$ with a maximal $\mathtt{fract}(v)$. A transition on a letter $(\mathtt{delay},\varepsilon)$ will make the time pass so that the valuation v goes to the next interval region and other valuations do not change their regions:

> $P \xrightarrow{(\mathtt{delay},\varepsilon)} P'$ 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 $\mathtt{fract}(v) \neq \mathtt{fract}(v')$ then the value $v'+t_1+t_2$ is in the same region as v'.

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.

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

$P \xrightarrow{(\mathtt{delay},a)} P'$ 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 $\mathtt{fract}(v) \neq \mathtt{fract}(v')$ 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. To make the lemma true, we need some additional assumption:

Lemma 5 *There is a non Zeno accepting sequence in $\mathcal{S}(\mathcal{A})$:*

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

iff there is an accepting sequence

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

where $\sigma_0, \sigma_1, \dots \in \overline{\Sigma}$ and $(\mathtt{delay}, \cdot)$ 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. Below we work with finite words of the form $\Lambda_I^* \cdot \Lambda_\infty$ where $\Lambda_I = \mathcal{P}(Q \times \mathtt{reg}_I)$ and $\Lambda_\infty = \mathcal{P}(Q) \times \{\infty\})$.

Definition 6 For a state P of $\mathcal{T}$ 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 $\langle q, \mathtt{reg}(v), \mathtt{fract}(v)\rangle$ if $v \leq d_{max}$ (this yields a finite set of triples)
- sort all these triples w.r.t. $\mathtt{fract}(v)$ (this yields a finite sequence of triples)
- group together triples having the same value of $\mathtt{fract}(v)$ (this yields a finite sequence of finite sets of triples)
- forget about $\mathtt{fract}(v)$, i.e., replace each triple $\langle q, \mathtt{reg}(v), \mathtt{fract}(v)\rangle$ by a pair $(q, \mathtt{reg}(v))$ (this yields a finite sequence of finite sets of pairs, a word in Λ_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 $\mathcal{H}(\mathcal{A})$. It has $\Lambda_I^* \times \Lambda_\infty$ as states, and for every letter $\alpha \in \overline{\Sigma}$ 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 $H(P) = c$, $H(P') = c'$.

Lemma 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 8 We say that a path in $\mathcal{H}(\mathcal{A})$ is *good*, if it passes through infinitely many transitions labeled by letters $(\mathtt{delay}, \cdot)$.

Corollary 9 *$\mathcal{A}$ 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_-.*

Proof
$\mathcal{A}$ accepts a non Zeno word iff there is a path in $\mathcal{S}(\mathcal{A})$ satisfying the acceptance condition. By Lemma 5 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 7 shows that this is equivalent to having a good path in $H(P)$. $\square$

In the rest of this section we give a more explicit characterization of transitions in $\mathcal{H}(\mathcal{A})$.

Lemma 10 *Consider a state $(\lambda_1 \dots \lambda_k, \lambda_\infty)$ of $\mathcal{H}(\mathcal{A})$. If $k = 0$ then there is no $(\mathtt{delay}, \cdot)$ 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 $(\mathtt{delay}, \cdot)$:*

$$(\lambda_1 \dots \lambda_k, \lambda_\infty) \xrightarrow{(\mathtt{delay},\epsilon)} (\lambda'_k \lambda_1 \dots \lambda_{k-1}, \lambda'_\infty) \qquad \text{if } \lambda'_k \neq \emptyset$$
$$(\lambda_1 \dots \lambda_k, \lambda_\infty) \xrightarrow{(\mathtt{delay},\epsilon)} (\lambda_1 \dots \lambda_{k-1}, \lambda'_\infty) \qquad \text{otherwise}$$

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

$$(q_1, \mathtt{nop}) \wedge \dots \wedge (q_l, \mathtt{nop}) \wedge (q'_1, \mathtt{reset}) \wedge \dots \wedge (q'_m, \mathtt{reset})$$

this choice gives 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

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

Lemma 11 *In $\mathcal{H}(\mathcal{A})$ transitions on an action a have the form:*

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

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

Note that neither δ nor any of λ'_i may be empty. Observe also that there are as many transitions $\xrightarrow{a}$ from λ as there are choices of different conjuncts for each pair (q, r) in λ.

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

Lemma 12 *In $\mathcal{H}(\mathcal{A})$ the transitions on an action $(\mathtt{delay}, a)$ have the form:*

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

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

- *First, we change regions in λ_k. Every pair $(q, \mathcal{I}_d) \in \lambda_k$ becomes $(q, \{d\})$. Let us denote the result by λ^1_k.*
- *For $i = 1, \dots, k, \infty$ we take λ'_i, δ'_i such that: $\lambda^1_k \xrightarrow{a}_{\mathcal{A}} (\lambda'_k, \delta'_k)$ and $\lambda_i \xrightarrow{a}_{\mathcal{A}} (\lambda'_i, \delta'_i)$ for $i \neq k$.*
- *We increase again regions in λ'_k: from $\{d\}$ they become $\mathcal{I}_{d+1}$, or $\mathcal{I}_\infty$ if $d = d_{max}$.*
- *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 $c \to c'$, $c \xrightarrow{(\mathtt{delay},\cdot)} c'$, $c \twoheadrightarrow c'$, $c \overset{\Sigma^*}{\twoheadrightarrow} c'$ to denote that we may go from a configuration c to c' using one transition, one transition reading a letter of the form $(\mathtt{delay}, \cdot)$, any number of transitions or any number of transitions reading only letters from Σ, respectively.

2.2 Finding a good path in $\mathcal{H}(\mathcal{A})$.

By the Corollary 9, 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 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.

2.3 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.

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

The order we need is a relation $\preceq$ over configurations of $\mathcal{H}(\mathcal{A})$. We say that $(\lambda_1 \dots \lambda_k, \lambda_\infty) \preceq (\lambda'_1 \dots \lambda'_{k'}, \lambda'_\infty)$ 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. The following lemma follows by a standard application of Highman's lemma.

Lemma 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 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 11 it follows that there is $c'_2 \preceq c_2$ such that $c_2 \xrightarrow{a} c'_2$. When $c_1 \xrightarrow{(\mathtt{delay},\varepsilon)} c_2$ we have two cases depending on the relation between one before last element of the two configurations. To be more precise, suppose that $c_1 = (\lambda_1 \dots \lambda_k, \lambda_\infty)$ and $c'_1 = (\lambda'_1 \dots \lambda'_{k'}, \lambda'_\infty)$. If $\lambda'_{k'} \subseteq \lambda_k$ then we may do $(\mathtt{delay}, \epsilon)$ 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 $(\mathtt{delay}, a)$: either we match it with $(\mathtt{delay}, a)$ 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 $(\mathtt{delay}, \cdot)$ 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 $c'_2 = (\lambda'_1 \dots \lambda'_{k'}, \lambda'_\infty)$ and $c_2 = (\lambda_1 \dots \lambda_k, \lambda_\infty)$. Because $c'_2 \preceq c_2$, we know that $\lambda'_{k'}$ is covered by some λ_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 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 $(\mathtt{delay}, \cdot)$ actions after c_2. If $(\mathtt{delay}, \cdot)$ action is done from c_2 and $i = k$ then it is matched by a $(\mathtt{delay}, \cdot)$ 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 λ_k, hence λ_i covering $\lambda'_{k'}$ gets closer to the end of the sequence. Repeating this argument, we get that the covering λ_i finally becomes the last element and the previous case applies. □

Corollary 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 16 *Let X be a downward closed set in $\mathcal{H}(\mathcal{A})$. It is decidable if from a given configuration one can reach a configuration in 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 degree of every node is finite, t is a finite tree. Then we check if in t there is 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), it is also some in t. Let c be such configuration reachable from initial configuration of $\mathcal{H}(\mathcal{A})$ by a path π of the shortest length. Assume that c is not in t, i.e. there are two nodes on π, 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 14, there exists $c' \preceq c$ that may be reached from c_1 and the path from c_1 to c' will be no longer than that from c_2 to c. So the path leading to c' from the initial configuration is strictly shorter than π. 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_-. A contradiction. □

2.4 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 need to use in an essential way a very special form of transitions our systems have.

We write $X\!\uparrow$ for an upward closure of set X:

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

Observe that by Lemma 14 the complement of $\widehat{G}$ is upward closed.

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

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

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

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

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 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 $i = -1, 0, 1 \ldots$. For $i = -1$ it is obvious. Assume by contrary, 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 14 we know that a good infinite computation exists also from c'. This computation may first read some letters from Σ, but finally it has to read a letter $(\mathtt{delay}, \cdot)$, 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 $(\mathtt{delay}, \cdot)$. Easy induction on k shows that $c \in Z_k$. $\square$

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 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 $pre^{\forall}_{\Sigma^*}(pre^{\forall}_{\mathtt{delay}}(X))$ 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 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 19 *For every upper-closed set X, the membership in $pre^{\forall}_{\Sigma^*}(pre^{\forall}_{\texttt{delay}}(X))$ is decidable.*

Together Lemmas 18 and 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. However Lemma 18 is substantially more complicated while Lemma 19 we get as a side effect. 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 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 Λ_I).

On compressed configurations we introduce an expansion operation parametrized by words from Λ_I^*.

Definition 21 A compressed configuration $\widehat{c} = (\lambda_1 \dots \lambda_l, f, \lambda_\infty)$ may be expanded in a context of some word $\lambda^0_1 \dots \lambda^0_k \in \Lambda^*_I$, resulting in the set of configurations $(\lambda_1 \dots \lambda_l \lambda'_{l+1} \dots \lambda'_{l+k}, \lambda_\infty)$ such that $\lambda'_{l+i} \in f(\lambda^0_i)$ for $1 \leq i \leq k$. We will use $exp(\widehat{c}, \lambda^0_1 \dots \lambda^0_k)$ to denote the set of obtained configurations. Similarly, if $\widehat{C}$ is a set of compressed configurations we write $Exp(\widehat{C}, \lambda^0_1 \dots \lambda^0_k)$ for $\bigcup\{exp(\widehat{c}, \lambda^0_1 \dots \lambda^0_k) : \widehat{c} \in \widehat{C}\}$.

Observe that the value $f(\lambda)$ for λ not appearing in $\lambda^0_1 \dots \lambda^0_k$ does not matter; moreover if some $f(\lambda^0_i) = \emptyset$ 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. However, due to nondeterminism, there is unbounded number of successors. To see this suppose that there is more than one choice of transition on action a form a letter λ then every occurrence of λ in a configuration may make a choice independently.

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 $\preceq$ relation. This is because $(\texttt{delay}, \cdot)$ action treats the one before last element of a configuration in a special way. So if something is inserted after λ_k in $(\lambda_1 \dots \lambda_k, \lambda_\infty)$ then the $\texttt{delay}$ operation uses this inserted element instead of λ_k. As a side remark let us mention that using the upward closure of $pre^{\forall}_{\mathtt{delay}}(Z_{i-1}\uparrow)$ in the definition of Z_i would be incorrect (Lemma 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 $pre^{\forall}_{\mathtt{delay}}(X)$ 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 $c_1 \xrightarrow{(\mathtt{delay},\cdot)} c_2$ then also $c'_1 \xrightarrow{(\mathtt{delay},\cdot)} c'_2$ with some $c'_2 \preceq c_2$. Hence, if $c_1 \notin pre^{\forall}_{\mathtt{delay}}(X)$ then $c'_1 \notin pre^{\forall}_{\mathtt{delay}}(X)$.

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 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 λ_k and λ_∞) such that:*

- *if $c_0 \xrightarrow{(\mathtt{delay},\cdot)} c$ 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 $c_0 \xrightarrow{(\mathtt{delay},\cdot)} c'$ for some $c' \preceq c$.*

Proof
The transition on $(\mathtt{delay}, \epsilon)$ is deterministic. If $c_0 \xrightarrow{(\mathtt{delay},\epsilon)} c'$ 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 λ_k. In the first case we add $\widehat{c} = (\lambda'_k, \mathbf{sgl}, \lambda'_\infty)$ into $\widehat{C}(\lambda_k, \lambda_\infty)$, 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 $(\mathtt{delay}, a)$. 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 $\lambda \xrightarrow{a}_{\mathcal{A}} (\lambda', \delta')$; intuitively these are allowed transitions from $\lambda_1, \dots, \lambda_{k-1}$. We also fix transitions $\lambda^1_k \xrightarrow{a}_{\mathcal{A}} (\lambda'_k, \delta'_k)$ and $\lambda_\infty \xrightarrow{a}_{\mathcal{A}} (\lambda'_\infty, \delta'_\infty)$ (where λ^1_k is λ_k with increased regions as in Lemma 12). This choice of transitions gives us a compressed configuration $\widehat{c} = (\delta, f, \lambda''_\infty)$ where

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

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

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

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

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

Lemma 23 *For a given $\widehat{C}$ and a set X upward closed with respect to relation $\preceq_r$ there exists a constant $B(X, \widehat{C})$ (and we may compute it) such that if for some $\lambda^0_1 \dots \lambda^0_k$:*

$$Exp(\widehat{C}, \lambda^0_1 \dots \lambda^0_k) \subseteq X$$

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

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

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/λ 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. Form the pumping lemma it follows that if the word $\lambda^0_1 \dots \lambda^0_k$ 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 $\lambda^0_{i_1} \dots \lambda^0_{i_m}$ of length $m \le B(X, \{\widehat{c}\})$, such that $exp(\widehat{c}, \lambda^0_{i_1} \dots \lambda^0_{i_m}) \subseteq$

X. We take all the elements from all these subsequences, getting a subsequence of length $\leq B(X, \widehat{C}) := \sum_{\widehat{c} \in \widehat{C}} B(X, \{\widehat{c}\})$ such that all the inclusions hold. $\square$

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

Lemma 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 $pre^{\forall}_{\mathtt{delay}}(X\uparrow)$ is bounded by $M_{delay}(X\uparrow)$.*

Proof
There are only finitely many different $\widehat{C}(\lambda_k, \lambda_\infty)$ as constructed in Lemma 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 $pre^{\forall}_{\mathtt{delay}}(X\uparrow)$. Take the set $\widehat{C}(\lambda^0_k, \lambda^0_\infty)$ as given by Lemma 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 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 22 we get that $(\lambda'_1 \dots \lambda'_l \lambda^0_k, \lambda^0_\infty) \in pre^{\forall}_{\mathtt{delay}}(X\uparrow)$. By the minimality of c_0, we have that $c_0 = (\lambda'_1 \dots \lambda'_l \lambda^0_k, \lambda^0_\infty)$, so its length is bounded by $M_{delay} + 2$. $\square$

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 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 letter $a \in \Sigma$ 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 $c_0 \in exp(\widehat{c}_0, \lambda^0_1 \dots \lambda^0_k)$.*

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 $\lambda \xrightarrow{a}_{\mathcal{A}} (\lambda', \delta')$; intuitively these are allowed transitions from $\lambda \in f(\lambda^0_i)$. We also fix transitions $\lambda_i \xrightarrow{a}_{\mathcal{A}} (\lambda'_i, \delta'_i)$ for $i = 1, \dots, l, \infty$. This choice of transitions gives us a compressed configuration $\widehat{c} = (\delta \lambda'_1 \dots \lambda'_l, f', \lambda'_\infty)$ where

$$\delta = \bigcup_{i=1\dots,l,\infty} \delta'_i \cup \bigcup \{\delta' : (\lambda \xrightarrow{a}_{\mathcal{A}} (\lambda', \delta')) \in \mathcal{T}, \lambda \in \Lambda_I\}$$
$$f'(\lambda^0) = \{\lambda' : (\lambda \xrightarrow{a}_{\mathcal{A}} (\lambda', \delta')) \in \mathcal{T}, \lambda \in f(\lambda^0)\}$$

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

For the first statement of the lemma, take $c_0 \in exp(\widehat{c}_0, \lambda_1^0 \dots \lambda_k^0)$ 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 $c \in exp(\widehat{c}, \lambda_1^0 \dots \lambda_k^0)$. 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 $(\delta\lambda_1' \dots \lambda_l', f', \lambda_\infty')$. According to the above, it was constructed from $\widehat{c}_0$ using some transitions $\lambda_i \xrightarrow{a}_{\mathcal{A}} (\lambda_i', \delta_i')$ for $i = 1, \dots, l, \infty$ and some set of transitions $\mathcal{T}$. Take $c \in exp(\widehat{c}, \lambda_1^0 \dots \lambda_k^0)$. We have that c is of the form $(\delta'\lambda_1' \dots \lambda_l', \lambda_{l+1}' \dots \lambda_{l+k}', \lambda_\infty')$ where $\lambda_1', \dots \lambda_l'$ are as in $\widehat{c}$ and for $i = 1, \dots, k$ we can choose from $\mathcal{T}$ transitions $\lambda_{l+i} \xrightarrow{a}_{\mathcal{A}} (\lambda_{l+i}', \delta_{l+i}')$ such that $\lambda_{l+i} \in f(\lambda_i^0)$. 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 $c_0 \in exp(\widehat{c}_0, \lambda_1^0 \dots \lambda_k^0)$ by the definition of expansion. Let $c' = (\delta'\lambda_1' \dots \lambda_{l+k}', \lambda_\infty')$ with $\delta' = \bigcup_{i=1,\dots,l+k,\infty} \delta_i'$. Observe that δ' may be a proper subset of δ if not all transitions from $\mathcal{T}$ has 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) and moreover that size of this portion is bounded by a constant.

Lemma 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 $\widehat{c}_0 = (\epsilon, \mathbf{sgl}, \lambda_\infty)$. We define a set $\widehat{C}$ of compressed configurations as a closure of $\{\widehat{c}_0\}$ on the operation defined in Lemma 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 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 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 27 *There is an algorithm that given Y upper closed with respect to $\preceq_r$ 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_{Σ^*} be the maximal possible value of $B(Y, \widehat{C}(\lambda_\infty))$.

Suppose $c_0 = (\lambda^0_1 \dots \lambda^0_k, \lambda^0_\infty)$ is a minimal element of $pre^{\forall}_{\Sigma^*}(Y)$. Take the set $\widehat{C}(\lambda^0_\infty)$ as given by Lemma 26. We have that $Exp(\widehat{C}_{\Sigma^*}(\lambda^0_\infty), \lambda^0_1 \dots \lambda^0_k) \subseteq Y$ by the second statement of this lemma. From Lemma 23 we get a subsequence $\lambda'_1 \dots \lambda'_l$ of $\lambda^0_1 \dots \lambda^0_k$ whose length is bounded by $B(X, \widehat{C}(\lambda^0_\infty)) \leq M_{\Sigma^*}$ and such that $Exp(\widehat{C}_{\Sigma^*}(\lambda^0_\infty), \lambda'_1 \dots \lambda'_l) \subseteq Y$. By the first statement of Lemma 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 18 and 19 consists of two simple observations.

Lemma 28 *For every set X upward closed with respect to $\preceq$ relation, the membership in $Y = pre^{\forall}_{\mathtt{delay}}(X)$ 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 10 and 12. The second part follows from the property that we have already noticed before (page 15): if $c'_1 \preceq_r c_1$ and $c_1 \xrightarrow{(\mathtt{delay},\cdot)} c_2$ then also $c'_1 \xrightarrow{(\mathtt{delay},\cdot)} c'_2$ with some $c'_2 \preceq c_2$. □

Lemma 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 $c \overset{\Sigma^*}{\twoheadrightarrow} c'$. Choose $c' \notin Y$ so that the length of a derivation $c \overset{\Sigma^*}{\twoheadrightarrow} c'$ is the smallest possible. We show that $c' \in t$. Recall that Lemma 11 characterizes transitions on letters. Directly form 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 $c \overset{\Sigma^*}{\twoheadrightarrow} d'$ is shorter than $c \overset{\Sigma^*}{\twoheadrightarrow} c'$. This is impossible by the choice of c'. $\square$

Proof (of Lemma 18)
Take an upward closed set X. By Lemma 24 we can compute a constant $M_{\mathtt{delay}}$ that bounds the size of minimal elements in $Y = pre^{\forall}_{\mathtt{delay}}(X)$. Using Lemma 28 we can find the minimal elements of Y by enumerating all configurations of size bounded by $M_{\mathtt{delay}}$. Observe that Y is $\preceq_r$ upward closed.

Once we have computed Y, Lemma 27 gives us a constant $M_{\Sigma^*}(Y)$ bounding the size of minimal elements in $pre^{\forall}_{\Sigma^*}(Y) = pre^{\forall}_{\Sigma^*}(pre^{\forall}_{\mathtt{delay}}(X))$.

$\square$

Proof (of Lemma 19)
We first compute the set $Y = pre^{\forall}_{\mathtt{delay}}(X)$ as described above. We can then use Lemma 29 to test for the membership in $pre^{\forall}_{\Sigma^*}(Y) = pre^{\forall}_{\Sigma^*}(pre^{\forall}_{\mathtt{delay}}(X))$.

$\square$

3 Constrained TPTL

In this section we discuss 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 [11, 26].

TPTL[10] 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:

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

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 $w = (a_1, t_1)(a_2, t_2)\dots$ 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 $a_i = p$
$w, i, v \vDash x \sim c$	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 $\exists_{j>i}\ (w, j, v \vDash \alpha$ and $\forall_{k\in(i,j)}\ w, k, v \vDash \beta)$
$w, i, v \vDash \alpha \widetilde{\mathbb{U}} \beta$	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:

$$F\alpha \equiv tt\mathbb{U}\alpha \qquad A\alpha \equiv f\!f\widetilde{\mathbb{U}}\alpha$$

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

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

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 [12] where it is also shown that the following TPTL^1 formula is not expressible in MTL:

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

The satisfiability problem over infinite timed sequences is undecidable for MTL [23], 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 automata where all states are accepting. The set of *positive formulas* is given by the following grammar:

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

The set of formulas of *Constrained TPTL* is:

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

Observe that the formula (1) belongs to the positive fragment if we add redundant $(x \leq 2)$ after b.

Theorem 30 *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.*

Proof
It is enough to give a translation from formulas to automata in the class from Theorem 2. The translation is on the syntax of the formula.

We start for the automaton for the positive formulas. The states of an automaton for a formula will be all subformulas of the formula. A state associated to a formula α 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.

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

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

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

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

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

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 ε over transitions. Transition $\xrightarrow{*}$ is just a shortcut for transitions on all letters of the alphabet. Transitions $\xrightarrow{\varepsilon}$ and $\xrightarrow[x:=0]{\varepsilon}$ can be seen as eager ε-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

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

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 β is of the form $(x < c) \wedge \beta'$, or possibly with non strict inequality. As we consider only non Zeno words, this assures that the language accepted from this state is correct even if the state is accepting. One way to look at them is as eager

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.

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 can be translated into Constrained TPTL. $\square$

3.1 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 have also the locality property (cf. [24]). The satisfiability problem for both logics is non-elementary.

Using equivalences mentioned above FlatMTL[13] with pointwise non Zeno semantics can be defined as a set of formulas of the grammar:

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

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.

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 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 [23] we get undecidability of the positive fragment extended with a formula expressing that some action appears infinitely often. Theorem 32 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 loosing decidability.

4 Undecidability without testing for equality

Ouaknine and Worell [23] have proved undecidability of MTL over infinite words in the case of pointed semantics. Their construction immediately implies that the decidability result from the last section is optimal if classes of accepting conditions are concerned.

Theorem 31 (Ouaknine, Worell) *It is undecidable whether a given one-clock 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 the satisfiability problem for MITL over infinite words is decidable [7].

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 it is only allowed to test if the clock is bigger than 1.

Theorem 32 *It is undecidable whether a given one-clock alternating timed automata $\mathcal{A}$ 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.*

To prove Theorem 32 we code a problem of deciding whether there is a successful run of a counter machine with insertion errors:

Definition 33 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 δ 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 δ 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 its configurations there appears state from Q_{acc}.

Theorem 34 (Ouaknine, Worell [23]) *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 34 by reduction to boundedness of a lossy 4-counter machine. The principle of *lossy k-counter machine* is similar to that with insertion errors, but for the fact 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.

Theorem 35 (Mayr [19]) *It is undecidable whether every run of a given lossy 4-counter machine $\mathcal{M}^l$ is bounded.*

Proof
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 will be 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 form $(q_0, 0, 0, 0, 0)$, 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 would have some auxiliary states, one of them would be 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 would increase c^5 by 1 and then (in a nondeterministic way) increase 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 successful computation of $\mathcal{M}^g$. 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 $\mathcal{M}^l$. $\square$

Now we return to the proof of Theorem 32. 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 successful run. 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_∞ and q_- will encode the condition on successful runs. State \$ is important for technical reasons. State q_{init} is the initial state that will not be reachable from other states.

In our description below we will consider a characterisation of acceptance given by Lemma 4. In this presentation a run of $\mathcal{A}$ is a sequence

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

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. Compared with Lemma 4 we have joined together a transition letting the time pass with an action transition and write just $\overset{a,t}{\hookrightarrow}$ transitions. We will use only two regions: $\mathcal{I}_1 = [0,1]$ and $\mathcal{I}_\infty = (1,\infty)$.

Definition 36 A configuration θ of $\mathcal{A}$ is *well-formed* if:

- For every $(q,v) \in \theta$: 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 $\sigma \in \Sigma$ with $(\sigma, v) \in \theta$.
- In θ there is exactly one pair with a state from $Q_{\mathcal{M}}$, exactly one pair with the state q_∞, and no pairs with q_{init};
- Suppose (q,v) is in θ where $q \in \{1,\ldots,5\}$. Then this pair is immediately preceded by some $(\$, v')$ (there is no pair (q'', v'') in θ with $v' < v'' < v$).

Intuitively, a well formed 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^i is equal to the number of i 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. There will be always one state from $Q_{\mathcal{M}}$ representing the state of the simulated machine. State q_∞ plus some number of states q_- will be there to encode a condition on a successful run.

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

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 σ then all states in its current configuration have to make a transition according to some rule labeled σ. In consequence, if there is a state in the configuration that does not have a rule for σ then the automaton cannot read σ.

The automaton starts in the state q_{init} and waits at least one time unit to start its two copies: one is a state q_0 and another is q_∞ (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 configuration becomes $\{(q_0, v), (q_\infty, v')\}$ with $v, v' \in \mathcal{I}_\infty$.

States \$ for time ≤ 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 appropriate 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 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 a 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 configuration. For $\sigma = (q : \texttt{if}\ c^i > 0\ \texttt{then}\ c^i := c^i - 1; \texttt{goto}\ q')$ we do:

$$\begin{array}{c}
q, \mathcal{I}_\infty \xrightarrow{\sigma} q' \\
i, \mathcal{I}_\infty \xrightarrow{\sigma} \top
\end{array}$$

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_∞ and q_-, that are used to ensure that states from Q_{acc} appear infinitely often. The state q_∞ 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. The states q_- disappear, when there there is a transition ending in state from Q_{acc}:

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

Lemma 37 *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 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 $\mathcal{M}^g$ does a transition $\sigma = (q : \texttt{if } c^i = 0 \texttt{ then goto } q')$ we may do the same in $\mathcal{A}$ reading letter σ. When $\mathcal{M}^g$ does $\sigma = (q : c^i := c^i + 1; \texttt{goto } q')$, we do the same reading letter σ and then `sh$`. It is easy to check, that after each step the resulting configuration remains well formed.

The most difficult 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 configuration θ. Let us look at the biggest valuation $v \leq 1$ appearing in θ. By the conditions of well-formedness there is exactly one state $q \in Q_+$ such that $(q, v) \in \theta$. 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 σ. Otherwise, it does `sh$` or `shc` followed by `sh$` that has an effect of putting \$ or \$ followed by q at the beginning of the configuration. After this we obtain a well formed 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 θ.

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 some accepting run of $\mathcal{A}$ on some word. In the first step, $\mathcal{A}$ has to reach a well-formed configuration corresponding to the initial configuration of $\mathcal{M}^g$. Let us see what may happen from any well-formed configuration. Suppose that time passes and timer for some states $1, \ldots, 5, \$$ becomes greater than 1. If it happens simultaneously for state \$ and some state i, then from the obtained 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 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 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 configuration.

Transition reading shc or sh\$ when no state of $1, \dots, 5, \$$ has timer above 1 do not change the configuration. Transition reading new, has to be followed by transition sh\$ and we get a well-formed configuration with one of the counters increased. Transition reading $\sigma = (q : \texttt{if } c^i = 0 \texttt{ then goto } q')$ is possible only when counter c^i is zero. After 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 configuration that corresponds to a correct configuration of $\mathcal{M}^g$. Transition reading $\sigma = (q : \texttt{if } c^i > 0 \texttt{ then } c^i := c^i - 1; \texttt{goto } q')$ gives us always a well-formed configuration. The obtained 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 argumentation gives some (finite or infinite) computation of $\mathcal{M}^g$. As $\mathcal{A}$ accepts, every q_- disappears after some time. 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, we have an infinite computation of $\mathcal{M}^g$ satisfying the Büchi condition. $\square$

References

[1] P. Abdulla and B. Jonsson. Veryfying networks of timed processes. In *Proc. TACAS'98*, 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] S. Adams, J. Ouaknine, and J. Worrell. Undecidability of universality for timed automata with minimal resources. In *FORMATS*, volume 4763 of *LNCS*, pages 25–37, 2007.

[6] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

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

[8] R. Alur, L. Fix, and T. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 204, 1997.

[9] R. Alur and T. A. Henzinger. A really temporal logic. In *FOCS*, pages 164–169, 1989.

[10] R. Alur and T. A. Henzinger. A really temporal logic. *J. ACM*, 41(1):181–204, 1994.

[11] 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.

[12] P. Bouyer, F. Chevalier, and N. Markey. On the expressiveness of tptl and mtl. In *FSTTCS*, volume 3821 of *LNCS*, pages 432–443, 2005.

[13] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. The cost of punctuality. In *LICS*, pages 109–120, 2007.

[14] 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.

[15] Y. Hirshfeld and A. M. Rabinovich. Logics for real time: Decidability and complexity. *Fundam. Inform.*, 62(1):1–28, 2004.

[16] 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.

[17] S. Lasota and I. Walukiewicz. Alternating timed automata. In *FOSSACS'05*, number 3441 in Lecture Notes in Computer Science, pages 250–265, 2005. Journal version available from WWW.

[18] S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Trans. Comput. Log.*, 9(2), 2008.

[19] R. Mayr. Undecidable problems in unreliable computations. *TCS*, 1-3(297):337–354, 2003.

[20] A. W. Mostowski. Hierarchies of weak automata and week monadic formulas. *Theoretical Computer Science*, 83:323–335, 1991.

[21] F. Murlak. Weak index versus borel rank. In *STACS*, Dagstuhl Seminar Proceedings, pages 573–584. Dagsr, 2008.

[22] 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.

[23] J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *LICS*, pages 188–197, 2005.

[24] J. Ouaknine and J. Worrell. Safety metric temporal logic is fully decidable. In *TACAS*, 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*, 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.