Alternating Timed Automata *

Sławomir Lasota[1]** and Igor Walukiewicz[2]

[1] Institute of Informatics, Warsaw University
Banacha 2, 02-097 Warszawa

[2] LaBRI, Université Bordeaux-1
351, Cours de la Libération, F-33 405, Talence cedex, France

Abstract. A notion of alternating timed automata is proposed. It is shown that such automata with only one clock have decidable emptiness problem. This gives a new class of timed languages which is closed under boolean operations and which has an effective presentation. We prove that the complexity of the emptiness problem for alternating timed automata with one clock is non-primitive recursive. The proof gives also the same lower bound for the universality problem for nondeterministic timed automata with one clock thereby answering a question asked in a recent paper by Ouaknine and Worrell.

1 Introduction

Timed automata is a widely studied model of real-time systems. It is obtained from finite nondeterministic automata by adding clocks which can be reset and whose values can be compared with constants. In this paper we consider alternating version of timed automata obtained by introducing universal transitions in the same way as it is done for standard nondeterministic automata. From the results of Alur and Dill [2] it follows that such a model cannot have decidable emptiness problem as the universality problem for timed automata is not decidable. In the recent paper [16] Ouaknine and Worrell has shown that the universality problem is decidable for nondeterministic automata with one clock. Inspired by their construction, we show that the emptiness problem for alternating timed automata with one clock is decidable as well. We also prove not primitive recursive lower bound for the problem. The proof implies the same bound for the universality problem for nondeterministic timed automata with one clock. This answers the question posed by Ouaknine and Worrell [16]. To complete the picture we also show that an extension of our model with epsilon-transitions has undecidable emptiness problem.

The crucial property of timed automata models is the decidability of the emptiness problem. The drawback of the model is that the class of languages recognized by timed automata is not closed under complement and the universality

* Work reported here has been partially supported by the European Community Research Training Network GAMES.

** Partially supported by the Polish KBN grant No. 4 T11C 042 25. This work was performed in part during the author's stay at LaBRI, Université Bordeaux-1.

question is undecidable (Π_1^1-hard) [2]. One solution to this problem is to restrict to deterministic timed automata. Another, is to restrict the reset operation; this gives the event-clock automata model [4]. A different ad-hoc solution could be to take the boolean closure of the languages recognized by timed automata. This solution does not seem promising due to the complexity of the universality problem. This consideration leads to the idea of using automata with one clock for which the universality problem is decidable. The obtained class of alternating timed automata is by definition closed under boolean operations. Moreover, using the method of Ouaknine and Worrell, we can show that the class has decidable emptiness problem. As it can be expected, there are languages recognizable by timed automata that are not recognizable by alternating timed automata with one clock. More interestingly, the converse is also true: there are languages recognizable by alternating timed automata with one clock that are not recognizable by nondeterministic timed automata with any number of clocks.

Once the decidability of the emptiness problem for alternating timed automata with one clock is shown, the next natural question is the complexity of the problem. We show a non-primitive recursive lower bound. For this we give a reduction of the reachability problem for lossy channel systems [17]. The reduction shows that the lower bound holds also for purely universal alternating timed automata. This implies non-primitive recursive lower bound for the universality problem for nondeterministic timed automata with one clock. We also point out that allowing epsilon transitions in our model permits to code perfect channel systems and hence makes the emptiness problem undecidable.

Related work Our work is strongly inspired by the results of Ouaknine and Worrell [16]. Except for [11], it seems that the notion of alternation in the context of timed automata was not studied before. The reason was probably undecidability of the universality problem. Some research (see [5, 10, 7, 3, 6] and references within) was devoted to the control problem in the timed case. While in this case one also needs to deal with some universal branching, these works do not seem to have direct connection to our setting. Finally, let us mention that restrictions to one clock have been already considered in the context of model-checking of timed systems [12, 15].

Organization of the paper In the next section we define alternating timed automata; we discuss their basic properties and relations with nondeterministic timed automata. In Section 3 we show decidability of the emptiness problem for alternating timed automata with one clock. In the following section we show a non-primitive recursive lower bound for the problem. Next we show the undecidability result for an extension of our model with epsilon-moves.

2 Alternating Timed Automata

In this section we introduce the alternating timed automata model and study its basic properties. The model is a quite straightforward extension of the nondeterministic model. Nevertheless some care is needed to have the desirable feature that complementation corresponds to exchanging existential and universal branchings (and final and non-final states). As can be expected, alternating timed automata can recognize more languages than their nondeterministic counterparts. The price to pay for this is that the emptiness problem becomes undecidable, in contrast to timed automata [2]. This motivates the restriction to automata with one clock. With one clock alternating automata can still recognize languages not recognizable by nondeterministic automata and moreover, as we show in the next section, they have decidable emptiness problem.

For a given finite set $\mathcal{C}$ of *clock variables* (or *clocks* in short), consider the set $\Phi(\mathcal{C})$ of clock constraints σ defined by

$$\sigma \quad ::= \quad x < c \ \mid \ x \leq c \ \mid \ \sigma_1 \wedge \sigma_2 \ \mid \ \neg\sigma,$$

where c stands for an arbitrary nonnegative integer constant, and $x \in \mathcal{C}$. For instance, note that $\mathbf{tt}$ (always true), or $x = c$, can be defined as abbreviations. Each constraint σ denotes a subset $[\sigma]$ of $(\mathbb{R}_+)^{\mathcal{C}}$, in a natural way, where $\mathbb{R}_+$ stands for the set of nonnegative reals.

Transition relation of a timed automaton [2] is usually defined by a finite set of rules δ of the form

$$\delta \subseteq Q \times \Sigma \times \Phi(\mathcal{C}) \times Q \times \mathcal{P}(\mathcal{C}),$$

where Q is a set of *locations* (control states) and Σ is an input alphabet. A rule $\langle q, a, \sigma, q', r\rangle \in \delta$ means, roughly, that when in a location q, if the next input letter is a and the constraint σ is satisfied by the current valuation of clock variables, the next location can be q' and the clocks in r should be reset to 0. Our definition below uses an easy observation, that the relation δ can be suitably rearranged into a finite partial function

$$Q \times \Sigma \times \Phi(\mathcal{C}) \dot{\to} \mathcal{P}(Q \times \mathcal{P}(\mathcal{C})).$$

The definition below comes naturally when one thinks of an element of the codomain as a disjunction of a finite number of pairs (q, r). Let $\mathcal{B}^+(X)$ denote the set of all positive boolean formulas over the set X of propositions, i.e., the set generated by:

$$\phi \quad ::= \quad X \ \mid \ \phi_1 \wedge \phi_2 \ \mid \ \phi_1 \vee \phi_2.$$

Definition 1 (Alternating timed automaton). *An* alternating timed automaton *is a tuple* $\mathcal{A} = (Q, q_0, \Sigma, \mathcal{C}, F, \delta)$ *where:* Q *is a finite set of locations,* Σ *is a finite input alphabet,* $\mathcal{C}$ *is a finite set of clock variables, and* $\delta : Q \times \Sigma \times \Phi(\mathcal{C}) \dot{\to} \mathcal{B}^+(Q \times \mathcal{P}(\mathcal{C}))$ *is a finite partial function. Moreover* $q_0 \in Q$ *is an* initial state *and* $F \subseteq Q$ *is a set of* accepting states*. We also put an additional restriction:*

(Partition) *For every* q *and* a*, the set* $\{[\sigma] : \delta(q, a, \sigma) \text{ is defined}\}$ *gives a (finite) partition of* $(\mathbb{R}_+)^{\mathcal{C}}$*.*

The (Partition) condition does not limit the expressive power of automata. We impose it because it permits to give a nice symmetric semantic for the automata. We will often write rules of the automaton in a form: $q, a, \sigma \ \mapsto \ b$.

By a *timed word* over Σ we mean a finite sequence

$$w = (a_1, t_1)(a_2, t_2) \dots (a_n, t_n) \tag{1}$$

of pairs from $\Sigma \times \mathbb{R}_+$. Each t_i describes the amount of time that passed between reading a_{i-1} and a_i, i.e., a_1 was read at time t_1, a_2 was read at time $t_1{+}t_2$, and so on. In Sections 4 and 5 it will be more convenient to use an alternative representation where t_i denotes the time elapsed since the beginning of the word.

To define an execution of an automaton, we will need two operations on valuations $\mathbf{v} \in (\mathbb{R}_+)^{\mathcal{C}}$. A valuation $\mathbf{v}{+}t$, for $t \in \mathbb{R}_+$, is obtained from $\mathbf{v}$ by augmenting value of each clock by t. A valuation $\mathbf{v}[r := 0]$, for $r \subseteq \mathcal{C}$, is obtained by reseting values of all clocks in r to zero.

For an alternating timed automaton $\mathcal{A}$ and a timed word w as in (1), 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 $(q_0, \mathbf{v}_0)$, where $\mathbf{v}_0 : \mathcal{C} \to \mathbb{R}_+$ is a valuation assigning 0 to each clock variable. It consists of n phases. The $(k{+}1)$-th phase starts in $(q_k, \mathbf{v}_k)$, ends in some configuration $(q_{k+1}, \mathbf{v}_{k+1})$ and proceeds as follows. Let $\bar{\mathbf{v}} := \mathbf{v}_k{+}t_{k+1}$. Let σ be the unique constraint such that $\bar{\mathbf{v}}$ satisfies σ and $\delta(q_k, a_{k+1}, \sigma)$ is defined. Now the outcome of the phase is determined by the formula b. There are three cases:

- $b = b_1 \wedge b_2$: Adam chooses one of subformulas b_1, b_2 and the play continues with b replaced by the chosen subformula;
- $b = b_1 \vee b_2$: dually, Eve chooses one of subformulas;
- $b = (q, r) \in Q \times \mathcal{P}(\mathcal{C})$: the phase ends with the result $(q_{k+1}, \mathbf{v}_{k+1}) := (q, \bar{\mathbf{v}}[r := 0])$. A new phase is starting from this configuration if $k{+}1 < n$.

The winner is Eve if q_n is accepting ($q_n \in F$), otherwise Adam wins.

Definition 2 (Acceptance). *The 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* $\mathcal{A}$.

To show the power of alternation we give an example of an automaton for a language not recognizable by standard (i.e. nondeterministic) timed automata (cf. [2]).

Example 1. Consider a language consisting of timed words w over a singleton alphabet $\{a\}$ that contain no pair of letters such that one of them is precisely one time unit later than the other. The alternating automaton for this language has three states q_0, q_1, q_2. State q_0 is initial. The automaton has a single clock x and the following transition rules:

$$q_0, a, \mathbf{tt} \ \mapsto \ (q_0, \emptyset) \wedge (q_1, \{x\}) \qquad q_1, a, x{\neq}1 \ \mapsto \ (q_1, \emptyset)$$
$$q_1, a, x{=}1 \ \mapsto \ (q_2, \emptyset) \qquad q_2, a, \mathbf{tt} \ \mapsto \ (q_2, \emptyset)$$

States q_0 and q_1 are accepting. Clearly, Adam has a strategy to reach q_2 iff the word is not in the language, i.e., some letter is one time unit after some other.

As one expects, we have the following:

Proposition 1. *The class of languages accepted by alternating timed automata is effectively closed under all boolean operations: union, intersection and complementation. These operations to do not increase the number of clocks of the automaton.*

The closure under conjunction and disjunction is straightforward since we permit positive boolean expressions as values of the transition function. Due to the condition (Partition) the automaton for the complement is obtained by exchanging conjunctions with disjunctions in all transitions and exchanging accepting states with non-accepting states.

Definition 3. *An alternating timed automaton $\mathcal{A}$ is called* purely universal *if the disjunction does not appear in the transition rules δ. Dually, $\mathcal{A}$ is* purely existential *if no conjunction appears in δ.*

It is obvious that every purely-existential automaton is a standard nondeterministic timed automaton. The converse is not immediately true because of the (Partition) condition. Nevertheless it is not difficult to show the following

Proposition 2. *Every standard nondeterministic automaton is equivalent to a purely-existential automaton.*

In the following sections, we consider emptiness, universality and containment for different classes of alternating timed automata. For clarity, we recall definitions here.

Definition 4. *For a class C of automata we consider three problems:*

- *Emptiness: given $\mathcal{A} \in C$ is $L(\mathcal{A})$ empty.*
- *Universality: given $\mathcal{A} \in C$ does $L(\mathcal{A})$ contain all timed words.*
- *Containment: given $\mathcal{A}, \mathcal{B} \in C$ does $L(\mathcal{A}) \subseteq L(\mathcal{B})$.*

It is well known that the universality is undecidable for non-deterministic timed automata [2] with at least two clocks. As a consequence, two other problems are also undecidable for alternating timed automata with two clocks. This is why in the rest of the paper we focus on automata with one clock only.

Proviso: In the following all automata have one clock.

Remark: The automaton from Example 1 uses only one clock. This shows that one clock alternating automata can recognize some languages not recognizable by nondeterministic automata with many clocks [2]. The converse is also true. It is enough to consider the language consisting of the words containing an appearance of a letter a at times t_1, t_2, t_1+1, t_2+1, for some $0<t_1<t_2<1$, and such that there is a at no time between t_1 and t_2 while there is one at a time between t_1+1 and t_2+1. We omit the proof.

3 Decidability

The main result of this section is that the emptiness problem for one-clock alternating timed automata is decidable. Due to closure under boolean operations, this implies the decidability of the universality and the containment problems.

Theorem 1. *Emptiness is decidable for one-clock alternating timed automata.*

Corollary 1. *The containment problem is decidable for one-clock alternating timed automata.*

The rest of this section is devoted to the proof of Theorem 1. Essentially, we have adapted the method of Ouaknine and Worrell [16] for our more general setting. We point out the differences below.

Fix a one-clock alternating timed automaton $\mathcal{A} = (Q, q_0, \Sigma, \{x\}, F, \delta)$. For readability, assume w.l.o.g. that the boolean conditions appearing in rules of δ are all in *disjunctive normal form*. In terms of acceptance games this means that each phase consists of a single move of Eve followed by a single move of Adam. Consider a labeled transition system $\mathcal{T}$ whose states are finite sets of configurations, i.e., finite sets of pairs $(q, \mathbf{v})$, where $q \in Q$ and $\mathbf{v} \in \mathbb{R}_+$. The initial position in $\mathcal{T}$ is $P_0 = \{(q_0, 0)\}$ and there is a transition $P \xrightarrow{a,t} P'$ in $\mathcal{T}$ iff P' can be obtained from P by the following nondeterministic process:

- First, for each $(q, \mathbf{v}) \in P$, do the following:
  - let $\mathbf{v}' := \mathbf{v}+t$,
  - let $b = \delta(q, a, \sigma)$ for the uniquely determined σ satisfied in $\mathbf{v}'$,
  - choose one of disjuncts of b, say
$$(q_1, r_1) \ \wedge \ \ldots \ \wedge \ (q_k, r_k) \qquad (k > 0),$$
  - let $\mathtt{Next}_{(q,\mathbf{v})} = \{(q_i, \mathbf{v}'[r_i := 0]) : i = 1 \ldots k\}$.
- Then, let $P' := \bigcup_{(q,\mathbf{v}) \in P} \mathtt{Next}_{(q,\mathbf{v})}$.

This construction is very similar to the translation from alternating to nondeterministic automata over (untimed) words: we just collect all universal choices in one set. Compared to [16], the essential difference is that we have to deal with both disjunction and conjunction, while in [16] only one of them appeared. We treat conjunction similarly to determinization in [16]. On the other hand, we leave the existential choice, i.e., nondeterminism, essentially unaffected in $\mathcal{T}$.

In what follows we will derive from $\mathcal{T}$ a finite-branching transition system $\mathcal{H}$, suitable for the decision procedure. Like in [16], the degree of the nodes of $\mathcal{H}$ will not be bounded but nevertheless finite. This is sufficient for our purposes.

A state $\{(q_1, \mathbf{v}_1), \ldots, (q_n, \mathbf{v}_n)\}$ of $\mathcal{T}$ is called *bad* iff all control states q_i are accepting ($q_i \in F$). The following proposition characterizes acceptance in $\mathcal{A}$ in terms of reachability of bad states in $\mathcal{T}$. As we consider finite words only, there are no issues concerning the quality of a strategy in the acceptance game.

Lemma 1. *$\mathcal{A}$ accepts a timed word w iff there is a path in $\mathcal{T}$, labeled by w, from P_0 to a bad state.*

Let $\widehat{\mathcal{T}}$ be a labeled transition system obtained from $\mathcal{T}$ by erasing time information from transition labels, i.e., there is a transition $P \xrightarrow{a} Q$ in $\widehat{\mathcal{T}}$ iff there is $P \xrightarrow{a,t} Q$ in $\mathcal{T}$, for some $t \in \mathbb{R}_+$. Now we cannot talk about particular timed words but still we have the following:

Lemma 2. *$L(\mathcal{A})$ is nonempty if and only if there is a path in $\widehat{\mathcal{T}}$ from P_0 to a bad state.*

Thus, the (non)emptiness problem for $\mathcal{A}$ is reduced to the reachability of a bad state in $\widehat{\mathcal{T}}$. The last difficulty is that even if each state of $\widehat{\mathcal{T}}$ is a finite set, there are uncountably many states. The following definition allows to abstract from the precise timing information in states. Let $c_{\max}$ denote the biggest constant appearing in constraints in δ. Let set $\mathtt{reg}$ of *regions* be a partition of $\mathbb{R}_+$ into $2 \cdot (c_{\max}+1)$ sets as follows:

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

For $\mathbf{v} \in \mathbb{R}_+$, let $\mathtt{reg}(\mathbf{v})$ denote its region; and let $\mathtt{fract}(\mathbf{v})$ denote the fractional part of $\mathbf{v}$. Below we work with finite words over the alphabet $\Lambda = \mathcal{P}(Q \times \mathtt{reg})$ consisting of finite sets of pairs $(q, \mathtt{r})$, where $q \in Q$ is a control state and $\mathtt{r} \in \mathtt{reg}$ is a region.

Definition 5. *For a state P of $\widehat{\mathcal{T}}$ we define a word $H(P)$ from Λ^* as the one obtained by the following procedure:*

- *replace each $(q, \mathbf{v}) \in P$ by a triple $\langle q, \mathtt{reg}(\mathbf{v}), \mathtt{fract}(\mathbf{v})\rangle$ (this yields a finite set of triples)*
- *sort all these triples w.r.t. $\mathtt{fract}(\mathbf{v})$ (this yields a finite sequence of triples)*
- *group together triples that have the same value of $\mathtt{fract}(\mathbf{v})$, ignoring multiple occurrences (this yields a finite sequence of finite sets of triples)*
- *forget about $\mathtt{fract}(\mathbf{v})$, i.e., replace each triple $\langle q, \mathtt{reg}(\mathbf{v}), \mathtt{fract}(\mathbf{v})\rangle$ by a pair $(q, \mathtt{reg}(\mathbf{v}))$ (this yields a finite sequence of finite sets of pairs, a word in Λ^*).*

Definition 6. *Define an equivalence relation $\sim$ over states of $\widehat{\mathcal{T}}$ as the kernel of H, i.e., $P \sim P'$ iff $H(P) = H(P')$.*

The following observations are straightforward:

Lemma 3. *Relation $\sim$ is a bisimulation over transition system $\widehat{\mathcal{T}}$.*

Lemma 4. *If P is bad and $P \sim P'$ then P' is bad.*

Let $\mathcal{H}$ denote the quotient of the transition system $\widehat{\mathcal{T}}$ by $\sim$. To put it more explicitly: states of $\mathcal{H}$ are all words $H(P)$, for a state P of $\widehat{\mathcal{T}}$; there is a transition $W_1 \xrightarrow{a} W_2$ in $\mathcal{H}$ if there is a transition $P_1 \xrightarrow{a} P_2$ in $\widehat{\mathcal{T}}$ with $H(P_1) = W_1$, $H(P_2) = W_2$. Since $\sim$ is a bisimulation, the definition does not depend on a particular choice of P_1 (and P_2). The initial state W_0 in $\mathcal{H}$ is $H(P_0)$.

By Lemma 4 it is correct to call a state W in $\mathcal{H}$ *bad* if $W = H(P)$ for a bad state P. Because $\mathcal{H}$ is a quotient of $\widehat{\mathcal{T}}$ by bisimulation, from Lemma 2 we derive:

Lemma 5. *$L(\mathcal{A})$ is nonempty iff a bad state is reachable in $\mathcal{H}$ from W_0.*

At this point, we have reduced emptiness of $L(\mathcal{A})$ to the reachability of a bad state in a countably infinite transition system $\mathcal{H}$. The rest of the proof is quite standard [1, 13] and exploits the fact that one can put an appropriate *well-quasi-order* (*wqo* in short) on states of $\mathcal{H}$. Unfortunately, we are obliged to redo the proofs as we could not find a theorem that fits precisely our setting.

Definition 7. *Let $\preceq$ denote the* monotone domination *ordering over Λ^* induced by the subset inclusion over Λ, defined as follows: $a_1 \dots a_n \preceq b_1 \dots b_m$ iff there exists a strictly increasing function $f : \{1, \dots, n\} \to \{1, \dots, m\}$ such that for each $i \leq n$, $a_i \subseteq b_{f(i)}$.*

Lemma 6 ([14]). *Relation $\preceq$ is a wqo, i.e., for arbitrary infinite sequence $W_1, W_2, \dots$ of words over Λ, there exist indexes $i < j$ such that $W_i \preceq W_j$.*

The decision procedure for reachability of bad states will work by an exhaustive search through a sufficiently large portion of the whole reachability tree. Thus we need to know that an arbitrarily large part of that tree can be effectively constructed. Roughly, all time delays of an action a from W can be captured by a finite number of cyclic shifts of W with an appropriate change of region.

Lemma 7. *For each state W in $\mathcal{H}$, its set of successors $\{W' \in \Lambda^* : W \xrightarrow{a} W' \text{ for some } a\}$ is finite and effectively computable.*

The following observation is proved in the same way as Lemma 15 in [16].

Lemma 8. *The inverse of $\preceq$ relation is a simulation: whenever $W_1 \preceq W_2$ and $W_2 \xrightarrow{a} W_2'$, there is some W_1' such that $W_1 \xrightarrow{a} W_1'$ and $W_1' \preceq W_2'$.*

The next observation is more specific to our setting but fortunately very easy.

Lemma 9 (Downward closedness of badness). *Whenever $W \preceq W'$ and W' is bad then W is bad as well.*

Proof. Take a letter w_i of W. We need to show that $q \in F$ for every $(q, \mathtt{r}) \in w_i$. By the definition of $W \preceq W'$ we have $w_i \subseteq w_j'$ for some letter w_j' of W'. Hence, $(q, \mathtt{r}) \in w_j'$ and $q \in F$ as W' is bad.

Now we are ready to finally prove the following:

Lemma 10. *It is decidable whether a bad state is reachable in $\mathcal{H}$ from W_0.*

Proof. The *reachability tree* is the unraveling of $\mathcal{H}$ from W_0. The algorithm constructs a portion t of the tree conforming to the following rule: do not add a node W' to t in a situation when among its ancestors there is some $W \preceq W'$. Now, Lemma 6 guarantees that each path in t is finite. Furthermore, since the degree of each node is finite, t is a finite tree.

We need only to prove that if a bad state is reachable in $\mathcal{H}$ from W_0 then t contains at least one bad state. Let W be such a bad state reachable from W_0 in $\mathcal{H}$ by a path π of the shortest length. Assume that W is not in t, i.e., there are two other nodes in π, say W_1 and W_2 such that W_1 is an ancestor of W_2 in reachability tree and $W_1 \preceq W_2$ (i.e., W_2 was *not* added into t). Since the inverse of $\preceq$ is a simulation by Lemma 8, the sequence of transitions in π from W_2 to W can be imitated by the corresponding sequence of transitions from W_1 to some other $W' \preceq W$. W' is bad as well by Lemma 9. Moreover, the path leading to W' is strictly shorter than π, a contradiction. $\square$

Theorem 1 follows immediately from Lemma 10 and Lemma 5.

Remark: In fact, Ouaknine and Worrell showed decidability of "$L(\mathcal{A}) \subseteq L(\mathcal{B})$" in a slightly more general case, namely when automaton $\mathcal{A}$ has an arbitrary many clocks. Along the same lines one can adapt our proof for "$L(\mathcal{A}) \subseteq L(\mathcal{B})$", assumed that $\mathcal{A}$ is an arbitrary nondeterministic timed automaton and $\mathcal{B}$ is a one-clock alternating timed automaton.

4 Lower Bound

In this section we prove the following lower bound result.

Theorem 2. *The complexity of the emptiness problem for one-clock purely universal alternating timed automata is not bounded by a primitive recursive function.*

Since emptiness and universality are dual in the setting of alternating automata, as a direct conclusion we get the following:

Corollary 2. *The complexity of the universality problem for one-clock purely existential alternating (i.e., nondeterministic) timed automata is not bounded by a primitive recursive function.*

This answers the question posed by Ouaknine and Worrell [16].

The rest of this section contains the proof of Theorem 2. The proof is a reduction of the reachability problem for *lossy one-channel systems* [17].

Definition 8 (Channel system). *A* channel system *is given by a tuple $\mathcal{S} = (Q, q_0, \Sigma, \Delta)$, where Q is a finite set of control states, $q_0 \in Q$ is an initial state, Σ is a finite channel alphabet and $\Delta \subseteq Q \times (\{!a : a{\in}\Sigma\} \cup \{?a : a{\in}\Sigma\} \cup \{\epsilon\}) \times Q$ is a finite set of transition rules.*

A configuration of $\mathcal{S}$ is a pair (q, w) of a control state q and a channel content $w \in \Sigma^*$. Transition rules allow the system to pass from one configuration to another. In particular, a rule $\langle q, !a, q' \rangle$ allows in a state q to write to the channel and to pass to the new state q'. Similarly, $\langle q, ?a, q' \rangle$ means reading from a channel and is allowed in state q only when a is at the end of the channel. The channel is a FIFO, and by convention $\mathcal{S}$ writes at the beginning and reads at the end. Finally, a rule $\langle q, \epsilon, q' \rangle$ allows for a silent change of control state, without reading or writing.

Formally, there is a (perfect) transition $(q, w) \xrightarrow{\gamma} (q', w')$ if one of the following conditions is satisfied:

– $\gamma = \langle q, \epsilon, q' \rangle$ and $w = w'$, or
– $\gamma = \langle q, !a, q' \rangle$ for some $a \in \Sigma$, and $w' = aw$, or
– $\gamma = \langle q, ?a, q' \rangle$ for some $a \in \Sigma$, and $w = w'a$.

The *initial configuration* is (q_0, ϵ), i.e., execution of $\mathcal{S}$ starts with the empty channel. For technical convenience, we assume w.l.o.g. that there is no rule returning back to the initial state: for each rule $\langle q, x, q' \rangle \in \Delta$, $q' \neq q_0$.

A *lossy channel system* differs from the perfect one in only one respect: during the transition step, an arbitrary number of messages stored in the channel may be lost. To define lossy transitions, we need the subsequence ordering on Σ^*, denoted by $\sqsubseteq$ (e.g., `tata` $\sqsubseteq$ `atlanta`). We say that there is a lossy transition from (q, w) to (q', w'), denoted by $(q, w) \overset{\gamma}{\Longrightarrow} (q', w')$, iff there exists $u, u' \in \Sigma^*$ such that $u \sqsubseteq w$, $(q, u) \overset{\gamma}{\longrightarrow} (q', u')$ and $w' \sqsubseteq u'$.

By a *lossy computation* of a channel system $\mathcal{S}$ we mean a finite sequence:

$$(q_0, \epsilon) \overset{\gamma_1}{\Longrightarrow} (q_1, w_1) \overset{\gamma_2}{\Longrightarrow} (q_2, w_2) \quad \ldots \quad \overset{\gamma_n}{\Longrightarrow} (q_n, w_n). \tag{2}$$

Definition 9. Lossy reachability problem *for channel systems is: given a channel system S and a configuration (q_f, w_f), with $q_f \neq q_0$, decide whether there is a lossy computation of S ending in (q_f, w_f).*

Theorem 3 ([17]). *The lossy reachability problem for channel systems has non-primitive recursive complexity.*

The result of [17] was showed for a slightly different model. Namely, during a single transition, a finite sequence of messages was allowed to be read or written to the channel. Clearly, reachability problems in both models are polynomial-time equivalent.

In the sequel we describe a reduction from the lossy reachability for channel systems to the emptiness problem for one-clock purely-universal alternating timed automata. Given a channel system $\mathcal{S} = (Q, q_0, \Sigma, \Delta)$, and a configuration (q_f, w_f), we effectively construct a purely-universal automaton $\mathcal{A}$ with a single clock x, and the input alphabet $\overline{\Sigma} = Q \cup \Sigma \cup \Delta$. The construction will assure that $\mathcal{A}$ accepts precisely correct encodings of lossy computations of $\mathcal{S}$ ending in (q_f, w_f). A computation as in (2) will be encoded as the following word over $\overline{\Sigma}$:

$$q_n \gamma_n w_n \quad q_{n-1} \gamma_{n-1} w_{n-1} \quad \ldots \quad q_1 \gamma_1 w_1 \quad q_0, \tag{3}$$

where $q_i \in Q$, $\gamma_i \in \Delta$, $w_i \in \Sigma^*$. Let $\mathcal{S}$ be fixed in this section.

It will be convenient here to write timed words in a slightly different way than before. From now on, whenever we write a word $w = (a_1, t_1)(a_2, t_2) \ldots (a_n, t_n)$ we mean that the letter a_i appeared t_i time units after the beginning of the word. In particular, a_{i+1} appeared $t_{i+1} - t_i$ time units after a_i. Clearly this is correct only when $t_{i+1} \geq t_i$, for $i = 1 \ldots n{-}1$.

Before the formal definition of encoding of a computation by a timed word we outline shortly the underlying intuition. We will require that the letter q_n appears at time 0 and then that each letter q_i appears at time $n - i$. Hence, each configuration will be placed in a unit interval. To ensure consistency of the channel contents at consecutive configurations we require that if a message survived during a step i (it was neither read nor written nor lost) then the distance in time between its appearances in the sequences w_i and w_{i-1} should be precisely 1.

We will need a new piece of notation : by $(w+1)$ we mean the word obtained from w by increasing all t_i by one time unit, i.e., $(w+1) = (a_1, t_1+1)(a_2, t_2+1)\dots(a_n, t_n+1)$.

Definition 10. *By a* lossy computation encoding ending in (q_f, w_f) *we mean any timed word over* $\overline{\Sigma}$ *of the form:*

$$(q_n, t_n)(\gamma_n, t'_n)v_n\ (q_{n-1}, t_{n-1})(\gamma_{n-1}, t'_{n-1})v_{n-1}\quad \dots\quad (q_1, t_1)(\gamma_1, t'_1)v_1\ (q_0, t_0),$$

where each $v_i = (a_i^1, u_i^1)\ \dots\ (a_i^{l_i}, u_i^{l_i})$ *is a timed word over* Σ*. Additionally we require that for each* $i \leq n$ *and* $j = 1, \dots, l_i$*, the following conditions hold:*

(P1) *Structure:*

$$q_i \in Q, \gamma_i \in \Delta, a_i^j \in \Sigma, \gamma_i = \langle q_{i-1}, x, q_i\rangle, q_n = q_f \text{ and } a_n^1 \dots a_n^{l_n} = w_f.$$

(P2) *Distribution in time:*

$$n-i = t_i < t'_i < u_i^1 < u_i^2 < \dots < u_i^{l_i} < t_{i+1} = n-i+1.$$

(P3a) *Epsilon move: if* $\gamma_i = \langle q_{i-1}, \epsilon, q_i\rangle$ *then* $(v_i+1) \sqsubseteq v_{i-1}$*.*
(P3b) *Write move: if* $\gamma_i = \langle q_{i-1}, !a, q_i\rangle$ *then either* $a_i^1 = a$ *and* $(a_i^2 \dots a_i^{l_i} + 1) \sqsubseteq v_{i-1}$*, or* $(v_i+1) \sqsubseteq v_{i-1}$*.*
(P3c) *Read move: if* $\gamma_i = \langle q_{i-1}, ?a, q_i\rangle$ *then* $v_{i-1} = v'(a,t)v''$ *for some timed words* v', v'' *and* $t \in \mathbb{R}_+$*, such that* $(v_i+1) \sqsubseteq v'$*.*

Lemma 11. $\mathcal{S}$ *has a computation of the form (2) ending in* $(q_n, w_n) = (q_f, w_f)$ *if and only if there exists a lossy computation encoding ending in* (q_f, w_f) *as in Definition 10.*

Our aim is:

Lemma 12. *A purely universal automaton* $\mathcal{A}$ *can be effectively constructed such that* $L(\mathcal{A})$ *contains precisely all lossy computation encodings ending in* (q_f, w_f)*.*

The proof of this lemma will occupy the rest of this section. Automaton $\mathcal{A}$ will be defined as a conjunction of four automata, each responsible for some of the conditions from Definition 10:

$$\mathcal{A} := \mathcal{A}_{\text{struct}}\ \wedge\ \mathcal{A}_{\text{unit}}\ \wedge\ \mathcal{A}_{\text{strict}}\ \wedge\ \mathcal{A}_{\text{check}}.$$

All four automata will be purely universal and will use at most one clock. Automaton $\mathcal{A}_{\text{struct}}$ verifies condition *(P1)*, automata $\mathcal{A}_{\text{unit}}$ and $\mathcal{A}_{\text{strict}}$ jointly check condition *(P2)*, and $\mathcal{A}_{\text{check}}$ enforces the most involved conditions *(P3a) – (P3c)*.

We omit an obvious definition of $\mathcal{A}_{\text{struct}}$. We also omit the construction of the automaton $\mathcal{A}_{\text{unit}}$ checking that letters from Q appear precisely at times $0, 1, \ldots, n$, and automaton $\mathcal{A}_{\text{strict}}$ that accepts a timed word iff the first letter is at time 0 and no two consecutive letters appear at the same time.

Till now, all the automata were not only purely universal but also purely existential, i.e., deterministic. The power of universal choice will be only used in the last automaton $\mathcal{A}_{\text{check}}$, that checks for correctness of each transition step of $\mathcal{S}$. While analyzing definition of $\mathcal{A}_{\text{check}}$ we will comfortably assume that an input word meets all conditions verified by the other automata, otherwise the word is anyway not accepted. For conciseness, We implicitly assume that the automaton fails to accept if no rule is applicable. Moreover, when no clock is reset, we will omit writing it explicitly.

The transition rules of $\mathcal{A}_{\text{check}}$ from the initial state s_0 are as follows:

$$s_0, \Sigma \cup \Delta, \mathbf{tt} \;\mapsto\; s_0 \qquad\qquad \begin{array}{ll} s_0, q, \mathbf{tt} \;\mapsto\; s_0 \wedge (s_{\text{step}}, \{x\}), & \text{for } q \in Q \setminus \{q_0\} \\ s_0, q_0, \mathbf{tt} \;\mapsto\; \top. & \end{array}$$

Intuitively, at each $q \in Q$, except at q_0, an extra automaton is run from the state s_{step}, in order to check correctness of a single step. Symbol $\top$ on the right-hand side stands for a distinguished state that accepts unconditionally.

Now the rules $s_{\text{step}}, \gamma, \ldots \;\mapsto\; \ldots$ depend on $\gamma = \langle q, x, q' \rangle$. There are three cases, corresponding to conditions *(P3a), (P3b)* and *(P3c)*, respectively. Case *(P3b)*, not much different from *(P3a)*, is omitted here.

I. Case $\gamma = \langle q, \varepsilon, q' \rangle$*:* $\qquad s_{\text{step}}, \langle q, \varepsilon, q' \rangle, \mathbf{tt} \mapsto s_{\text{channel}}.$

In state s_{channel}, the automaton checks the condition (P3a), i.e., whether all consecutive letters from Σ are copied one time unit later. This is done by:

$$s_{\text{channel}}, Q, \mathbf{tt} \;\mapsto\; \top \qquad\qquad s_{\text{channel}}, a, \mathbf{tt} \;\mapsto\; s_{\text{channel}} \wedge (s_a^{+1}, \{x\}), \quad \text{for } a \in \Sigma.$$

Hence, the automaton starts a check from s_a^{+1} at every letter read. Note that this is precisely here where the universal branching is essential. The task of s_a^{+1} is to check that there is letter a one time unit later:

$$s_a^{+1}, a, x = 1 \;\mapsto\; \top \qquad\qquad s_a^{+1}, \overline{\Sigma}, x < 1 \;\mapsto\; s_a^{+1}$$

II. Case $\gamma = \langle q, ?a, q' \rangle$*:* $\qquad s_{\text{step}}, \langle q, ?a, q' \rangle, \mathbf{tt} \mapsto s_{?a} \wedge (s_{\text{try}?a}, \{x\}).$

The behaviour of $s_{?a}$ is very similar to s_{channel} but additionally it will start a new copy of the automaton in the state $s_{\text{try}?a}$. The goal of $s_{\text{try}?a}$ is to check for the letter a at the end of the present configuration.

$$s_{?a}, Q, \mathbf{tt} \;\mapsto\; \top \qquad\qquad s_{?a}, b, \mathbf{tt} \;\mapsto\; s_{?a} \wedge (s_b^{+1}, \{x\}) \wedge (s_{\text{try}?a}, \{x\}), \quad \text{for } b \in \Sigma.$$

Note the clock reset when entering to $s_{\text{try}?a}$. As we cannot know when the configuration ends we start $s_{\text{try}?a}$ at each letter read. If we realize that this was not the end (we see another channel letter) then the check just succeeds. If this was the end (we see a state) then the true check starts from the state $s_{\text{check?}a}$:

$$s_{\text{try?}a}, \Sigma, \mathbf{tt} \ \mapsto\ \top \qquad\qquad s_{\text{try?}a}, Q, \mathbf{tt} \ \mapsto\ s_{\text{check?}a}.$$

From $s_{\text{check?}a}$ we look for some a that appears more than one time unit later:

$$s_{\text{check?}a}, \overline{\Sigma}, x \leq 1 \ \mapsto\ s_{\text{check?}a}$$
$$s_{\text{check?}a}, a, x > 1 \ \mapsto\ \top \qquad\qquad s_{\text{check?}a}, b, x > 1 \ \mapsto\ s_{\text{check?}a}, \quad \text{for } b \in \Sigma \backslash \{a\}.$$

Automaton $\mathcal{A}_{\text{check}}$ has no other accepting states but $\top$.

By the very construction, $\mathcal{A}$ satisfies Lemma 12. By Lemma 11, $\mathcal{S}$ has a computation (2) ending in (q_f, w_f) if and only if $L(\mathcal{A})$ is nonempty. This completes the proof of Theorem 2.

5 Undecidability

In this section we point out that the alternating timed automata model cannot be extended with ϵ-transitions. It is known that ϵ-transitions extend the power of nondeterministic timed automata [2, 9]. Here we show some evidence that every extension of alternating timed automata with ϵ-transitions will have undecidable emptiness problem.

It turns out that there are many possible ways of introducing ϵ-transitions to alternating timed automata. To see the issues involved consider the question of whether such an automaton should be allowed to start uncountably many copies of itself or not. Facing these problems we have decided not to present any precise definition but rather to show where the problem is. We will show that the universality problem for purely existential automata with a very simple notion of ϵ-transitions is undecidable.

Timed words are written here in the same convention as in previous section: $w = (a_1, t_1)(a_2, t_2) \dots (a_n, t_n)$ means that the letter a_i appeared at time t_i.

We consider purely existential (i.e. nondeterministic) automata with one clock. We equip them now with additional ϵ-transitions of the form $q, \epsilon, \sigma \ \mapsto\ b$. The following trick is used to shorten formal definitions.

Definition 11. *A* nondeterministic timed automaton with ϵ-transitions *over Σ is a nondeterministic timed automaton over the alphabet $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$.*

For convenience, we want to distinguish an automaton $\mathcal{A}$ with ϵ-transitions over Σ from the corresponding automaton over Σ_ϵ; the latter will be denoted $\mathcal{A}_\epsilon$. Given a timed word v over Σ_ϵ, by $|v|_\epsilon$ we mean the timed word over Σ obtained from w by erasing all (timed) occurrences of ϵ.

Definition 12. *A timed word over Σ is accepted by a timed automaton $\mathcal{A}$ with ϵ-transitions if there is a timed word v over Σ_ϵ accepted by $\mathcal{A}_\epsilon$ such that $w = |v|_\epsilon$.*

Note that according to the definition, an accepting run is always finite. The main result of this section is:

Theorem 4. *The universality problem for one-clock nondeterministic timed automata with ϵ-transitions is undecidable.*

The proof is by reduction of the reachability problem for perfect channel systems, defined similarly as lossy reachability in Definition 9, but w.r.t. *perfect computation* of a channel systems. Not surprisingly, a perfect computation is any finite sequence of (perfect) transitions:

$$(q_0, \epsilon) \xrightarrow{\gamma_1} (q_1, w_1) \xrightarrow{\gamma_2} (q_2, w_2) \quad \ldots \quad \xrightarrow{\gamma_n} (q_n, w_n),$$

Theorem 5 ([8]). *The perfect reachability problem for channel systems is undecidable, assumed $|\Sigma| \geq 2$.*

Given a channel system $\mathcal{S} = (Q, q_0, \Sigma, \Delta)$ and a configuration (q_f, w_f), we effectively construct a one-clock nondeterministic timed automaton with ϵ-transitions $\mathcal{A}'$ over $\overline{\Sigma}$. Automaton $\mathcal{A}'$ will accept precisely the complement of the set of all *perfect computations encodings ending in* (q_f, w_f), defined by:

Definition 13. *A* perfect computation encoding ending in (q_f, w_f) *is defined as in Definition 10, but with the conditions* (P3a) – (P3c) *replaced by:*

(P3a) *if* $\gamma_i = \langle q_{i-1}, \epsilon, q_i \rangle$ *then* $(v_i + 1) = v_{i-1}$,
(P3b) *if* $\gamma_i = \langle q_{i-1}, !a, q_i \rangle$ *then* $(v_i + 1) = (a, t)v_{i-1}$, *for some* $t \in \mathbb{R}_+$.
(P3c) *if* $\gamma_i = \langle q_{i-1}, ?a, q_i \rangle$ *then* $(v_i(a, t) + 1) = v_{i-1}$, *for some* $t \in \mathbb{R}_+$.

Since each perfect computation encoding is a lossy one, $\mathcal{A}'$ will be defined as a disjunction, $\mathcal{A}' := \neg\mathcal{A} \vee \widehat{\mathcal{A}}$, of the complement of the automaton $\mathcal{A}$ from the previous section and another automaton $\widehat{\mathcal{A}}$. As automaton $\neg\mathcal{A}$ takes care of all timed words that are not lossy computation encodings, it is enough to have:

Lemma 13. *Automaton $\widehat{\mathcal{A}}$ accepts precisely these lossy computation encodings ending in (q_f, w_f) that are not perfect computation encodings.*

This will be enough for correctness of our reduction: $\mathcal{A}'$ will accept precisely the complement of the set of all perfect computation encodings. The construction of $\widehat{\mathcal{A}}$, omitted here, will be given in the full version of this paper.

6 Final Remarks

In this paper we have explored the possibilities opened by the observation that the universality problem for nondeterministic timed automata is decidable. We have extended this result to obtain a class of timed automata that is closed under boolean operations and that has decidable emptiness problem. We have shown that despite being decidable the problem has prohibitively high complexity. We have also considered the extension of the model with epsilon transitions which points out what makes the model decidable and what further extensions are not possible. Maybe somewhat surprisingly universality for 1-clock nondeterministic timed automata but over infinite words is undecidable. We plan to discuss this issue in the full version of the paper.

We see several topics for further work: (1) Adding event-clocks to the model. It seems that one would still obtain a decidable model. (2) Finding logical characterizations of the languages accepted by alternating timed automata with one clock. Since we have the closure under boolean operations, we may hope to find one. (3) Finding a different syntax that will avoid the prohibitive complexity of the emptiness problem. There may well be another way of presenting alternating timed automata that will give the same expressive power but for which the emptiness test will be easier.

Acknowledgments We thank anonymous referees for valuable remarks.

References

1. P. Abdulla, K. Čerāns, B. Jonsson, and Y. Tsay. General decidability theorems for infinite state systems. In *LICS'96*, p. 313–323, 1996.
2. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. R. Alur, M. Bernadsky, and P. Madhusudan. Optimal reachability for weighted timed games. In *ICALP'04*, volume 3124 of LNCS, p. 122–133, 2004.
4. R. Alur, L. Fix, and T. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 204:253-273, 1999.
5. E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symp. System Structure and Control*, p. 469–474, 1998.
6. P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *FSTTCS'04*, LNCS, 2004.
7. P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. In *CAV'03*, volume 2725 of LNCS, p. 180–192, 2003.
8. D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
9. B. Bérard, V. Diekert, P. Gastin, and A. Petit. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2):145–182, 1998.
10. F. Cassez, T. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *Hybrid Systems Computation and Control (HSCC'02)*, volume 2289 of LNCS, p. 134–148, 2002.
11. M. Dickhöfer, T. Wilke. Timed alternating tree automata: the automata-theoretic solution to the TCTL model checking problem. In *ICALP'99*, volume 1644 of LNCS, p. 281-290, 1999.
12. C. Dima. Real-time automata and the Kleene algebra of sets of real numbers. In *STACS'00*, volume 1170 of LNCS, p. 279–289, 2000.
13. A. Finkel and Ph. Schnoebelen. Well structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.
14. G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.*, 2(7):326–336, 1952.
15. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model checking timed automata with one or two clocks. In *CONCUR'04*, volume 3170 of LNCS, p. 387–401, 2004.
16. J. Ouaknine and J. Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *LICS'04*, p. 54–63, 2004.
17. Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002.