Decidability and Complexity Results for Timed Automata via Channel Machines

Parosh Aziz Abdulla[1], Johann Deneux[1], Joël Ouaknine[2] and James Worrell[3]

[1] Department of Computer Systems, Uppsala University, Sweden
{parosh,johannd}@it.uu.se
[2] Oxford University Computing Laboratory, UK
joel@comlab.ox.ac.uk
[3] Department of Mathematics, Tulane University, USA
jbw@math.tulane.edu

Abstract. This paper is concerned with the language inclusion problem for timed automata: given timed automata $\mathcal{A}$ and $\mathcal{B}$, is every word accepted by $\mathcal{B}$ also accepted by $\mathcal{A}$? Alur and Dill [5] showed that the language inclusion problem is decidable if $\mathcal{A}$ has no clocks and undecidable if $\mathcal{A}$ has two clocks (with no restriction on $\mathcal{B}$). However, the status of the problem when $\mathcal{A}$ has one clock is not determined by [5]. In this paper we close this gap for timed automata over infinite words by showing that the one-clock language inclusion problem is undecidable. For timed automata over finite words, building on our earlier paper [19], we show that the one-clock language inclusion problem is decidable with nonprimitive recursive complexity. This reveals a surprising divergence between the theory of timed automata over finite words and over infinite words. Finally, we show that if ε-transitions or non-singular postconditions are allowed, then the one-clock language inclusion problem is undecidable over both finite and infinite words.

1 Introduction

An execution of a real-time system can be modelled as a *timed word* consisting of a sequence of events and their associated timestamps, and properties of such systems can be expressed as languages of timed words. Timed automata were introduced by Alur and Dill [5] to specify languages of timed words, and have since been extensively studied by many researchers. In particular, timed automata have been used as the foundation for several verification algorithms and tools (see [6] for a survey).

One of the most fundamental results about timed automata is the undecidability of the language inclusion problem: 'Given timed automata $\mathcal{A}$ and $\mathcal{B}$, is every word accepted by $\mathcal{B}$ also accepted by $\mathcal{A}$?' This problem is undecidable irrespective of whether one considers automata over finite words or automata over infinite words with a Büchi acceptance condition. In this context it is natural to seek subclasses of timed automata, with reduced expressive power, for which the language inclusion problem is decidable [3, 4, 6, 7, 13, 18, 19]. In this

paper we consider subclasses defined by the number of clocks an automaton has. In particular, we consider the *n-clock language inclusion problem* in which $\mathcal{A}$ is allowed n clocks but where no restriction is placed on $\mathcal{B}$.

A close analysis of the proof of the undecidability of language inclusion in [5] reveals that the tightest possible formulation of their result is that the two-clock language inclusion problem is undecidable. On the other hand, from the decidability of language emptiness for timed automata, also proved in [5], it follows that the zero-clock language inclusion problem is (PSPACE-complete) decidable. This leaves an interesting open question about the status of the one-clock language inclusion problem. In fact, many interesting specifications can be expressed by automata with a single clock, or parallel combinations thereof. This is particularly so for *alternating timed automata* [16, 20]. For instance, every formula of *Metric Temporal Logic* [8, 7] can be translated into an alternating timed automaton with a single clock [20]. The model checking problem then corresponds to language inclusion.

Recently, using techniques from the theory of well-quasi-ordered transition systems [11], we showed that over finite words the one-clock language inclusion problem is decidable [19]. However, while finite words are sufficient to capture safety properties, to capture liveness or fairness properties it is most natural to consider automata over infinite words. The main result of this paper is that, for timed automata over infinite words (with Büchi acceptance condition) the one-clock language inclusion problem is undecidable. This reveals a surprising divergence between the theory of timed automata over finite words and over infinite words. We also show that over finite words the one-clock language inclusion problem has nonprimitive recursive complexity. Finally, language inclusion becomes undecidable over both finite and infinite words if ε-transitions are allowed or if reset clocks have nonsingular postconditions (as in [9, 12]).

We use *channel machines* [10] as a convenient middleware between Turing machines and timed automata. This allows us to develop a schematic approach to proving undecidability and complexity results for various classes of timed automata. In each case we show how to encode a certain class of channel computations as a timed language, whose complement can be recognized by a timed automaton of a certain type.

Related Work. The nonprimitive recursive complexity of language inclusion over finite words and the undecidability of language inclusion over finite words with ε-transitions have recently and independently been proved by Lasota and Walukiewicz [16]. They also make use of channel machines in their work, although via a different encoding of channel histories as timed words.

Alur, La Torre and Madhusudan [4] consider automata with *perturbed clocks* whose rates may vary; they show that for every automaton with a single perturbed clock there is an equivalent deterministic timed automaton. It follows that the language inclusion problem is decidable for this class of automata. Laroussinie, Markey and Schnoebelen [15] classify the complexity of deciding language emptiness for timed automata with one, two and three clocks respectively.

2 Timed Automata and Timed Words

Let Σ be a finite alphabet and write Σ^ε for $\Sigma \cup \{\varepsilon\}$, where $\varepsilon \notin \Sigma$. Let $\mathbb{R}_+$ be the set of non-negative reals. A *timed event* is a pair (t, a), where $t \in \mathbb{R}_+$ is called the *timestamp* of the *event* $a \in \Sigma$. A *timed word* is a finite or infinite sequence $u = (t_0, a_0)(t_1, a_1)(t_2, a_2)\ldots$ of timed events whose sequence of timestamps $t_0 t_1 t_2 \ldots$ is non-decreasing and is either finite or diverges to infinity. (This last assumption rules out so-called *Zeno words*.) We say that a timed word is *strictly monotonic* if its sequence of timestamps is strictly monotonic increasing. We write $T\Sigma^*$ for the set of finite timed words over alphabet Σ and $T\Sigma^\omega$ for the set of infinite timed words over alphabet Σ.

Let X be a finite set of clocks, denoted x, y, z, etc. We define the set $\Phi(X)$ of clock constraints over X via the following grammar (here $k \in \mathbb{N}$ is a non-negative integer).

$$\phi ::= x < k \;\mid\; x \leqslant k \;\mid\; \phi \wedge \phi \;\mid\; \neg\phi \;\; .$$

Definition 1. *A* timed automaton *is a tuple* $\mathcal{A} = (\Sigma, S, S_0, F, X, E)$*, where*

- Σ *is a finite alphabet of events,*
- S *is a finite set of control states,*
- $S_0 \subseteq S$ *is a set of initial control states,*
- $F \subseteq S$ *is a set of accepting control states,*
- X *is a finite set of clocks, and*
- $E \subseteq S \times S \times \Phi(X) \times \Sigma^\varepsilon \times 2^X \times \Phi(X)$ *is a finite set of edges. An edge* $(s, s', \phi, \alpha, R, \phi')$ *allows* α*-labelled transition from* s *to* s'*, provided the precondition* ϕ *on clocks is met. Afterwards, the clocks in* R *are nondeterministically reset to values satisfying the postcondition* ϕ'*, and all other clocks remain unchanged.*

A *clock valuation* of $\mathcal{A}$ is a function $\nu : X \to \mathbb{R}_+$. If $\delta \in \mathbb{R}_+$, we let $\nu + \delta$ be the clock valuation such that $(\nu + \delta)(x) = \nu(x) + \delta$ for all $x \in X$. A *global state* of $\mathcal{A}$ is a pair (s, ν), where $s \in S$ is a control state and ν is a clock valuation. Write $Q = S \times (\mathbb{R}_+)^X$ for the set of global states.

Automaton $\mathcal{A}$ induces an $(\mathbb{R}_+ \times \Sigma^\varepsilon)$-labelled transition relation on the set of global states as follows. Write $(s, \nu) \xrightarrow{\delta,\, \alpha} (t, \nu')$ iff there is an edge $(s, t, \phi, \alpha, R, \phi') \in E$ such that $\nu + \delta$ satisfies ϕ, ν' satisfies ϕ' and $(\nu + \delta)(x) = \nu'(x)$ for all $x \notin R$.

A *run* of $\mathcal{A}$ is a finite or infinite sequence of transitions

$$(s_0, \nu_0) \xrightarrow{\delta_0,\, \alpha_0} (s_1, \nu_1) \xrightarrow{\delta_1,\, \alpha_1} (s_2, \nu_2) \xrightarrow{\delta_2,\, \alpha_2} \cdots \tag{1}$$

where $s_0 \in S_0$ is an initial control state and $\nu_0(x) = 0$ for all $x \in X$. We require that an infinite run contain infinitely many transitions labelled from Σ and that $\sum_{i=0}^{\infty} \delta_i$ be infinite.

A finite run is *accepting* if the last control state in the run is accepting. An infinite run is accepting if infinitely many control states in the run are accepting. Let $\alpha_{i_0} \alpha_{i_1} \alpha_{i_2} \ldots$ be the sequence of non-ε-labels occurring in an accepting run and write $t_j = \sum_{i=0}^{j} \delta_i$. Then we say that the timed word $(t_{i_0}, \alpha_{i_0})(t_{i_1}, \alpha_{i_1})(t_{i_2}, \alpha_{i_2})\ldots$

is *accepted by* $\mathcal{A}$. We write $L_f(\mathcal{A})$ for the set of finite timed words accepted by $\mathcal{A}$ and $L_\omega(\mathcal{A})$ for the set of infinite timed words accepted by $\mathcal{A}$.

Remark 1. Definition 1 represents quite a general model of timed automata. We will adopt the convention that, unless otherwise specified, a given timed automaton has no ε-transitions, and has *singular postconditions*, i.e., for each edge $(s, t, \phi, \alpha, R, \phi')$, if clock valuations ν and ν' both satisfy ϕ', then $\nu(x) = \nu'(x)$ for all $x \in R$.

3 Hardness Results over Finite Words

In [19] we showed that it is decidable whether $L_f(\mathcal{B}) \subseteq L_f(\mathcal{A})$ for an arbitrary timed automaton $\mathcal{B}$ and a one-clock automaton $\mathcal{A}$. In this section we show that this problem has nonprimitive recursive complexity and is undecidable if $\mathcal{A}$ is allowed ε-transitions or non-singular postconditions. We prove these results by reduction from the reachability problem for channel machines.

A *channel machine* [1, 10, 21] consists of a finite-state automaton acting on an unbounded fifo channel (or buffer). More precisely, a channel machine is a tuple $\mathcal{C} = (S, s_0, M, \Delta)$, where S is a finite set of *control states*, $s_0 \in S$ is the *initial control state*, M is a finite set of *messages*, and $\Delta \subseteq S \times L \times S$ is the transition relation over label set $L = \{m!, m? : m \in M\}$.

A *global state* of $\mathcal{C}$ is a pair (s, x), where $s \in S$ is the control state and $x \in M^*$ is the contents of the channel. The rules in Δ induce an L-labelled transition relation on the set of global states as follows: $(s, m!, t) \in \Delta$ yields a transition $(s, x) \xrightarrow{m!} (t, x \cdot m)$ that writes $m \in M$ to the tail of the channel, and $(s, m?, t) \in \Delta$ yields a transition $(s, m \cdot x) \xrightarrow{m?} (t, x)$ that reads $m \in M$ from the head of the channel. If we only allow the transitions indicated above, then we call $\mathcal{C}$ an *error-free* channel machine.

We also consider channel machines that operate with *insertion errors*. Given $x, y \in M^*$, write $x \sqsubseteq y$ if x can be obtained from y by deleting any number of letters, e.g. sub $\sqsubseteq$ stubborn, as indicated by the underlining. Following [21] we introduce *insertion errors* by extending the transition relation on global states with the following clause: if $(s, x) \xrightarrow{\alpha} (t, y)$, $x' \sqsubseteq x$ and $y \sqsubseteq y'$, then $(s, x') \xrightarrow{\alpha} (t, y')$. Dually, we define *lossy channel machines* by adding by a clause: if $(s, x) \xrightarrow{\alpha} (t, y)$, $x \sqsubseteq x'$ and $y' \sqsubseteq y$, then $(s, x') \xrightarrow{\alpha} (t, y')$.

A *computation* of $\mathcal{C}$ is a finite or infinite sequence of transitions between global states $(s_0, x_0) \xrightarrow{\alpha_0} (s_1, x_1) \xrightarrow{\alpha_1} (s_2, x_2) \xrightarrow{\alpha_2} \cdots$.

The *control-state reachability problem* asks, given a channel machine $\mathcal{C} = (S, s_0, M, \Delta)$ and a control state $t \in S$, whether there is a computation of $\mathcal{C}$ starting in global state (s_0, ε) and ending in global state (t, ε). It is well-known that the control-state reachability problem for error-free channel machines is undecidable[1]. Next we show how to reduce the control-state reachability problem

[1] The usual formulation of the problem asks whether there is a computation from (s_0, ε) to (t, x) for some $x \in M^*$. It is straightforward to reduce this problem to the formulation above.

for error-free channel machines to the universality problem (which is a special case of language inclusion) for various classes of timed automata.

Let $\mathcal{C} = (S, s_0, M, \Delta)$ and $t \in S$ be an instance of the control-state reachability problem. Given this data, let $\Sigma = \{m!, m? : m \in M\} \cup \{\checkmark\}$ be a finite alphabet. We encode the finite control of $\mathcal{C}$ as an untimed automaton (i.e., a timed automaton with no clocks) $\mathcal{A}_{cont}$ over alphabet Σ. $\mathcal{A}_{cont}$ is just the underlying control automaton of $\mathcal{C}$ with a $\checkmark$-labelled self-transition added to every control state, with s_0 as the initial control state and $t \in S$ as the only accepting control state. Let L_{cont} denote the timed language $L_f(\mathcal{A}_{cont})$.

Definition 2. *Let $L_{chan} \subseteq T\Sigma^*$ consist of those timed words u such that:*

1. *u is strictly monotonic.*
2. *u contains a $\checkmark$-event at time zero, and thereafter consecutive $\checkmark$-events are separated by one time unit.*
3. *Every $m!$-event in u is followed one time unit later by an $m?$-event.*
4. *Every $m?$-event is preceded one time unit earlier by an $m!$-event.*

Clauses 3 and 4 capture the channel discipline: every message written to the channel is read from the channel one time unit later, and every message that is read from the channel was written to the channel one time unit earlier. The one-to-one unit-time-delayed correspondence between read and write events ensures that messages are read from the channel in the order that they were written to the channel. The requirement that every message written to the channel is eventually read corresponds to the fact that we consider computations that end with an empty channel. The $\checkmark$-events in L_{chan} have no particular significance other than to aid the encoding below.

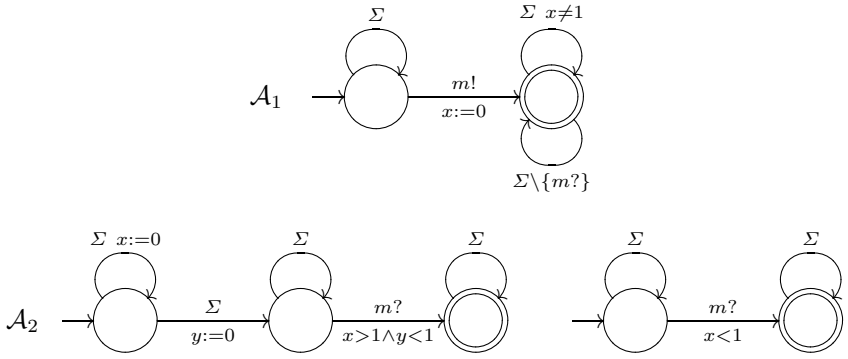
Proposition 1. *$\mathcal{C}$ has an error-free computation from (s_0, ε) to (t, ε) iff $L_{cont} \cap L_{chan} \neq \emptyset$.*

Let $\overline{\mathcal{A}_{cont}}$ denote the complement of $\mathcal{A}_{cont}$ as an untimed automaton. It is clear that $\overline{\mathcal{A}_{cont}}$ is also the complement of $\mathcal{A}_{cont}$ with respect to timed languages, i.e., $L_f(\overline{\mathcal{A}_{cont}}) = T\Sigma^* - L_{cont}$. Now suppose that $\mathcal{A}_{chan}$ is a timed automaton such that $L_f(\mathcal{A}_{chan}) = T\Sigma^* - L_{chan}$. From Proposition 1 it holds that $\overline{\mathcal{A}_{cont}} \cup \mathcal{A}_{chan}$ is universal (i.e. accepts every timed word) iff $\mathcal{C}$ has no error-free computation from (s_0, ε) to (t, ε). Since the control-state reachability problem is undecidable for error-free channel machines, it follows that the universality problem is undecidable for any class of timed automata that is closed under unions and can capture the complement of L_{chan}.

3.1 Two clocks

We show how to define a timed automaton $\mathcal{A}_{chan}$ with two clocks such that $L_f(\mathcal{A}_{chan}) = T\Sigma^* - L_{chan}$. We define $\mathcal{A}_{chan}$ as the disjunction of several automata, each of which accepts the set of words that fail to satisfy a particular clause in the definition of L_{chan}. The interesting clauses here are 3 and 4.

Automaton $\mathcal{A}_1$, below, accepts those timed words in which some $m!$-event is not followed one time unit later by an $m?$-event, i.e., those words that fail to satisfy Clause 3 in Definition 2. Automaton $\mathcal{A}_2$ (both left-hand and right-hand components) accepts those timed words in which some $m?$-event is not preceded one time unit earlier by *any* event. Note that a strictly monotonic timed word fails to satisfy Clause 4 in Definition 2 if and only if it is either accepted by $\mathcal{A}_2$ or contains an α-event followed one time unit later by an $m?$-event, with $\alpha \neq m!$. It is straightforward to capture this last condition with a one-clock timed automaton. In fact $\mathcal{A}_2$ is the only component of $\mathcal{A}_{chan}$ that uses two clocks.

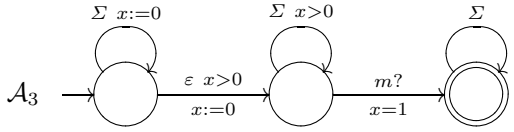


Thus we obtain a new proof of Alur and Dill's classical result [5].

Theorem 1. *The universality problem for timed automata with two clocks is undecidable.*

3.2 ε-transitions

By allowing ε-transitions, we can replace the left-hand component of automaton $\mathcal{A}_2$, above, with the following automaton which uses only one clock. The ε-transition and the $m?$-transition in $\mathcal{A}_3$ are separated by exactly one time unit; this prevents any visible event from preceding this occurrence of $m?$ by one time unit.

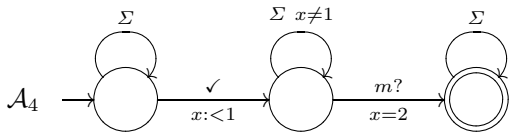


Theorem 2. *The universality problem for the class of timed automata with one clock and ε-transitions is undecidable.*

3.3 Non-singular postconditions

Instead of ε-transitions we can consider non-singular postconditions for clock resets. In this case we can replace automaton $\mathcal{A}_3$ with the following one-clock

timed automaton, where the ✓-labelled edge non-deterministically resets x to a value strictly less than 1.



Theorem 3. *The universality problem for the class of timed automata with one clock and with non-singular postconditions is undecidable.*

3.4 Complexity

The control-state reachability problem for lossy channel machines was shown to be decidable, in contrast to the error-free case, by Abdulla and Jonsson [2]. Later Schnoebelen [21] proved that it has nonprimitive recursive complexity.

Proposition 2. *The control-state reachability problem for channel machines with insertion errors has nonprimitive recursive complexity.*

Proof. Given a channel machine $\mathcal{C} = (S, s_0, M, \Delta)$, define a new transition relation $\Delta^{\mathrm{op}} \subseteq S \times \{m!, m? : m \in M\} \times S$ by

$$\Delta^{\mathrm{op}} = \{(s, m!, t) : (t, m?, s) \in \Delta\} \cup \{(s, m?, t) : (t, m!, s) \in \Delta\} \ .$$

Notice that there is a transition from global state (s, x) to global state (t, y) under Δ iff there is a transition from $(t, rev(y))$ to $(s, rev(x))$ under Δ^{op}, where $rev : M^* \to M^*$ reverses the order of a finite string. Thus there is computation with lossiness errors from (s, ε) to (t, ε) under Δ iff there is a computation with insertion errors from (t, ε) to (s, ε) under Δ^{op}. This observation allows us to reduce the control-state reachability problem for channel machines with lossiness errors to the control-state reachability problem for channel machines with insertion errors. □

Define the timed language $L_{ins} \subseteq T\Sigma^*$ to consist of those timed words satisfying Clauses 1-3 in Definition 2. Thus for a word in L_{ins}, every $m!$-event is followed one time unit later by an $m?$-event, but every $m?$-event need not be preceded one time unit earlier by an $m!$-event. This corresponds to a channel with insertion errors.

Proposition 3. *$\mathcal{C}$ has a computation with insertion errors starting in state (s_0, ε) and ending in state (t, ε) iff $L_{cont} \cap L_{ins} \neq \emptyset$.*

Note that we can express $T\Sigma^* - L_{ins}$ as the language of a one-clock timed automaton. This automaton incorporates $\mathcal{A}_1$ in Section 3.1, but not $\mathcal{A}_2$. Thus we obtain

Theorem 4. *The universality problem for the class of timed automata with a single clock has nonprimitive recursive complexity.*

4 Universality for One-clock Büchi Automata

In this section we prove the undecidability of the following universality problem: 'Given a one-clock timed automaton $\mathcal{A}$ (without ε-transitions and with singular postconditions) does $L_\omega(\mathcal{A}) = T\Sigma^\omega$?'

As in Section 3 the idea behind the proof is to encode the computations of a certain type of channel machine as a timed language. To this end, say that a non-terminating computation of a channel machine is *space-bounded* if there exists $N \in \mathbb{N}$ such that the number of messages stored on the channel during the computation never exceeds N. We define a timed language $L_{bound} \subseteq T\Sigma^\omega$ encoding space-bounded computations of a channel with insertion errors. We capture the space bound by requiring an upper bound on the number of events per time unit for each word $u \in L_{bound}$. Besides space-boundedness the other key ingredient in our proof is the notion of *alternation*, defined below.

Define a channel machine $\mathcal{C}$ to be *alternating* if we can partition the set of control states into two classes, called *read states* and *write states* respectively, such that each edge is of the form $(s, m?, t)$ with s a read state and t a write state, or of the form $(s, m!, t)$ with s a write state and t a read state. Then any computation of $\mathcal{C}$ consists of an alternating sequence of read transitions and write transitions.

The *recurrent-state problem* for alternating channel machines is as follows. Given an alternating channel machine $\mathcal{C} = (S, s_0, M, \Delta)$, does there exist $x \in M^*$ such that $\mathcal{C}$ has a non-terminating computation starting in global state (s_0, x) and visiting s_0 infinitely often? The following proposition is proved in Appendix A.

Proposition 4. *The recurrent-state problem for error-free alternating channel machines is undecidable.*

The *space-bounded recurrent-state problem* for alternating channel machines with insertion errors asks if a given alternating channel machine has a space-bounded computation, possibly with insertion errors, passing infinitely often through the initial control state. The following proposition, which is reminiscent of a result of Mayr [17] on lossy counter machines, asserts that this problem is undecidable.

Proposition 5. *The space-bounded recurrent-state problem for alternating channel machines with insertion errors is undecidable.*

Proof. Given an alternating channel machine $\mathcal{C}$, we claim that $\mathcal{C}$ has an error-free recurrent computation iff it has a space-bounded recurrent computation with insertion errors. Then Proposition 5 follows from Proposition 4. Indeed, since $\mathcal{C}$ is alternating, at any point in a computation the total number of insertion errors up to that point is within one of the current size of the channel minus the size of the channel at the start of the computation. Thus any error-free computation of $\mathcal{C}$ is space-bounded, and any space-bounded computation of $\mathcal{C}$ with insertion errors is eventually error-free (the space-bound gives an upper bound on the total number of insertion errors). □

Definition 3. *Given a strictly monotonic timed word* $u = (t_0, a_0)(t_1, a_1)\ldots$, *define* $\text{density}(u) = \sup\{j - i : t_j - t_i \leqslant 1\}$. *The density of a timed word measures the maximum number of events in any time unit along the word.*

Theorem 5. *The universality problem for one-clock Büchi timed automata is undecidable.*

The proof is by reduction from the space-bounded recurrent-state problem for alternating machines with insertion errors. Given an alternating channel machine $\mathcal{C} = (S, s_0, M, \Delta)$, we define a one-clock Büchi timed automaton $\mathcal{A}$ such that $\mathcal{C}$ has a space-bounded recurrent computation with insertion errors iff $\mathcal{A}$ is non-universal.

Let $\Sigma = \{m!, m? : m \in M\} \cup \{\checkmark\}$ be a finite alphabet. We encode the finite control of $\mathcal{C}$ as a Büchi timed automaton $\mathcal{A}_{cont}$ with no clocks over alphabet Σ. $\mathcal{A}_{cont}$ is just the underlying control automaton of $\mathcal{C}$ with a $\checkmark$-labelled self-transition added to every control state and with s_0 as the initial control state and only accepting control state. Let L_{cont} denote the timed language $L_\omega(\mathcal{A}_{cont})$.

Next we capture the behaviour of a space-bounded channel with insertion errors using a timed language L_{bound} over alphabet Σ.

Definition 4. L_{bound} *consists of those timed words* u *satisfying:*

1. u *is strictly monotonic and contains infinitely many non-*$\checkmark$*-events.*
2. *There is a* $\checkmark$*-event at time zero, and thereafter consecutive* $\checkmark$*-events are separated by one time unit.*
3. *For every* $m!$*-event in* u *there is an* $m?$*-event one time unit later.*
4. *For every* $m?$*-event in* u *there is a* $n!$*-event one time unit later, for some* $n \in M$.
5. $\text{density}(u) < \infty$.

As with the corresponding clause in Definition 2, Clause 3 captures the channel discipline: every message sent is received. The channel has insertion errors because not every $m?$-event is preceded one time unit earlier by an $m!$-event. On the other hand, Clause 4 has nothing to do with the channel discipline. However its presence, together with Clauses 2 and 3, ensures that for every event of $u \in L_{bound}$ there is an event exactly one time unit later. (This fact will play a significant role later.) Since we are dealing with alternating channel machines, the imposition of Clause 4 will prove to be no restriction when we seek to match words in L_{bound} with channel computations. Finally, Clause 5 corresponds to the space-boundedness of the channel.

Proposition 6. $\mathcal{C}$ *has a space-bounded recurrent computation with insertion errors iff* $L_{cont} \cap L_{bound} \neq \emptyset$.

Proof. ($\Leftarrow$) Let $u \in L_{cont} \cap L_{bound}$. We show how to recover a space-bounded recurrent computation of $\mathcal{C}$ from u. Since $u \in L_{cont}$, the automaton $\mathcal{A}_{cont}$, which represents the finite control of $\mathcal{C}$, has a run

$$(s_0, \nu_0) \xrightarrow{\delta_0, \alpha_0} (s_1, \nu_1) \xrightarrow{\delta_1, \alpha_1} (s_2, \nu_2) \xrightarrow{\delta_2, \alpha_2} \cdots \tag{2}$$

on u. Let $\alpha_{i_0}\alpha_{i_1}\alpha_{i_2}\ldots$ be the sequence of non-✓-events in u. Then we obtain a recurrent computation of $\mathcal{C}$

$$(s_{i_0}, x_0) \xrightarrow{\alpha_{i_0}} (s_{i_1}, x_1) \xrightarrow{\alpha_{i_1}} (s_{i_2}, x_2) \xrightarrow{\alpha_{i_2}} \cdots$$

where $x_j \in M^*$ is the sequence of messages that occur as read events in the unit time interval $\big(t_{i_{j-1}}, t_{i_{j-1}} + 1\big]$, where, by convention, $i_{-1} = 0$. Since $u \in L_{bound}$, Clause 3 in Definition 4 ensures that this is a legitimate computation of $\mathcal{C}$, albeit with insertion errors. Since u has finite density this computation is space-bounded.

($\Rightarrow$) We have already observed that if $\mathcal{C}$ has a space-bounded recurrent computation with insertion errors, then it has a space-bounded recurrent error-free computation. The trace of channel events along such an error-free computation can easily be encoded as a word in L_{bound} as we now explain. Since $\mathcal{C}$ is alternating, there is a number $N \in \mathbb{N}$ such that the size of the channel is either N or $N-1$ at any point in the computation. When any message is written to the channel, the machine performs exactly $2N-1$ (read- and write-) operations before that message is read off the channel. We transform the sequence of read and write events along a computation into a timed word u by putting exactly $1/(2N-1)$ time units between consecutive events. This automatically guarantees that Clauses 3-5 in Definition 4 hold. Finally, adding ✓-events at integer times yields a timed word in L_{bound}. □

Similar to the development in Section 3, the undecidability of the universality problem will follow from Proposition 6 provided that we can define a one-clock timed automaton $\mathcal{A}_{bound}$ such that $L_\omega(\mathcal{A}_{bound}) = T\Sigma^\omega - L_{bound}$. We define $\mathcal{A}_{bound}$ to be the disjunction of several automata, corresponding to the different clauses in the definition of L_{bound}. It is straightforward, for each clause 1–4, to define an automaton that accepts precisely the timed words that fail to satisfy that clause. Below we define two automata $\mathcal{A}_{inc}$ and $\mathcal{A}_{dec}$ such that, if a timed word u already satisfies 1–4, then it is accepted by $\mathcal{A}_{inc}$ or $\mathcal{A}_{dec}$ precisely if it fails Clause 5, i.e., it has infinite density.

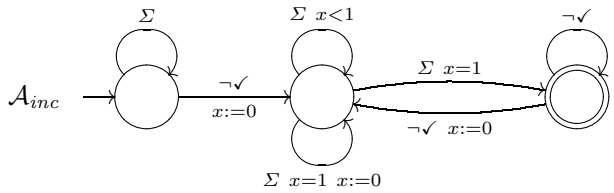
First we recall the following simple proposition about real numbers.

Proposition 7. *If $\mathbf{x} = \langle x_n : n \in \mathbb{N}\rangle$ is a sequence of real numbers in the open interval $(0,1)$ that takes on infinitely many values, then $\mathbf{x}$ has either a strictly increasing subsequence or a strictly decreasing subsequence.*

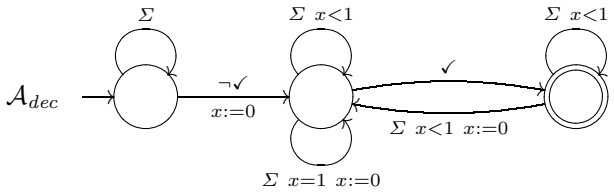
Let $u = (t_0, a_0)(t_1, a_1)(t_2, a_2)\ldots$ be a timed word satisfying Clauses 1–4 in Definition 4. Then for every event of u there is an event exactly one time unit later. Thus u has infinite density iff $\{\mathit{frac}(t_i) : i \in \mathbb{N}\}$ is infinite. By Proposition 7, this holds iff the sequence $\langle \mathit{frac}(t_i) : i \in \mathbb{N}\rangle$ has either a strictly increasing subsequence or a strictly decreasing subsequence. We define an automaton $\mathcal{A}_{inc}$ that accepts u iff $\langle \mathit{frac}(t_i) : i \in \mathbb{N}\rangle$ has a strictly increasing subsequence, and an automaton $\mathcal{A}_{dec}$ that accepts u iff $\langle \mathit{frac}(t_i) : i \in \mathbb{N}\rangle$ has a strictly decreasing subsequence.

Consider a run of $\mathcal{A}_{inc}$ (depicted below) on $u = (t_0, a_0)(t_1, a_1)(t_2, a_2)\ldots$. Let t_{i_j} be the timestamp of the transition that resets clock x for the j-th time. Notice

that either $t_{i_{j+1}} = t_{i_j} + 1$ or $frac(t_{i_{j+1}}) > frac(t_{i_j})$. The Büchi condition ensures that the second eventuality holds infinitely often in the run, and so the sequence $frac(t_{i_j})$ has a strictly increasing subsequence. Thus, among those timed words u satisfying Clauses 1–4 in Definition 4, $\mathcal{A}_{inc}$ accepts precisely those for which $\langle frac(t_i) : i \in \mathbb{N} \rangle$ has a strictly increasing subsequence. (Notice the importance in the operation $\mathcal{A}_{inc}$ of the fact that for each event in u there is an event one time unit later.)



$\mathcal{A}_{dec}$ (depicted below) operates in a similar manner to $\mathcal{A}_{inc}$ except that it accepts those words $u = (t_0, a_0)(t_1, a_1)(t_2, a_2)\ldots$ for which $\langle frac(t_i) : i \in \mathbb{N} \rangle$ has a strictly decreasing subsequence.



5 Remarks and Future Work

The main result of this paper is that the universality problem for one-clock Büchi timed automata is undecidable. A closely related problem concerns the decidability of the satisfiability and model checking problems for Metric Temporal Logic (MTL) [8, 7]. This logic is known to be undecidable under an *interval semantics*. However, under a *point-based semantics*—i.e., interpreting the logic over timed words—the satisfiability problem is open, cf. [20].

In [20] we show how to translate an MTL formula into an *alternating timed automaton* with a single clock. In terms of alternating automata, the present paper shows that the emptiness problem for one-clock alternating automata with co-Büchi acceptance condition is undecidable. However one can express MTL formulas using a particularly simple acceptance condition—the so-called *weak parity acceptance condition*—which specializes both the Büchi and co-Büchi acceptance conditions[2]. It does not seem possible to capture the timed language L_{bound} by an MTL formula, or, more generally, a one-clock alternating automata

[2] In the untimed case an alternating automaton with a Büchi or co-Büchi acceptance condition can be translated into an alternating automaton with a weak parity acceptance condition, however this translation does not apply in the timed case.

with weak parity acceptance condition. So it remains possible that the satisfiability problem for MTL is decidable, and we are currently investigating this question.

References

1. P. Abdulla and B. Jonsson. Undecidable Verification Problems with Unreliable Channels. *Information and Computation*, 130:71–90, 1996.
2. P. Abdulla and B. Jonsson. Model checking of systems with many identical timed processes. *Theoretical Computer Science*. 290(1):241-264, 2003.
3. R. Alur, L. Fix and T. Henzinger. Event-clock automata: a determinizable class of timed automata. *Theoretical Computer Science* 211(1-2):253–273, 1999.
4. R. Alur, S. La Torre and P. Madhusudan. Perturbed Timed Automata. To appear in proceedings of *HSCC 05*.
5. R. Alur and D. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.
6. R. Alur and P. Madhusudan. Decision problems for timed automata: A survey, *4th Intl. School on Formal Methods for Computer, Communication, and Software Systems: Real Time*, LNCS 3185, Springer-Verlag, 2004.
7. R. Alur, T. Feder and T. Henzinger. The Benefits of Relaxing Punctually. *Journal of the ACM*, 43:116–146, 1996.
8. R. Alur and T. Henzinger. Real-time Logics: Complexity and Expressiveness. *Information and Computation*, 104:35–77, 1993.
9. P. Bouyer, C. Dufourd, E. Fleury and A. Petit. Updatable timed automata. *Theoretical Computer Science* 321(2-3):291-345, 2004.
10. G. Cécé, A. Finkel and S. Purushothaman Iyer. Unreliable Channels are Easier to Verify Than Perfect Channels. *Information and Computation*, 124:20–31, 1996.
11. A. Finkel and P. Schnoebelen. Well-Structured Transition Systems Everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.
12. T.A. Henzinger, P.W. Kopke, A. Puri and P. Varaiya. What's Decidable About Hybrid Automata? *Journal of Computer and System Sciences*, **57**:94–124, 1998.
13. T.A. Henzinger, Z. Manna, and A. Pnuelli. What Good Are Digital Clocks? *Proc. ICALP 1992*, LNCS 623, Springer-Verlag, 1992.
14. J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
15. F. Laroussinie, N. Markey and Ph. Schnoebelen. Model Checking Timed Automata with One or Two Clocks. *Proc. CONCUR 2004*, LNCS 3170, Springer-Verlag, 2004.
16. S. Lasota and I. Walukiewicz. Alternating Timed Automata. To appear in *Proceedings of FOSSACS 05*.
17. R. Mayr. Undecidable problems in unreliable computations. *Theoretical Computer Science* 1-3(297):337-354, 2003.
18. J. Ouaknine and J. Worrell. Universality and Language Inclusion for Open and Closed Timed Automata. *Proc. HSCC 2003* LNCS 2623, Springer-Verlag, 2003.
19. J. Ouaknine and J. Worrell. On the Language Inclusion Problem for Timed Automata: Closing a Decidability Gap. *Proc. LICS 2004*, IEEE Computer Society Press, 2004.
20. J. Ouaknine and J. Worrell. On the Decidability of Metric Temporal Logic. Submitted.
21. P. Schnoebelen. Verifying Lossy Channel Systems has Nonprimitive Recursive Complexity. *Information Processing Letters*, 83(5):251–261, 2002.

A Some Proofs

Recall the notion of an *input-bounded Turing machine* $\mathcal{M}$ (also called a Linear Bounded Automaton [14]). The tape alphabet of $\mathcal{M}$ includes two special symbols $\triangleright$ and $\triangleleft$, respectively called the left and right endmarkers. We assume that any computation of $\mathcal{M}$ starts with tape contents of the form $\triangleright x_1 \ldots x_n \triangleleft$. Moreover we require that the instruction set of $\mathcal{M}$ does not include any instructions that try to write over or go beyond the left and right endmarkers.

The *recurrent-state problem (RSP)* for input-bounded Turing machines is as follows. Given an input-bounded Turing machine $\mathcal{M}$ with initial state s_0, does there exist a tape contents γ such that $\mathcal{M}$ has a non-terminating computation, starting in configuration (s_0, γ) and visiting state s_0 infinitely often?

Proposition 8. *The recurrent-state problem for input-bounded Turing machines is undecidable.*

Proof. Given an ordinary Turing machine $\mathcal{N}$, we construct an input-bounded Turing machine $\mathcal{M}$ such that $\mathcal{M}$ has a recurrent computation iff $\mathcal{N}$ halts on the empty word (i.e., when given a blank tape as input).

Starting in its initial control state s_0 with tape contents $\triangleright x_1 \ldots x_n \triangleleft$, $\mathcal{M}$ replaces each tape symbol x_i with a blank, ending up in control state s_1. From state s_1, $\mathcal{M}$ then simulates the behaviour of $\mathcal{N}$ by using the tape cells between the left and right endmarkers to simulate $\mathcal{N}$'s tape. This simulation carries on until either $\mathcal{M}$ runs outs of space, in which case $\mathcal{M}$ just terminates, or $\mathcal{N}$ enters its halting state, in which case $\mathcal{M}$ returns to s_0. If $\mathcal{N}$ halts on the empty word, then $\mathcal{M}$ has a recurrent computation obtained by repeatedly simulating $\mathcal{N}$'s halting computation. Conversely, if $\mathcal{M}$ has a recurrent computation then clearly $\mathcal{N}$ halts on the empty word. □

Proposition 4. *The recurrent-state problem for error-free alternating channel machines is undecidable.*

Proof. We reduce the RSP for input-bounded Turing machines to the RSP for error-free alternating channel machines. Given an input-bounded Turing machine $\mathcal{M}$, we define a channel machine $\mathcal{C}$ that simulates $\mathcal{M}$ in such a way that $\mathcal{C}$ is a positive instance of the RSP for channel machines iff $\mathcal{M}$ is a positive instance of the RSP for Turing machines.

In the simulation, $\mathcal{C}$ stores the tape and the head position of $\mathcal{M}$ on its channel. For each symbol α in the alphabet of $\mathcal{M}$ (including $\triangleright$ and $\triangleleft$), the channel machine $\mathcal{C}$ maintains two symbols α and α^*. Here, α^* indicates that the head is pointing to the current square of the tape. For instance, the string $\triangleright abc^*a \triangleleft$ in the channel of $\mathcal{C}$ encodes a tape $\triangleright abca \triangleleft$ with the head pointing to the square where c resides.

A single step of $\mathcal{M}$ is simulated by a sequence of steps in which $\mathcal{C}$ cycles through its channel by alternately performing read and write operations, updating each tape cell in turn. The set of control states in $\mathcal{C}$ contains both the control states of $\mathcal{M}$ and a number of "intermediate" states which are used to

implement changes in the head position. We distinguish between two kinds of cycles, namely cycles corresponding to steps where the head is moved to the right and those where the head is moved to the left.

First, we consider cycles corresponding to right moves of the head. An instruction[3] $(s_1, a/b, R, s_2)$ is simulated by the transitions depicted in Figure 1. Here we use letters $\alpha \ldots \omega$ to denote the channel alphabet of $\mathcal{M}$. The symbols in the channel are copied one by one until we reach the symbol a^* which is replaced by b. Then the next symbol is converted to its $*$-version, simulating a right move of the head.

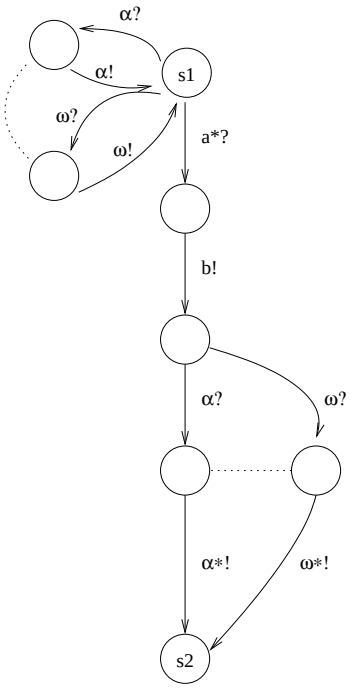


Fig. 1. Simulating a right move $(s_1, a/b, R, s_2)$

Simulating left moves of the head is slightly more complicated since we have to decide the next position of the head before we have figured out its current position. To solve this problem, we guess nondeterministically during the simulation that the next position is the one with the head. If the guess turns out to be wrong, the computation is suspended ($\mathcal{C}$ deadlocks). Otherwise the simulation is continued in a similar manner to above—see Figure 2. □

[3] Such an instruction has its standard interpretation in Turing machines, i.e., the control state is changed from s_1 to s_2, the current cell is changed from a to b, and the head is moved one step to the right.

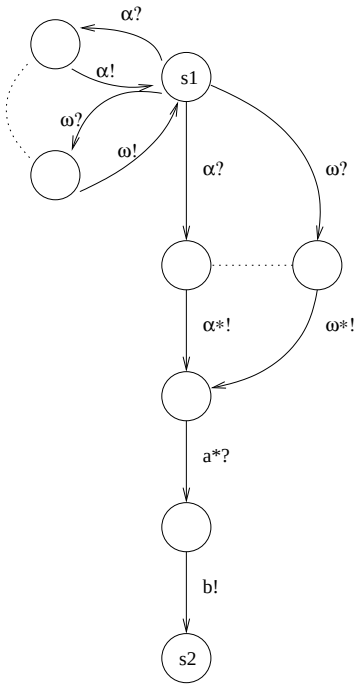


Fig. 2. Simulating a left move $(s_1, a/b, L, s_2)$