Universality and Language Inclusion for Open and Closed Timed Automata*

Joël Ouaknine[1] and James Worrell[2]

[1] Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh PA 15213, USA
joelo@andrew.cmu.edu

[2] Department of Mathematics, Tulane University, New Orleans LA 70118, USA
jbw@math.tulane.edu

Abstract. The algorithmic analysis of timed automata is fundamentally limited by the undecidability of the universality problem. For this reason and others, there has been considerable interest in restricted classes of timed automata. In this paper we study the universality problem for two prominent such subclasses: *open* and *closed* timed automata. This problem is described as open in [6, 8] in the case of open timed automata. We show here that the problem is undecidable for open timed automata over *strongly monotonic* time (no two events are allowed to occur at the same time), and decidable over *weakly monotonic* time. For closed timed automata, we show that the problem is undecidable regardless of the monotonicity assumptions on time. As a corollary, we settle the various language inclusion problems over these classes of timed automata.

1 Introduction

Timed automata were introduced by Alur and Dill [1] and have since become a standard modelling paradigm for real-time systems. Unfortunately, the algorithmic analysis of timed automata is limited by the undecidability of the universality problem (can a given timed automaton perform every timed trace?) [1]. It has also been argued that timed automata provide unrealistic and too powerful expressive power to the system designer. In attempting to address these difficulties, a number of researchers have studied restricted classes of timed automata [6, 2, 4, 5].

* The first author was supported by the Defense Advanced Research Project Agency (DARPA) and the Army Research Office (ARO) under contract no. DAAD19-01-1-0485, and the Office of Naval Research (ONR) under contract no. N00014-95-1-0520. The second author was supported by ONR and NSF. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, ARO, ONR, NSF, the U.S. Government or any other entity.

Two prominent such subclasses are *open* timed automata (all clock constraints must be strict, as in $x < 3$ as opposed to $x \leqslant 3$) and *closed* timed automata (clock constraints are non-strict). Open timed automata have the desirable property of being 'acceptance-robust': whenever they accept a timed trace, they also accept all neighbouring traces that are sufficiently 'close' to the trace in question. Closed timed automata, on the other hand, are 'rejection-robust', in that rejected traces are stable under small temporal perturbations. Closed timed automata also precisely correspond to the finite-state fragment of Timed CSP [10], and can conservatively approximate mixed timed automata with infinitesimal 'precision' [6, 9]. For this reason, they are very often used in practice [4, 3]. In addition, open and closed timed automata have certain complementary 'digitization' properties which can prove extremely valuable to the efficient algorithmic analysis of their behaviour [7, 3, 4].

In this paper, we study the universality problem for both these important classes of timed automata. This problem is described as open in [6, 8] in the case of open timed automata. We show here that it is undecidable if the underlying dense-time domain is *strongly monotonic* (no two events can occur at the same time), and decidable (for open timed automata) if the dense-time domain is *weakly monotonic* (several events are allowed to occur simultaneously). This is a rather surprising result as most researchers usually regard the monotonicity assumptions on time as unimportant, and indeed often pick one and never look back. In the case of closed timed automata, we show that the universality problem is undecidable regardless of the dense-time domain used.

Alur and Dill's original proof of the undecidability of the universality problem for timed automata encodes the halting computations of a Turing-complete machine M as a set of timed traces $L_{\mathrm{AD}}(M)$, and then shows that the complement of this language can be captured by some timed automaton [1]. As noted in [5, 8], this encoding is quite fragile, and requires the timed automaton to differentiate points in time with infinite precision. This, of course, can be difficult to achieve with either exclusively open or exclusively closed clock constraints. Nonetheless, we show here that, by at most doubling the number of clocks used, open timed automata are still powerful enough to capture the required undecidable language, under the semantic assumption of strongly monotonic time.

The reason this device breaks down over weakly monotonic time is that it is impossible to construct an open timed automaton which captures precisely all timed traces that fail to be strongly monotonic, in other words all timed traces that have at least two events occurring at the same time. And indeed, digitization techniques [7] can be used to show that the universality problem is decidable in that case.

Alur and Dill's construction is unfortunately not directly applicable when it comes to closed timed automata. The reason is that closed timed automata only accept languages that are closed in the 'd-topology' [6], whereas the language meant to be captured (the complement of $L_{\mathrm{AD}}(M)$) is not d-closed. A similar problem was noticed by Henzinger and Raskin in the context of robust timed automata. In particular they established the undecidability of the universality problem for robust timed automata and tube languages using a variant of $L_{\mathrm{AD}}(M)$ stable under small temporal perturbations [8]. Unfortunately, when interpreted over mere timed traces, the language complement they define again fails to be d-closed. Nonetheless, we were able to utilize their ideas to define a suitable language having d-closed complement and which can be captured by a closed timed automaton. The basic trick is to alter the definition of $L_{\mathrm{AD}}(M)$ so that events have strictly positive durations; this is achieved by having separate signals explicitly denote the beginning and end of a previously instantaneous event. These delimiters are then required to lie in certain iteratively defined open sets, yielding a language with d-closed complement. This establishes that the universality problem for closed timed automata is indeed undecidable.

These results enable us to settle the various language inclusion problems (does a timed automaton accept all the timed traces of another one?) over these classes of timed automata. It turns out that the only decidable instance is whether the language of a closed timed automaton is a subset of that of an open timed automaton, when interpreted over weakly monotonic time.

2 Timed Automata and Timed Traces

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

$$\phi ::= \mathbf{true} \mid x < k \mid x \leqslant k \mid x > k \mid x \geqslant k \mid \phi \wedge \phi \mid \phi \vee \phi \ .$$

Definition 1. *A (mixed) timed automaton is a tuple* $(\Sigma, S, S_0, S_f, C, E)$*, where*

- Σ *is a finite alphabet of events,*
- S *is a finite set of locations,*
- $S_0 \subseteq S$ *is a set of start locations,*
- $S_f \subseteq S$ *is a set of final locations,*
- C *is a finite set of clocks, and*
- $E \subseteq S \times S \times \Phi_C \times \Sigma \times \mathcal{P}(C) \times \Phi_C$ *is a finite set of transitions. A transition* $(s, s', \phi, a, R, \phi')$ *allows a jump from location* s *to* s'*, communicating event* $a \in \Sigma$ *in the process, 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. We assume that all clocks appearing in* ϕ' *are in* R*, and that* ϕ' *is satisfiable.*

An open timed automaton is a timed automaton in which all pre- and post-conditions $\phi, \phi' \in \Phi_C$ *on edges are open, i.e., are generated by the grammar*

$$\phi ::= \mathbf{true} \mid x < k \mid x > k \mid \phi \wedge \phi \mid \phi \vee \phi \ .$$

A closed timed automaton is a timed automaton in which all pre- and post-conditions on edges are closed, i.e., are generated by the grammar

$$\phi ::= \mathbf{true} \mid x \leqslant k \mid x \geqslant k \mid \phi \wedge \phi \mid \phi \vee \phi \ .$$

Remark 2. Our definitions of mixed, open, and closed timed automata follow [6, 8]. One however finds many variants in the literature: allowing direct comparisons between clocks, e.g., $x - y > k$; allowing rational, rather than integral, bounds in constraints; including invariant clock constraints on locations; allowing clocks to be reset to zero only; considering infinite trace semantics with Büchi or Muller acceptance conditions, rather than finite traces as we do in this paper. It is however not difficult to verify that all the results presented here extend straightforwardly to any combination of these variants.

A *clock interpretation* is a function $\nu : C \longrightarrow \mathbb{R}^+$, where $\mathbb{R}^+$ stands for the non-negative real numbers. If $t \in \mathbb{R}^+$, we let $\nu + t$ be the clock interpretation such that $(\nu + t)(x) = \nu(x) + t$ for all $x \in C$.

A *state* is a triple (s, t, ν), where $s \in S$ is a location, $t \in \mathbb{R}^+$ is the global time elapsed since the automaton was switched on, and ν is a clock interpretation.

A *run* of a timed automaton $A = (\Sigma, S, S_0, S_f, C, E)$ is a finite alternating sequence of states and transitions $e = (s_0, t_0, \nu_0) \xrightarrow{\alpha_1} (s_1, t_1, \nu_1) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (s_n, t_n, \nu_n)$, with the t_i's non-decreasing, and each state (s_i, t_i, ν_i) recording the data immediately following the previous transition $\alpha_i = (s_{i-1}, s_i, \phi_i, a_i, R_i, \phi'_i) \in E$. In addition,

1. $s_0 \in S_0$ and $t_0 = 0$.
2. For all $0 \leqslant i \leqslant n-1$: $\nu_i + (t_{i+1} - t_i)$ satisfies ϕ_{i+1}, $\nu_{i+1}(x) = \nu_i(x) + (t_{i+1} - t_i)$ for all $x \in C \setminus R_{i+1}$, and ν_{i+1} satisfies ϕ'_{i+1}.
3. $s_n \in S_f$.

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 trace* is a finite sequence of timed events with non-decreasing timestamps. The set of all timed traces is denoted **WMTT**, which stands for 'Weakly Monotonic Timed Traces'. The set of all timed traces in which all timestamps are strictly positive and no two timed events have the same timestamp is denoted **SMTT** (the 'S' stands for 'Strongly').

If $u = \langle (t_1, a_1), (t_2, a_2), \ldots, (t_n, a_n) \rangle$ is a timed trace, we define an operator $\mathsf{untime}(u) \mathrel{\widehat{=}} a_1 a_2 \ldots a_n$ which removes all timestamps from u, retaining only the relative order of events. This operator extends to sets of timed traces in the obvious way.

Given a run $e = (s_0, t_0, \nu_0) \xrightarrow{\alpha_1} (s_1, t_1, \nu_1) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (s_n, t_n, \nu_n)$, we produce an associated timed trace $\mathsf{tt}(e) = \langle (t_1, a_1), (t_2, a_2), \ldots, (t_n, a_n) \rangle$, where each a_i is the event component of the transition α_i.

Finally, we define the following two trace semantics for timed automata: $\mathcal{W}[\![A]\!] \mathrel{\widehat{=}} \{\mathsf{tt}(e) \mid e \text{ is a run of } A\}$ represents the set of dense-time weakly monotonic timed traces of A, whereas $\mathcal{S}[\![A]\!] \mathrel{\widehat{=}} \mathcal{W}[\![A]\!] \cap \mathbf{SMTT}$ denotes the set of dense-time strongly monotonic timed traces of A.

3 Regions, Digitization, and Topology

In this section we review the region automaton construction of Alur and Dill [1], the digitization results of Henzinger, Manna, and Pnueli [7], and the d-topology of Jagadeesan, Henzinger, and Gupta [6].

3.1 Region Automata

Let $A = (\Sigma, S, S_0, S_f, C, E)$ be a timed automaton. Let k be the largest integer constant appearing in any of the clock constraints associated with the transitions of A. We define an equivalence relation $\sim$ on the set of clock interpretations as follows: $\nu \sim \nu'$ if

1. For all clocks $x \in C$, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$, or both $\nu(x)$ and $\nu'(x)$ are greater than k.
2. For all $x, y \in C$ with $\nu(x), \nu(y) \leqslant k$, we have $\mathit{fract}(\nu(x)) \leqslant \mathit{fract}(\nu(y)) \Leftrightarrow \mathit{fract}(\nu'(x)) \leqslant \mathit{fract}(\nu'(y))$.
3. For all $x \in C$ with $\nu(x) \leqslant k$, $\mathit{fract}(\nu(x)) = 0 \Leftrightarrow \mathit{fract}(\nu'(x)) = 0$.

It is easy to check that $\sim$ partitions the set of clock interpretations into finitely many equivalence classes, termed *clock regions*.

We define a partial order $\preccurlyeq$ on clock regions as follows: $r \preccurlyeq r'$ if, for any $\nu \in r$, there exists a non-negative real $t \in \mathbb{R}^+$ such that $\nu + t \in r'$. We also define a transitive and antisymmetric relation $\prec$ on clock regions in the same way except that t is required to be strictly positive. Note that $r \prec r$ for some, but not all, clock regions r.

We now define the *(weakly monotonic) region automaton* $\mathit{WREG}(A)$ of A as follows. Its alphabet is the same as that of A, Σ. The states of $\mathit{WREG}(A)$ consist of all pairs (s, r), where $s \in S$ is a location of A and r is a clock region of A. The start states of $\mathit{WREG}(A)$ consist of all states of the form (s_0, r), where $s_0 \in S_0$, and its accepting states consist of all states of the form (s_f, r), where $s_f \in S_f$. $\mathit{WREG}(A)$ has a transition $(s, r) \xrightarrow{a} (s', r')$ provided there exist a clock region $r'' \succcurlyeq r$ and an A-transition $(s, s', \phi, a, R, \phi') \in E$ such that all clock interpretations in r'' satisfy ϕ, r'' and r' agree when restricted to clocks not belonging to R, and all clock interpretations in r' meet ϕ'.

We also define the *(strongly monotonic) region automaton* $\mathit{SREG}(A)$ in exactly the same way except that we replace the relation $\preccurlyeq$ with $\prec$.

The (untimed) languages accepted by $\mathit{WREG}(A)$ and $\mathit{SREG}(A)$ are denoted $[\![\mathit{WREG}(A)]\!]$ and $[\![\mathit{SREG}(A)]\!]$ respectively.

We now have:

Theorem 3 (Alur and Dill [1]). *For any timed automaton A, $[\![\mathit{WREG}(A)]\!] = \mathsf{untime}(\mathcal{W}[\![A]\!])$ and $[\![\mathit{SREG}(A)]\!] = \mathsf{untime}(\mathcal{S}[\![A]\!])$.*

We refer the reader to [1] for the proof.

The *emptiness problem* is to decide whether the set of timed traces of a timed automaton is empty. As an immediate consequence of Theorem 3, the emptiness problem for timed automata over either weakly or strongly monotonic time is decidable.

The *reachability problem* is to decide, given an arbitrary fixed event, whether a timed automaton has at least one trace in which this event occurs. This problem is equivalent to the emptiness problem, and is thus always decidable.

3.2 Digitization

Let $t \in \mathbb{R}^+$ and let $0 \leqslant \varepsilon \leqslant 1$ be real numbers. If $\mathit{fract}(t) < \varepsilon$, let $[t]_\varepsilon \mathrel{\widehat{=}} \lfloor t \rfloor$, otherwise let $[t]_\varepsilon \mathrel{\widehat{=}} \lceil t \rceil$.

We can then extend $[\cdot]_\varepsilon$ to timed traces by pointwise application to the timestamps of the trace's events. We then further extend $[\cdot]_\varepsilon$ to sets of timed traces in the usual way.

Definition 4. *Let T be a set of timed traces.*

*T is closed under digitization if, for any $0 \leqslant \varepsilon \leqslant 1$, $[T]_\varepsilon \subseteq T$. T is closed under inverse digitization if, whenever a timed trace $u \in$ **WMTT** is such that $[u]_\varepsilon \in T$ for all $0 \leqslant \varepsilon \leqslant 1$, then $u \in T$.*

For A a timed automaton, the above definitions apply to $\mathcal{W}[\![A]\!]$.

For $T \subseteq$ **WMTT** a set of timed traces, let $\mathbb{Z}(T)$ be the set of all integral timed traces of T, i.e., those timed traces in T all of whose events have integral timestamps.

The main digitization result is as follows:

Theorem 5 (Henzinger, Manna, and Pnueli [7]). *Let T be a set of timed traces closed under digitization, and let T' be a set of timed traces closed under inverse digitization. Then $T \subseteq T'$ if and only if $\mathbb{Z}(T) \subseteq \mathbb{Z}(T')$.*

The right-to-left implication is trivial. For the positive direction, let $u \in T$. Since T is closed under digitization, $[u]_\varepsilon \in T$ for any ε. However $\mathbb{Z}(T) \subseteq \mathbb{Z}(T')$, thus $[u]_\varepsilon \in T'$ for any ε. Since T' is closed under inverse digitization, $u \in T'$ as required.

Observe that integral timed traces over alphabet Σ are in natural one-to-one correspondence with untimed traces over alphabet $\Sigma \cup \{\checkmark\}$, where the event $\checkmark \notin \Sigma$ represents the passage of one time unit. For T a set of integral timed traces, we write $T^{\checkmark}$ for the corresponding unique set of untimed $\checkmark$-traces.

Proposition 6. *Let A be a timed automaton. Then $(\mathbb{Z}(\mathcal{W}[\![A]\!]))^{\checkmark}$ is a regular language. In other words, the integral timed traces of A can essentially be generated by an untimed finite automaton.*

The required untimed automaton can be obtained by a straightforward modification of the region automaton $\mathit{WREG}(A)$. The construction is identical but for transitions. Postulate a transition $(s, r) \xrightarrow{a} (s', r')$ of the untimed automaton if there is a transition $(s, s', \phi, a, R, \phi')$ of A such that all clock interpretations in r satisfy ϕ, r and r' agree when restricted to clocks not belonging to R, and all clock interpretations in r' meet ϕ'. In addition, postulate a transition $(s, r) \xrightarrow{\checkmark} (s, r')$ of the untimed automaton if, for all clock interpretations $\nu \in r$, $\nu + 1 \in r'$. It is easily checked that this untimed automaton accepts precisely $(\mathbb{Z}(\mathcal{W}[\![A]\!]))^{\checkmark}$, as required.

Corollary 7. *Let A and B be timed automata with A closed under digitization and B closed under inverse digitization. Then the timed language inclusion problem of whether $\mathcal{W}[\![A]\!] \subseteq \mathcal{W}[\![B]\!]$ is decidable.*

We will also make use of the following result:

Proposition 8. *Closed timed automata are closed under digitization, and open timed automata are closed under inverse digitization.*

Our proof follows that presented in [7].

Let us first consider the case in which A is a closed timed automaton. Let $e = (s_0, t_0, \nu_0) \xrightarrow{\alpha_1} (s_1, t_1, \nu_1) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (s_n, t_n, \nu_n)$ be a run of A, with $\alpha_j = (s_{j-1}, s_j, \phi_j, a_j, R_j, \phi'_j)$. Observe that, for any clock x and index j, $\nu_j(x) = t_j - t_i + r_i(x)$, where $i \leqslant j$ is the index of the last transition which reset clock x (or is 0 if x was never reset), and $r_i(x)$ is the nondeterministic value that x was reset to. For $i \geqslant 1$, $r_i(x)$ must satisfy the closed postcondition $\phi'_i(x)$.

To show that A is closed under digitization, it suffices to show, given $0 \leqslant \varepsilon \leqslant 1$, that the prospective run $e' = (s_0, [t_0]_\varepsilon, \nu'_0) \xrightarrow{\alpha_1} (s_1, [t_1]_\varepsilon, \nu'_1) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (s_n, [t_n]_\varepsilon, \nu'_n)$ is a valid run of A. Here $\nu'_j(x) = [t_j]_\varepsilon - [t_i]_\varepsilon + r'_i(x)$, where the indices j and i are obtained from e as explained above, and the $r'_i(x)$'s are carefully chosen as we now explain. For e' to be a valid run of A, each reset value $r'_i(x)$ and clock interpretation ν'_i must meet the relevant closed pre- and postconditions. Since by assumption the r_i's and ν_i's do meet these constraints, it suffices to show that one can choose each $r'_i(x)$ subject to: (i) $r_i(x) \leqslant k \Rightarrow r'_i(x) \leqslant k$ and $r_i(x) \geqslant k \Rightarrow r'_i(x) \geqslant k$, for any integer k, and (ii) $\nu_j(x) \leqslant k \Rightarrow \nu'_j(x) \leqslant k$ and $\nu_j(x) \geqslant k \Rightarrow \nu'_j(x) \geqslant k$, again for any integer k.

First observe that, for any integer k, any real numbers p and q, and any ε, $p - q \leqslant k \Rightarrow [p]_\varepsilon - [q]_\varepsilon \leqslant k$, and $p - q \geqslant k \Rightarrow [p]_\varepsilon - [q]_\varepsilon \geqslant k$.

Now choose $r'_i(x) = [t_i]_\varepsilon - [t_i - r_i(x)]_\varepsilon$. By the above, $r'_i(x)$ clearly satisfies (i). For any index j, we also have $\nu'_j(x) = [t_j]_\varepsilon - [t_i]_\varepsilon + r'_i(x) = [t_j]_\varepsilon - [t_i - r_i(x)]_\varepsilon$ by definition of $r'_i(x)$. Since $\nu_j(x) = t_j - (t_i - r_i(x))$, our earlier observation implies that (ii) must too be satisfied, as required.

We now tackle the case in which A is an open timed automaton. We establish the stronger claim that whenever $u \in \mathbf{WMTT}$ is a timed trace such that $[u]_0 \in \mathcal{W}[\![A]\!]$, then $u \in \mathcal{W}[\![A]\!]$. Thus consider $u = \langle (t_1, a_1), (t_2, a_2), \ldots, (t_n, a_n) \rangle \in \mathbf{WMTT}$ such that $[u]_0 = \lceil u \rceil \in \mathcal{W}[\![A]\!]$. $\lceil u \rceil$ must originate from a run $e = (s_0, \lceil t_0 \rceil, \nu_0) \xrightarrow{\alpha_1} (s_1, \lceil t_1 \rceil, \nu_1) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (s_n, \lceil t_n \rceil, \nu_n)$ of A. For each clock x, let $\nu_j(x) = \lceil t_j \rceil - \lceil t_i \rceil + r_i(x)$ as above. We must show that the prospective run $e' = (s_0, t_0, \nu'_0) \xrightarrow{\alpha_1} (s_1, t_1, \nu'_1) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (s_n, t_n, \nu'_n)$ is a valid run of A, where $\nu'_j(x) = t_j - t_i + r'_i(x)$, for suitable values of $r'_i(x)$.

Choose δ a strictly positive real number such that $\delta < \mathit{fract}(t_j)$ for every non-integral t_j. Let $r'_i(x) = \mathit{fract}(t_i) + \lfloor r_i(x) \rfloor - \delta$ if t_i is not integral, and let $r'_i(x) = \lceil r_i(x) \rceil - \delta$ otherwise. A simple case analysis (considering integral and non-integral cases for t_i, t_j, and $r_i(x)$ as needed) establishes the following facts: (i) $r_i(x) < k \Rightarrow r'_i(x) < k$ and $r_i(x) > k \Rightarrow r'_i(x) > k$, for any integer k, and (ii) $\nu_j(x) < k \Rightarrow \nu'_j(x) < k$ and $\nu_j(x) > k \Rightarrow \nu'_j(x) > k$, again for any integer k. This shows that e' is a valid run of A and completes the proof.

3.3 The d-Topology

Define a metric d on **WMTT** as follows. For $u = \langle(t_1, a_1), \ldots, (t_n, a_n)\rangle$ and $u' = \langle(t'_1, a'_1), \ldots, (t'_m, a'_m)\rangle$ two timed traces, if $\mathsf{untime}(u) \neq \mathsf{untime}(u')$, then $d(u, u') \mathrel{\widehat{=}} \infty$. Otherwise, $d(u, u') \mathrel{\widehat{=}} \max\{|t_i - t'_i| : 1 \leqslant i \leqslant n\}$.

Proposition 9. *The semantic mapping $\mathcal{W}[\![\cdot]\!]$ takes open timed automata to d-open sets of timed traces, and closed timed automata to d-closed sets of timed traces.*

The assertion concerning open timed automata appears in [6], with the following proof. Let A be an open timed automaton, and consider a run $e = (s_0, t_0, \nu_0) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} (s_n, t_n, \nu_n)$ that accepts the timed trace u. Since all clock constraints are open, for each $0 \leqslant i \leqslant n$, there is an $\varepsilon_i > 0$ such that substituting $\nu_i - \varepsilon_i$ or $\nu_i + \varepsilon_i$ (or any clock interpretation in between) for ν_i in e still gives a valid run of A. Let $\varepsilon = \min\{\varepsilon_i/2 \mid 0 \leqslant i \leqslant n\}$. It is clear that any timed trace within ε of u can be accepted by A.

Let us now consider the case of a closed timed automaton A. Let u be any timed trace, and let $\langle u_i\rangle_{i\geqslant 1}$ be a sequence of timed traces in $\mathcal{W}[\![A]\!]$ converging to u. Without loss of generality, since A has only finitely many transitions, we can assume that the runs e_i corresponding to these timed traces share the same transitions, in the same order. The reset sets associated with these transitions are required to be closed; we may assume that they are bounded as well (if they are not, pick an artificial bound that is large enough not to disrupt anything). The sequence of e_i's therefore essentially lies in a compact subset of $\mathbb{R}^n$ (for some finite n) and must therefore have an accumulation point e. The run e is clearly a valid run of A, since its clock interpretations are limits of clock interpretations of the e_i's, and the constraints these must satisfy are all closed. It is also plain that the run e gives rise to the timed trace u, so that $u \in \mathcal{W}[\![A]\!]$ as required.

4 Universality

The *universality problem* is to decide whether a timed automaton can perform all possible timed traces. In our framework, this problem gives rise to six subcases, which depend on the class of automata considered (mixed, open, or closed), as well as on the semantic assumptions on the dense-time domain (weakly or strongly monotonic time). In the case of mixed timed automata and either weakly or strongly monotonic dense time, this problem was shown to be undecidable in [1]. We now address the remaining cases.

A *two-counter machine* M is a triple $(\{b_0, b_1, \ldots, b_k\}, C, D)$, where the b_i's are instructions and C and D are two counters ranging over the non-negative integers. Both counters are initially empty, and the first instruction M executes is b_0. Each instruction b_i, for $i < k$, either: (i) increments or decrements (if nonzero) one of the counters, and subsequently jumps to the next instruction, or (ii) tests one of the counters for emptiness and conditionally jumps to the next instruction. The instruction b_k represents successful termination. A *configuration* of M is a triple (b_i, c, d), where c and d are the respective values of the counters C and D. A *halting computation* of M is a finite sequence of configurations starting with $(b_0, 0, 0)$ and ending with a b_k-configuration, subject to the constraint that each successive configuration be a valid successor of the previous one. The problem of deciding whether a two-counter machine has a halting computation is undecidable.

Let M be a two-counter machine. Following [1], we define a set of timed traces $L_{\mathrm{AD}}(M)$ as follows. Given any halting computation $\langle(b_{i_0}, c_0, d_0), (b_{i_1}, c_1, d_1), \ldots, (b_{i_n}, c_n, d_n)\rangle$ of M, we include in $L_{\mathrm{AD}}(M)$ the following timed trace u over the alphabet $\Sigma = \{b_0, b_1, \ldots, b_k, c, d\}$: $\mathsf{untime}(u) = b_{i_0} c^{c_0} d^{d_0} b_{i_1} c^{c_1} d^{d_1} \ldots b_{i_n} c^{c_n} d^{d_n}$; u is strongly monotonic (no two events occur at the same time); the timestamp of b_{i_j} is $j + t_0$, where t_0 is the timestamp of b_{i_0}; for all $0 \leqslant j \leqslant n-1$: (i) if $c_{j+1} = c_j$, then for each timed event (t, c) in the time interval $(j, j+1) + t_0$, there is a timed event $(t+1, c)$ in the time interval $(j+1, j+2) + t_0$; (ii) if $c_{j+1} = c_j + 1$, then for every (t, c) in the time interval $(j+1, j+2) + t_0$, except the last one, there is a timed event $(t-1, c)$; (iii) if $c_{j+1} = c_j - 1$, then for every (t, c) in the time interval $(j, j+1) + t_0$, except the last one, there is a timed event $(t+1, c)$; (iv) the same requirements hold of the d's.

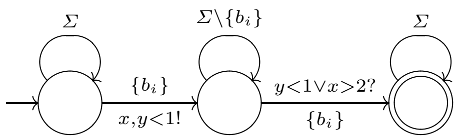
By construction, M has a halting computation if and only if $L_{\mathrm{AD}}(M) \neq \emptyset$, which is equivalent to $\mathbf{SMTT} \setminus L_{\mathrm{AD}}(M) \neq \mathbf{SMTT}$. We now show that there exists an open timed automaton A such that $\mathcal{S}[\![A]\!] = \mathbf{SMTT} \setminus L_{\mathrm{AD}}(M)$. The following theorem then immediately follows.

Theorem 10. *The universality problem for open timed automata over strongly monotonic dense time is undecidable. In other words, given an open timed automaton A, it is undecidable whether $\mathcal{S}[\![A]\!] = \mathbf{SMTT}$.*

It remains to exhibit said open automaton A. The construction we sketch is similar to that of [1]; even though the automaton they construct is not open, with a little care and a few additional clocks it is possible to produce an open automaton A such that $\mathcal{S}[\![A]\!] = \mathbf{SMTT} \setminus L_{\mathrm{AD}}(M)$.

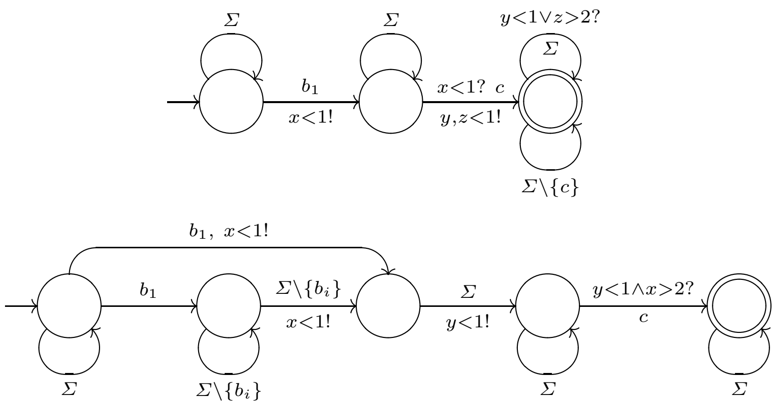
Since open automata are trivially closed under finite unions, it is sufficient to exhibit a collection of open automata, each of which accepts timed traces not in $L_{\mathrm{AD}}(M)$, and the sum total of which accepts all such traces. We illustrate two cases; the remainder are left to the reader.

The automaton below accepts exactly those timed traces u of the form $\mathsf{untime}(u) = (\Sigma \setminus \{b_i\})^* b_{i_0} (\Sigma \setminus \{b_i\})^* b_{i_1} (\Sigma \setminus \{b_i\})^* \ldots b_{i_n} (\Sigma \setminus \{b_i\})^*$ with the property that some b_{i_j} fails to occur j time units after the occurrence of b_{i_0}. In what follows, start locations are depicted with an incoming arrow not originating from any other location, and final locations are doubly circled. Preconditions are decorated with a question mark (?), and postconditions with an exclamation mark (!). An edge labelled with a set of events stands for a collection of edges with the same source and target, one for each of the events in the set. The rest of the notation is self-explanatory.



We claim that this automaton accepts exactly the required timed traces. Let u be such a trace, with b_{i_0} occurring at time t_0. There must be some m such that b_{i_j} occurred at time $j + t_0$ for all $j < m$, but b_{i_m} occurred at some time other than $m+t_0$. If the occurrence of b_{i_m} was too early, that can be captured with the clock y reset to 0. If, on the other hand, b_{i_m} occurred too late, then by resetting x sufficiently close to 1 the trace will also be accepted. Of course, if each b_{i_j} did occur exactly one time unit after the previous one, then no assignments of values to x or y can make the automaton accept the trace.

As a second example, suppose that instruction b_1 is meant to leave the value of counter C unchanged. The automaton below accepts precisely those timed traces u in which some instance of b_1 is followed, within one time unit, by a list of c's which cannot be put in one-to-one unit-duration-delayed correspondence with the list of c's in the following unit-duration time interval.



Our claim can be justified as follows. We are asserting one of two things: either there is a c in the unit-duration time interval immediately following b_1 which has no counterpart one time unit later, or vice-versa. The former is captured by the top component, whereas the latter is captured by the bottom component. For simplicity, we are assuming that only traces in which the b_{i_j}'s happen exactly at unit-duration time intervals need be considered (cf. previous automaton). In the bottom automaton, this forces all transitions up to that resetting y to occur within one time unit of the occurrence of b_1.

The other cases are left to the reader. We remark that, while these constructions produce an open automaton A with $\mathcal{S}[\![A]\!] = \mathbf{SMTT} \setminus L_{\mathrm{AD}}(M)$, it is in general not possible to exhibit an open automaton A such that $\mathcal{W}[\![A]\!] = \mathbf{WMTT} \setminus L_{\mathrm{AD}}(M)$. Intuitively, this is because we would need in addition to provide A with the ability to accept any timed trace in which at least two events occurred at the same time. However, no open timed automaton can capture precisely this requirement, since it does not correspond to a d-open set of timed traces. This difficulty turns out to be insuperable, as the following result demonstrates.

Theorem 11. *The universality problem for open timed automata over weakly monotonic dense time is decidable. In other words, there is an algorithm which, given an open timed automaton A, decides whether $\mathcal{W}[\![A]\!] = \mathbf{WMTT}$.*

Indeed, let A be an open timed automaton. Note that deciding whether $\mathcal{W}[\![A]\!] = \mathbf{WMTT}$ is clearly equivalent to deciding whether $\mathbf{WMTT} \subseteq \mathcal{W}[\![A]\!]$. But A is closed under inverse digitization (Proposition 8), and $\mathbf{WMTT}$ is obviously closed under digitization. By Corollary 7, the inclusion $\mathbf{WMTT} \subseteq \mathcal{W}[\![A]\!]$ can therefore be decided.

We now move on to closed timed automata. We begin by stating the chief undecidability result:

Theorem 12. *The universality problem for closed timed automata over either weakly or strongly monotonic dense time is undecidable. In other words, given a closed timed automaton A, whether $\mathcal{W}[\![A]\!] = \mathbf{WMTT}$ and whether $\mathcal{S}[\![A]\!] = \mathbf{SMTT}$ are undecidable.*

Unfortunately, we cannot employ the above method to establish this result, since the complement of $L_{\mathrm{AD}}(M)$ is clearly not a d-closed set; indeed, $L_{\mathrm{AD}}(M)$ is in general not d-open since we require certain events to occur exactly one time unit apart, etc.

Instead, we draw upon a construction of Henzinger and Raskin [8] to manufacture a suitable language.[1] The basic idea is to alter the definition of $L_{\mathrm{AD}}(M)$ so that events have strictly positive durations; this is achieved by having separate signals explicitly denote the beginning and end of a previously instantaneous event. These delimiters are then required to lie in certain iteratively defined open sets. As a result, the complement of this language is then d-closed.

To this end, define a *slot* to be a non-empty open interval of the non-negative real numbers of length less than one. Given $t_1 < t_2 < t_1 + 1$, the slot *between* t_1 and t_2 is the open interval (t_1, t_2), and the slot *generated* by t_1 and t_2 is the open interval $(t_1 + 1, t_2 + 1)$.

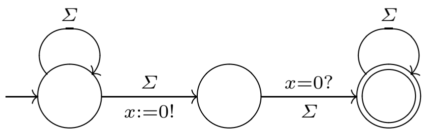
Let $M = (\{b_0, b_1, \ldots, b_k\}, C, D)$ be a two-counter machine. We define a set of strongly monotonic timed traces $L_{\mathrm{HR}}(M)$. The untimed traces of $L_{\mathrm{HR}}(M)$ are the same as those of $L_{\mathrm{AD}}(M)$ except that every event p in $L_{\mathrm{AD}}(M)$ is replaced by a pair of consecutive events pp' in $L_{\mathrm{HR}}(M)$. Configurations are encoded in successive slots. Moreover, whenever there was a requirement in $L_{\mathrm{AD}}(M)$ that some event q should occur exactly one time unit after some event p, this translates

[1] The undecidable language defined by Henzinger and Raskin was also meant to be stable under small temporal perturbations. However, the primary purpose of their construction was to establish the undecidability of universality for robust timed automata and tube languages. When interpreted over mere timed traces, the language complement they define unfortunately fails to be d-closed.

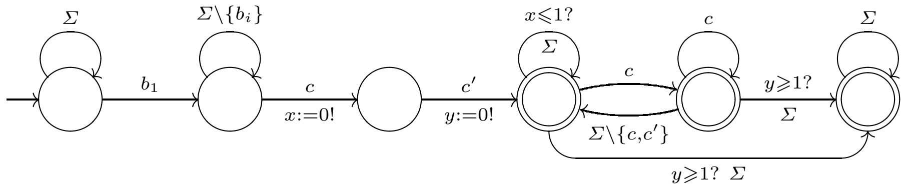
for $L_{\mathrm{HR}}(M)$ into a requirement that the pair qq' appear in the slot generated by the pair pp'. Likewise, $L_{\mathrm{AD}}(M)$ requirements that some event q appear between events p and r translate into $L_{\mathrm{HR}}(M)$ requirements that qq' lie in some slot between pp' and rr'.

It is clear that $L_{\mathrm{HR}}(M)$ is a d-open strongly monotonic subset of **WMTT**, and that M has a halting computation if and only if $L_{\mathrm{HR}}(M) \neq \emptyset$. The latter is equivalent to both $\mathbf{WMTT} \setminus L_{\mathrm{HR}}(M) \neq \mathbf{WMTT}$ and $\mathbf{SMTT} \setminus L_{\mathrm{HR}}(M) \neq \mathbf{SMTT}$. One can now adapt the constructions appearing in the proof of Theorem 10 (or simpler yet the constructions of [1] or [8]) to manufacture a closed timed automaton A such that $\mathcal{W}[\![A]\!] = \mathbf{WMTT} \setminus L_{\mathrm{HR}}(M)$ (from which it also follows that $\mathcal{S}[\![A]\!] = \mathbf{SMTT} \setminus L_{\mathrm{HR}}(M)$). Taken together, these facts establish Theorem 12.

We illustrate two cases of the automaton construction. Let Σ stand for the set $\{b_0, b_0', b_1, b_1', \ldots, b_k, b_k', c, c', d, d'\}$. The automaton below accepts any trace which is not strongly monotonic.



Suppose now that instruction b_1 is meant to leave the value of counter C unchanged. A possible timed trace violating this requirement is one in which the event b_1 is eventually followed, prior to the next instruction, by a pair of events $\langle (t,c),(t',c')\rangle$, with no pair of consecutive events cc' appearing in the slot $(t+1, t'+1)$. (This, of course, handles only one half of the required bijection.) The automaton below accepts all such timed traces.



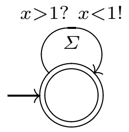
We justify our claim as follows. A pair of consecutive events cc' fails to appear in a slot $(t+1, t'+1)$ precisely when every occurrence of c (if any) after time $t+1$ fails to be immediately followed by a c' before time $t'+1$.

5 Language Inclusion

Theorem 13. *Let A and B be timed automata drawn independently from the classes of mixed, open, or closed timed automata. The language inclusion problem over weakly monotonic dense time as to whether $\mathcal{W}[\![A]\!] \subseteq \mathcal{W}[\![B]\!]$ is only decidable when A is drawn from the class of closed timed automata and B is drawn from the class of open timed automata. The language inclusion problem over strongly monotonic dense time as to whether $\mathcal{S}[\![A]\!] \subseteq \mathcal{S}[\![B]\!]$ is undecidable in all instances.*

Let us first consider the case of weakly monotonic time. The single decidable instance (A closed and B open) follows directly from Proposition 8 and Corollary 7. On the other hand, by choosing A such that $\mathcal{W}[\![A]\!] = \mathbf{WMTT}$ (so that the question $\mathcal{W}[\![A]\!] \subseteq \mathcal{W}[\![B]\!]$ reduces to the universality problem for B), we can dispose of all cases in which B is either mixed or closed (Theorem 12).

For the remaining two cases (B open and A either mixed or open), let A be the following timed automaton:



Notice that A accepts exactly the strongly monotonic timed traces—in other words, $\mathcal{W}[\![A]\!] = \mathbf{SMTT}$. The question $\mathcal{W}[\![A]\!] \subseteq \mathcal{W}[\![B]\!]$ therefore reduces to the universality problem for B over strongly monotonic time, and is therefore undecidable (Theorem 10).

The case of strongly monotonic time is dealt with in similar fashion.

6 Summary

The two tables below summarize the universality and language inclusion results discussed in this paper.

Class of Timed Automata	Universality	
	Weakly Monotonic Time	Strongly Monotonic Time
Mixed	Undecidable	Undecidable
Open	Decidable	Undecidable
Closed	Undecidable	Undecidable

A	B	$\mathcal{W}[\![A]\!] \subseteq \mathcal{W}[\![B]\!]$?	$\mathcal{S}[\![A]\!] \subseteq \mathcal{S}[\![B]\!]$?
Mixed	Mixed	Undecidable	Undecidable
Open	Mixed	Undecidable	Undecidable
Closed	Mixed	Undecidable	Undecidable
Mixed	Open	Undecidable	Undecidable
Open	Open	Undecidable	Undecidable
Closed	Open	Decidable	Undecidable
Mixed	Closed	Undecidable	Undecidable
Open	Closed	Undecidable	Undecidable
Closed	Closed	Undecidable	Undecidable

References

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

[2] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 211:253–273, 1999.

[3] E. Asarin, O. Maler, and A. Pnueli. On discretization of delays in timed automata and digital circuits. In *Proceedings of CONCUR 98*, volume 1466, pages 470–484. Springer LNCS, 1998.

[4] D. Bošnački. Digitization of timed automata. In *Proceedings of FMICS 99*, 1999.

[5] M. Fränzle. Analysis of Hybrid Systems: An ounce of realism can save an infinity of states. In *Proceedings of CSL 99*, volume 1683, pages 126–140. Springer LNCS, 1999.

[6] V. Gupta, T. A. Henzinger, and R. Jagadeesan. Robust timed automata. In *Proceedings of HART 97*, volume 1201, pages 331–345. Springer LNCS, 1997.

[7] T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proceedings of ICALP 92*, volume 623, pages 545–558. Springer LNCS, 1992.

[8] T. A. Henzinger and J.-F. Raskin. Robust undecidability of timed and hybrid systems. In *Proceedings of HSCC 00*, volume 1790, pages 145–159. Springer LNCS, 2000.

[9] J. Ouaknine and J. B. Worrell. Revisiting digitization, robustness, and decidability for timed automata. Submitted, 2003. Available from `www.andrew.cmu.edu/~joelo`.

[10] J. Ouaknine and J. B. Worrell. Timed CSP = closed timed ε-automata. Submitted, 2003. Available from `www.andrew.cmu.edu/~joelo`.