Revisiting Digitization, Robustness, and Decidability for Timed Automata

Joël Ouaknine
Computer Science Department, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh PA 15213, USA
Email: joelo@andrew.cmu.edu

James Worrell
Department of Mathematics, Tulane University
6823 St. Charles Ave., New Orleans LA 70118, USA
Email: jbw@math.tulane.edu

Abstract

We consider several questions related to the use of digitization techniques for timed automata. These very successful techniques reduce dense-time language inclusion problems to discrete time, but are applicable only when the implementation is closed under digitization and the specification is closed under inverse digitization. We show that, for timed automata, the former (whether the implementation is closed under digitization) is decidable, but not the latter. We also investigate digitization questions in connection with the robust semantics for timed automata. The robust modelling approach introduces a timing fuzziness through the semantic removal of equality testing. Since its introduction half a decade ago, research into the robust semantics has suggested that it yields roughly the same theory as the standard semantics. This paper shows that, surprisingly, this is not the case: the robust semantics is significantly less tractable, and differs from the standard semantics in many key respects. In particular, the robust semantics yields an undecidable (non-regular) discrete-time theory, in stark contrast with the standard semantics. This makes it virtually impossible to apply digitization techniques together with the robust semantics. On the positive side, we show that the robust languages of timed automata remain recursive.

1. Introduction

Timed automata were introduced by Alur and Dill [2] and have since become a standard modelling paradigm for real-time systems. Unfortunately, the algorithmic analysis of timed automata is limited both by the (PSPACE-complete) complexity of the emptiness problem (does a given timed automaton accept any timed trace?), and by the undecidability of the universality problem (does a given timed automaton accept every timed trace?) [2]. Several attempts have been made to circumvent these difficulties, often by restricting or altering either the syntax or the semantics of timed automata—see, e.g., [4], [15], [3], [13].

Digitization techniques [17], [9], [8], [24], [7], [10], [20], [21] have proved to be one of the most successful theoretical and practical advances in tackling the problems at hand. Under appropriate conditions, digitization techniques reduce the dense-time language inclusion problem between two timed automata to discrete time, not only rendering it decidable but also drastically speeding-up the verification process (see [12], [5], [11], [25]). The twin prerequisites are that the implementation be 'closed under digitization', and that the specification be 'closed under inverse digitization'.

It has often been noted that for real-world modelling and verification tasks, the restriction to implementations closed under digitization is relatively unobtrusive, since any timed automaton can always be safely infinitesimally over-approximated by one that is closed under digitization (cf. Propositions 14 and 30). On the other hand, the most common specifications on timed systems happen to be closed under inverse digitization [17]. Nevertheless, it is clearly a problem of significant theoretical and practical importance to be able to determine whether a given timed automaton is closed under digitization and/or closed under inverse digitization. One of the contributions of this paper is a proof that the former is decidable, but not the latter.

Another (related) complaint registered against the Alur-Dill timed automata paradigm is the 'excessive' expressive power timed automata derive through their ability to differentiate points in time with infinite precision (see, e.g., [15], [23], [13], [6], [14]). As a remedy, a robust semantics for timed automata was proposed in [15], whereby a particular timed trace is accepted by a timed automaton if and only if neighbouring traces are also accepted. It was originally believed that the robust semantics might have a decidable universality problem, but this was finally disproved in [18]. Up to now, the consensus has since been that the robust semantics yields roughly the same theory as the standard one [16], [15], [18]. In this paper, we show that, surprisingly, this is not the case: the robust semantics is significantly less tractable, and differs from the standard semantics in many key respects, as we now detail.

The robust semantics for timed automata is based on a metric on timed traces (sequences of visible events with real-valued timestamps) whereby two timed traces are 'close' if each can be obtained from the other by slight temporal perturbations of the events' timestamps. In the associated topology, open sets of timed traces are termed 'tubes'. The original formulation of the robust semantics assigned sets of tubes to timed automata, rather than sets of timed traces as in the standard (or precise) semantics. It was postulated that a timed automaton should accept a given tube if and only if the set of timed traces it accepted under the precise semantics was dense in the tube.

In order to facilitate the comparison between the robust and

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.

precise semantics, it is preferable to work in a common setting. We achieve this by considering the largest tube accepted by a timed automaton, which we call the 'robust language' of the automaton. Since an individual timed trace lies in the robust language if and only if it belongs to some accepted tube, the two formulations are equivalent.

Following this interpretation, we discover that the robust languages of timed automata closely resemble the precise languages of open timed automata. (A timed automaton is 'open' if all its clock constraints consist of positive boolean combinations of strict inequalities, as in $x < 3$ rather than $x \leqslant 3$.) In both cases, whenever a timed trace is accepted, so are all other neighbouring timed traces that are sufficiently close. As pointed out in [18], the similarity between timed automata under the robust semantics and open timed automata can be made quite precise: the former restricts timed automata through the semantic removal of (exact) equality testing, whereas the latter restricts timed automata through the syntactic removal of equality. A legitimate question is therefore whether the expressive powers of these two formalisms are comparable. We show here, that, surprisingly, the answer is negative: there are timed automata whose robust languages cannot be captured as the precise language of any timed automaton (and vice-versa).

The undecidability of the robust universality problem for timed automata was established in [18], but the related matter of the universality problem for open timed automata was left there as an open question. Very recently, we showed that the answer depends on the monotonicity assumptions on time [22]; more precisely, if time is assumed to be 'strongly monotonic', in that no two events are allowed to occur at the same time, then the universality problem for open timed automata is undecidable. On the other hand, the problem becomes decidable if time is assumed to be weakly monotonic. The crux of this decidability result resides in the fact that, over weakly monotonic time, open timed automata are closed under inverse digitization.[1] Note that the undecidability of the robust universality problem quoted above was obtained over strongly monotonic time; since (as we shall see) robust timed automata are also closed under inverse digitization over weakly monotonic time, it seemed very likely that the corresponding universality problem should too be decidable. We were very surprised to discover that this was not the case.

Our ensuing investigation into the failure of digitization techniques for timed automata under the robust semantics revealed that the robust discrete-time languages of timed automata are in general not regular; in particular, we show that the robust discrete-time universality problem is undecidable, contrary to the situation under the precise semantics.

We summarize the contributions of this paper as follows. We study two decision problems underlying the applicability of digitization techniques for timed automata: given a timed automaton, we show that it is decidable whether the automaton is closed under digitization, and undecidable whether it is closed under inverse digitization. We also show that these results are reversed under the robust semantics. We show that the expressive powers of timed automata under the standard and robust semantics respectively are incomparable. More generally, we find that the robust semantics yields an undecidable discrete-time theory, in stark contrast with the standard semantics. As a result, it appears to be impossible to combine digitization techniques with the robust semantics. On the positive side, we show that the robust languages of timed automata remain recursive.

[1] Digitization makes little sense over strongly monotonic time.

2. Timed Automata

We define timed automata together with their standard and robust trace semantics, along the lines of [2], [15]. We also define the d-topology, which is used heavily in the remainder of this paper.

2.1. Syntax

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).

$$\phi ::= x < k \mid x \leqslant k \mid x > k \mid x \geqslant k \mid \phi \wedge \phi \mid \phi \vee \phi \ .$$

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 accepting locations,
- C is a finite set of clocks, and
- $E \subseteq S \times S \times \Phi_C \times \Sigma \times \mathcal{P}(C)$ is a finite set of transitions. A transition (s, s', ϕ, a, R) allows a jump from location s to s', communicating event $a \in \Sigma$ in the process, provided the constraint ϕ on clocks is met. Afterwards, all clocks in R are reset to zero.

An *open timed automaton* is a timed automaton in which all constraints $\phi \in \Phi_C$ on transitions are open, i.e., are generated by the grammar $\phi ::= x < k \mid x > k \mid \phi \wedge \phi \mid \phi \vee \phi$.

A *closed timed automaton* is a timed automaton in which all constraints on transitions are closed, i.e., are generated by the grammar $\phi ::= x \leqslant k \mid x \geqslant k \mid \phi \wedge \phi \mid \phi \vee \phi$.

Remark 2: One finds many variants of these definitions in the literature: allowing direct comparisons between clocks, e.g., $x - y > k$; allowing rational, rather than integral, bounds in clock constraints; including invariant clock constraints on locations; nondeterministically resetting clocks in some Φ_C-defined sets, rather than to zero. It is however not difficult to verify that all the results presented here extend straightforwardly to any combination of these variants.

2.2. Precise and Robust Semantics; the d-Topology

Assume a timed automaton $A = (\Sigma, S, S_0, S_f, C, E)$.

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 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 non-decreasing t_i's, where each state (s_i, t_i, ν_i) records the data immediately following the previous transition $\alpha_i = (s_{i-1}, s_i, \phi_i, a_i, R_i) \in E$. In addition,

1) $s_0 \in S_0$, $t_0 = 0$, and $\nu_0(x) = 0$ for all $x \in C$.
2) For all $0 \leqslant i \leqslant n-1$: $\nu_i + (t_{i+1} - t_i)$ satisfies ϕ_{i+1}, and $\nu_{i+1}(x) = 0$ for all $x \in R_{i+1}$.
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 $\mathbf{TT}$.

Remark 3: Note that our semantics is *weakly monotonic* with respect to time, i.e., it allows several events to share the same timestamp. By contrast, a *strongly monotonic* semantics would require that no two events happen at the same time. Outside of Section 6, most of the results presented here also hold under a strongly monotonic semantics. For more details on the relationship between the weakly and strongly monotonic semantics, we refer the reader to [22].

Given a run $e = (s_0, t_0, \nu_0) \xrightarrow{\alpha_1} (s_1, t_1, \nu_1) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (s_n, t_n, \nu_n)$ of the timed automaton A, there is 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.

Definition 4: The *precise traces* semantics of the timed automaton A is given by $[\![A]\!] \mathrel{\widehat{=}} \{\mathsf{tt}(e) \mid e \text{ is a run of } A\}$.

The set $[\![A]\!]$ represents the set of precise dense-time timed traces of A, i.e., those timed traces accepted by A under the standard semantics. From now on, we shall usually drop the qualifiers 'precise' and 'standard', except when needed to avoid confusion with the forthcoming robust semantics.

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.

Definition 5: The metric d on $\mathbf{TT}$ is given 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, let

$$d(u, u') \mathrel{\widehat{=}} \infty \quad \text{if } \mathsf{untime}(u) \neq \mathsf{untime}(u'), \text{ and}$$
$$d(u, u') \mathrel{\widehat{=}} \max\{|t_i - t'_i| : 1 \leqslant i \leqslant n\} \quad \text{otherwise.}$$

This metric is defined in [15], where it is also argued that any other 'reasonable' metric on timed traces yields the same topology. From now on, we shall usually omit reference to d and use the terms 'open', 'closed', etc. to mean d-open, d-closed, and so on. In particular, for $T \subseteq \mathbf{TT}$ a set of timed traces, we let $\overline{T}$ denote the d-closure of T, and T^{int} denote its d-interior.

Definition 6: The *robust traces* semantics of the timed automaton A is given by $\widetilde{[\![A]\!]} \mathrel{\widehat{=}} \overline{[\![A]\!]}^{\mathrm{int}}$.

(Note that $\overline{[\![A]\!]}^{\mathrm{int}}$ stands for $\left(\overline{[\![A]\!]}\right)^{\mathrm{int}}$.)

The set $\widetilde{[\![A]\!]}$ represents the set of robust dense-time timed traces of A, i.e., those timed traces 'accepted' by A under the robust semantics. As explained in the introduction, the presentation of this semantics differs slightly from that of [15] in that $\widetilde{[\![A]\!]}$ corresponds to the largest tube accepted by A under Gupta *et al.*'s definition. The two semantics are nonetheless equivalent, as each one can be recovered from the other.

We note that a timed trace u belongs to $\widetilde{[\![A]\!]}$ if and only if the set $[\![A]\!]$ of precise timed traces of A is dense in some open neighbourhood of u. In that case it can be shown that the set $\mathbf{TT} - [\![A]\!]$ of rejected traces of A is nowhere dense in that neighbourhood [15], so that this definition of robustness is sensible.

It is often argued that 'robustness' in general is desirable not only to derive realistic models of real-world systems, but also to ensure that any numerical package one may wish to use to probe a given model will not produce incorrect results due to inevitable small rounding errors. We refer the reader to [15], [23], [13], [6], [14], among others, for more leisurely discussions on the subject.

By convention, the default semantics we consider in the remainder of this paper is the precise semantics; all uses of the robust semantics are explicitly noted as such.

2.3. Integral Timed Traces

For $T \subseteq \mathbf{TT}$ 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.

This leads us to define $\mathbb{Z}[\![A]\!]$, the integral trace semantics of a timed automaton A, and $\mathbb{Z}\widetilde{[\![A]\!]}$, the robust integral trace semantics of A. These notions naturally figure prominently in digitization techniques.

3. Regions Constructions

We first review the notion of regions introduced in [1], and the associated definitions of the region and integral automata [2]. We then use the regions construction in a novel way, to obtain a discrete partition of the set of timed traces accepted by a timed automaton.

3.1. Regions

Let $n, K \in \mathbb{N}$ be fixed. We define an equivalence relation $\sim$ on $(\mathbb{R}^+)^n$ as follows: $(x_1, \ldots, x_n) \sim (x'_1, \ldots, x'_n)$ if

1) For all $1 \leqslant i \leqslant n$, either $\lfloor x_i \rfloor = \lfloor x'_i \rfloor$, or both x_i and x'_i are greater than K.
2) For all $1 \leqslant i, j \leqslant n$, with $x_i, x_j \leqslant K$, we have $\mathit{fract}(x_i) \leqslant \mathit{fract}(x_j) \Leftrightarrow \mathit{fract}(x'_i) \leqslant \mathit{fract}(x'_j)$.
3) For all $1 \leqslant i \leqslant n$ with $x_i \leqslant K$, $\mathit{fract}(x_i) = 0 \Leftrightarrow \mathit{fract}(x'_i) = 0$.

It is easy to see that $\sim$ partitions $(\mathbb{R}^+)^n$ into finitely many equivalence classes, termed *regions*.

Proposition 7: Every region r as defined above can be uniquely represented as a conjunction of integral linear constraints

$$\bigwedge_{i=1}^{n} \rho_i \wedge \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} \sigma_{i,j}$$

where each ρ_i is of the form $x_i = k$ (for $0 \leqslant k \leqslant K$), or of the form $k < x_i < k+1$ (for $0 \leqslant k < K$), or of the form $K < x_i$; and each $\sigma_{i,j}$ is of the form $x_j - x_i = k$ (for $-K \leqslant k \leqslant K$), or of the form $k < x_j - x_i < k+1$ (for $-K \leqslant k < K$), or is simply *true*. In addition, we specify that $\sigma_{i,j}$ be *true* exactly when at least one of x_i, x_j exceeds K (as per the constraints ρ_i and ρ_j). Note that the $\sigma_{i,j}$'s can equivalently be written (given the ρ_i's) as constraints of the form $\mathit{fract}(x_i) \sharp \mathit{fract}(x_j)$, with $\sharp \in \{=, <, >\}$, or *true*.

Conversely, every such satisfiable conjunction of linear constraints represents a unique region.

Proof: This is a straightforward (if somewhat tedious) exercise. ■

Proposition 8: Consider the usual Euclidean topology on $(\mathbb{R}^+)^n$.

1) The open regions are exactly those whose associated conjunction of constraints comprise only either (strict) inequalities or *true*.
2) The closed regions are exactly those whose associated conjunction of constraints comprise only equalities; they consist in all singletons containing a tuple of integers.
3) Let r be an open region. For any region r', either $r' \subseteq \overline{r}$ or $r' \cap \overline{r} = \emptyset$.

Proof: Follows directly from Proposition 7. ■

3.2. Region and Integral Automata

Let $A = (\Sigma, S, S_0, S_f, C, E)$ be a timed automaton. We define the following two untimed automata: the *region automaton* A^{reg} accepts the same untimed language as A, whereas the *integral automaton* $A^{\checkmark}$ accepts (essentially) the same integral language as A. We begin by describing the region automaton construction.

Let n be the number of clocks in A, and let K be the largest integer constant appearing in any of the clock constraints associated with the transitions of A. Consider the regions construction over $(\mathbb{R}^+)^n$ described earlier. Since every clock interpretation for A can be uniquely identified with a point of $(\mathbb{R}^+)^n$, we term the equivalence classes *clock regions* of A.

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 $\nu + t \in r'$. In other words, $\preccurlyeq$ represents the passage of time.

A^{reg} is an untimed automaton with ε-transitions (silent transitions). As is well-known, such untimed automata can easily be transformed into ones without ε-transitions [19]. The alphabet Σ of A^{reg} is the same as that of A. The states of A^{reg} 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 A^{reg} consist of all states of the form $(s_0, \mathbf{0})$, where $s_0 \in S_0$ and $\mathbf{0}$ is the clock interpretation which maps every clock to 0. The accepting states of A^{reg} consist of all states of the form (s_f, r), where $s_f \in S_f$. For any pair of clock regions $r \preccurlyeq r'$ and location s, A^{reg} has an ε-transition $(s, r) \xrightarrow{\varepsilon} (s, r')$. Moreover, for any states (s, r) and (s', r'), A^{reg} has an a-transition $(s, r) \xrightarrow{a} (s', r')$ whenever A has a transition (s, s', ϕ, a, R) 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 $\nu \in r'$ satisfy $\nu(x) = 0$ for every $x \in R$.

We write $L(A^{\mathrm{reg}})$ to denote the (untimed) language accepted by A^{reg}.

We now have:

Theorem 9—[2]: For any timed automaton A, $L(A^{\mathrm{reg}}) = \mathsf{untime}([\![A]\!])$.

We refer the reader to [2] 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 9, the emptiness problem for timed automata is decidable.

We now give the construction of the integral automaton $A^{\checkmark}$. First 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 10: The set of integral timed traces of a timed automaton is (essentially) regular. In other words, for any timed automaton A, one can construct an untimed automaton $A^{\checkmark}$ such that $L(A^{\checkmark}) = \mathbb{Z}[\![A]\!]^{\checkmark}$.

Proof: The untimed automaton $A^{\checkmark}$ can be obtained by a straightforward modification of the region automaton A^{reg}. The only clock regions that need be considered are those all of whose projections are either integral singletons or are unbounded. All ε-transitions are discarded. The Σ-transitions of $A^{\checkmark}$ are inherited directly from A^{reg}. Lastly, postulate a transition $(s, r) \xrightarrow{\checkmark} (s, r')$ of $A^{\checkmark}$ if, for all clock interpretations $\nu \in r$, $\nu + 1 \in r'$. It is easily checked that $A^{\checkmark}$ accepts precisely $\mathbb{Z}[\![A]\!]^{\checkmark}$, as required. ■

Remark 11: The $\checkmark$-representation of integral timed traces is necessary to ensure regularity. For instance, the set $\{\langle (1, a), (2, a), \ldots, (n, a) \rangle \mid n \in \mathbb{N}\}$ is not regular, whereas its $\checkmark$-counterpart $\{\langle \checkmark a \rangle^n \mid n \in \mathbb{N}\}$ clearly is.

Proposition 10 immediately yields the decidability of the integral emptiness and integral universality problems for timed automata; in other words, it is decidable whether $\mathbb{Z}[\![A]\!] = \emptyset$ and whether $\mathbb{Z}[\![A]\!] = \mathbb{Z}\mathbf{TT}$.

3.3. Trace Regions

We now make use of the regions construction in a very different way, to discretely partition the set of timed traces of a timed automaton, and to endow these partitions with a useful geometric and topological structure.

Let $s = a_1 a_2 \ldots a_n$ be some untimed trace of length n. We can uniquely identify timed traces having untimed projection s with weakly monotonic points of $(\mathbb{R}^+)^n$, i.e., those points of $(\mathbb{R}^+)^n$ lying in the hyper-plane $T_n = \{x_1, \ldots, x_n \mid x_1 \leqslant x_2 \leqslant \ldots \leqslant x_n\}$.

Let A be a timed automaton and let K be the largest integer constant appearing in any of the clock constraints associated with the transitions of A. We consider the regions construction over T_n, and call resulting equivalence classes *trace regions*. Accordingly, we also view the equivalence relation $\sim$ as applying to timed traces. Note that for two timed traces u and u', the equivalence $u \sim u'$ implies that $\mathsf{untime}(u) = \mathsf{untime}(u')$. We also observe that the d-topology on traces agrees with the Euclidean topology on T_n, and we shall therefore not distinguish between the two.

Proposition 12: Let A be a timed automaton. For any pair of timed traces $u \sim u'$, we have $u \in [\![A]\!]$ iff $u' \in [\![A]\!]$.

Proof: Assume that $u = \langle (t_1, a_1), \ldots, (t_n, a_n) \rangle \in [\![A]\!]$. Then u originates from some run $e = (s_0, t_0, \nu_0) \xrightarrow{\alpha_1} (s_1, t_1, \nu_1) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (s_n, t_n, \nu_n)$, with $\alpha_j = (s_{j-1}, s_j, \phi_j, a_j, R_j)$. Observe that, for any clock x and index j, $\nu_j(x) = t_j - t_i$, where $i \leqslant j$ is the index of the last transition which reset clock x, or is 0 if x was never reset.

Write $u = \langle (t'_1, a_1), \ldots, (t'_n, a_n) \rangle$. Since $u \sim u'$, one readily verifies that, for any $1 \leqslant i \leqslant j \leqslant n$ and $k \in \mathbb{N}$, whenever $t_j - t_i \sharp k$ we must have $t'_j - t'_i \sharp k$, for any $\sharp \in \{<, \leqslant, >, \geqslant\}$. It follows immediately that A accepts u' along the same path as u. ■

4. Automata Closure and Interior

Closed and open timed automata (cf. Definition 1) possess many desirable properties, some of which are very useful to our endeavour and are presented here. We then address the problem of constructing the closure and interior automata of a given timed automaton. We are subsequently able to establish several facts concerning the robust semantics of timed automata, most notably that its expressive power is incomparable to that of the standard semantics. On the positive side, we show that timed automata have recursive robust languages.

Proposition 13: Let A be a timed automaton.

1) $\widetilde{[\![A]\!]}$ is open.
2) If A is an open timed automaton, then $[\![A]\!]$ is open.
3) If A is a closed timed automaton, then $[\![A]\!]$ is closed.

Proof: The first claim follows from the definition of the robust trace semantics.

The second claim appears in [15], 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 third claim. Let A be a closed timed automaton, and consider an arbitrary timed trace u. Let $\langle u_i \rangle_{i \geqslant 1}$ be a sequence of timed traces in $[\![A]\!]$ 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 sequence of e_i's therefore essentially lies in a compact subset of $\mathbb{R}^l$ (for some finite l), where we identify l-tuples with sequences of clock interpretations. Consequently, these e_i's must have an accumulation point e. This 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 $u \in [\![A]\!]$ as required. ■

Proposition 14: Let A be a timed automaton. Then one can construct a closed timed automaton $\overline{A}$ such that $[\![\overline{A}]\!] = \overline{[\![A]\!]}$.

This result is stated in [15], but unfortunately the construction and proof given there are incorrect. It is claimed in that paper that $\overline{A}$ can be obtained by replacing every strict inequality appearing in A's clock constraints by its non-strict version. However, the following automata demonstrate that this construction can lead to extra unwanted behaviours:

The left-hand automaton cannot accept any timed trace, and its language closure is therefore empty as well. On the other hand, the right-hand automaton can accept the timed trace $\langle (1, a) \rangle$.

We must therefore ensure that any new behaviour of $\overline{A}$ introduced through the replacement of strict inequalities by non-strict ones can always be 'shadowed' by a neighbouring behaviour of A. For this purpose, a regions-based construction is introduced. We now give the proof.

Proof: Let $A = (\Sigma, S, S_0, S_f, C, E)$ be a timed automaton. Without loss of generality, assume that all clock constraints of A are disjunction-free. This can be achieved by first writing the clock constraints in disjunctive normal form, and then introducing a new distinct transition for each disjunct. Note that these disjunction-free constraints are conjunctions of basic formulas of the form $x_i > k$, $x_i \geqslant k$, or $x_i = k$. They therefore represent convex hyper-rectangles (products of intervals).

We construct an equivalent timed automaton $A' = (\Sigma, S', S'_0, S'_f, C, E')$ as follows. A' is essentially a timed version of the region automaton A^{reg}: $S' = S \times \mathit{CLREG}$, $S'_0 = S_0 \times \mathit{CLREG}$, and $S'_f = S_f \times \mathit{CLREG}$, where CLREG denotes the set of clock regions of A. Let $\alpha = (s, s', \phi, a, R) \in E$ be a transition of A. For every pair of clock regions r, r', include the transition $((s, r), (s', r'), \phi, a, R)$ in E', provided there exist a clock region $r'' \succcurlyeq r$ such that r'' satisfies ϕ, r'' and r' agree on all clocks not belonging to R, and r' evaluates all clocks in R to zero.

It is easily checked that $[\![A']\!] = [\![A]\!]$. Now define $\overline{A}$ to be the same as A' but with every strict inequality appearing in a clock constraint of A' replaced by its corresponding non-strict version. We now show that $[\![A]\!]$ is a dense subset of $[\![\overline{A}]\!]$.

Let $e = ((s_0,r_0),t_0,\nu_0) \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n} ((s_n,r_n),t_n,\nu_n)$ be a run of $\overline{A}$ yielding the timed trace $u = \langle (t_1,a_1),\dots,(t_n,a_n)\rangle \in [\![\overline{A}]\!]$. We have $\alpha_i = ((s_{i-1},r_{i-1}),(s_i,r_i),\phi_i,a_i,R_i)$, for each $1 \leqslant i \leqslant n$.

By construction, the projection $(s_0,r_0) \xrightarrow{a_1} \dots \xrightarrow{a_n} (s_n,r_n)$ of e is a run of the region automaton A^{reg} (where we are ignoring ε-transitions). It follows that there must be a run $(s_0,t'_0,\nu'_0) \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n} (s_n,t'_n,\nu'_n)$ of A with each $\nu'_i \in r_i$. This run yields the timed trace $u' = \langle (a_1,t'_1),\dots,(a_n,t'_n)\rangle \in [\![A]\!]$.

Let $0 < r < 1$ be fixed. For $0 \leqslant i \leqslant n$, let $\nu''_i = r\nu_i + (1-r)\nu'_i$ and $t''_i = rt_i + (1-r)t'_i$. Consider a clock $x \in C$. As in the proof of Proposition 12, we have, for any $0 \leqslant j \leqslant n$, that $\nu_j(x) = t_j - t_i$, where $i \leqslant j$ is the index of the last transition which reset clock x in e, or is 0 if x was never reset. Likewise, $\nu'_j(x) = t'_j - t'_i$, and $\nu''_j(x) = t''_j - t''_i$.

Since e is a valid run of $\overline{A}$, for any $0 \leqslant j \leqslant n-1$, we have by definition that $\nu_j(x) + (t_{j+1} - t_j)$ satisfies $\overline{\phi_{j+1}(x)}$. Since ϕ_{j+1} is a hyper-rectangle, $\phi_{j+1}(x)$ is some non-empty interval I^x_{j+1}, and $\overline{\phi_{j+1}(x)}$ is its closure $\overline{I^x_{j+1}}$. Substituting, we have $(t_j - t_i) + (t_{j+1} - t_j) \in \overline{I^x_{j+1}}$, and likewise $(t'_j - t'_i) + (t'_{j+1} - t'_j) \in I^x_{j+1}$. It immediately follows that $(t''_j - t''_i) + (t''_{j+1} - t''_j) \in I^x_{j+1}$, in other words that $\nu''_j(x) + (t''_{j+1} - t''_j)$ satisfies $\phi_{j+1}(x)$. We therefore conclude that $(s_0,t''_0,\nu''_0) \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n} (s_n,t''_n,\nu''_n)$ is a valid run of A yielding the timed trace $u'' = \langle (a_1,t''_1),\dots,(a_n,t''_n)\rangle = ru + (1-r)u' \in [\![A]\!]$. Letting r tend to 1 shows that there are timed traces in $[\![A]\!]$ that are arbitrarily close to those in $[\![\overline{A}]\!]$.

Hence we have shown that $[\![A]\!]$ is a dense subset of $[\![\overline{A}]\!]$. Since $[\![\overline{A}]\!]$ is closed by the previous proposition, $\overline{[\![A]\!]} = [\![\overline{A}]\!]$ as required. ■

Corollary 15: For any timed automaton A, there exists a closed timed automaton, namely $\overline{A}$, having the same robust trace semantics: $\widetilde{[\![A]\!]} = \widetilde{[\![\overline{A}]\!]}$.

Proof: This follows directly from the fact that a closed set is equal to its closure. ■

Corollary 15 implies that, when it comes to robust trace semantics, we may restrict our attention to closed timed automata. The following proposition indicates that we may equally well focus on open timed automata instead.

Proposition 16: Let A be a timed automaton. Then one can construct an open timed automaton A^{op} such that $\widetilde{[\![A]\!]} = \widetilde{[\![A^{\mathrm{op}}]\!]}$.

Proof: This proof is a simplified version of that originally given in [15], the simplifications resulting from our requirement that clocks can only be reset to zero.

One obtains A^{op} by replacing every non-strict inequality in A by its strict counterpart.[2] Since we clearly have $[\![A^{\mathrm{op}}]\!] \subseteq \widetilde{[\![A]\!]}$, it suffices to show that $[\![A^{\mathrm{op}}]\!]$ is dense in $\widetilde{[\![A]\!]}$. To this end, consider a small neighbourhood around some trace $u \in \widetilde{[\![A]\!]}$. In this neighbourhood one can find some trace $u' \in [\![A]\!]$ which lies in an open trace region. It is then easily seen that, in any run of A producing u', no clock can have an integral value immediately prior to any transition. Consequently, all clock constraints on such runs are satisfied strictly, and hence $u' \in [\![A^{\mathrm{op}}]\!]$ as required. ■

As pointed out in [15], Proposition 16 immediately entails the decidability of the *robust emptiness problem*, i.e., whether the set of robust timed traces of a timed automaton is empty or not. More generally, we have:

Proposition 17: For any timed automaton A, the set $\mathsf{untime}(\widetilde{[\![A]\!]})$ is regular.

Proof: By the previous proposition, $\mathsf{untime}(\widetilde{[\![A]\!]}) = \mathsf{untime}(\widetilde{[\![A^{\mathrm{op}}]\!]})$. Thanks to Theorem 9, it thus remains to show that $\mathsf{untime}(\widetilde{[\![A^{\mathrm{op}}]\!]}) = \mathsf{untime}([\![A^{\mathrm{op}}]\!])$.

Clearly, $[\![A^{\mathrm{op}}]\!] \subseteq \widetilde{[\![A^{\mathrm{op}}]\!]}$ since A^{op} is open. On the other hand, let $u \in \widetilde{[\![A^{\mathrm{op}}]\!]}$. We must show there exists $u' \in [\![A^{\mathrm{op}}]\!]$ such that $\mathsf{untime}(u') = \mathsf{untime}(u)$. Since $\widetilde{[\![A^{\mathrm{op}}]\!]}$ is open, there exists an open neighbourhood $U \subseteq \widetilde{[\![A^{\mathrm{op}}]\!]}$ with $u \in U$. Moreover, we can also clearly require that all timed traces in U have the same untimed projection as u. Now since $U \subseteq \widetilde{[\![A^{\mathrm{op}}]\!]}$, $[\![A^{\mathrm{op}}]\!]$ must be dense in U, and in particular there must exist some $u' \in U \cap [\![A^{\mathrm{op}}]\!]$, as required. ■

A natural question to ask (particularly in view of Propositions 14 and 16), is whether, given a timed automaton A, one can construct an *interior automaton* for A, i.e., an (open) timed automaton B such that $[\![B]\!] = [\![A]\!]^{\mathrm{int}}$. Unfortunately, the following theorem shows that this is impossible.

Theorem 18: There exists a timed automaton A such that, for any timed automaton B, $\widetilde{[\![A]\!]} \neq [\![B]\!]$. (Moreover, A may be chosen to be closed, open, or mixed.)

This result states that the expressive powers of timed automata under the standard and robust semantics are incomparable.[3] This is rather surprising since the robust semantics can be viewed as a restriction of the standard semantics in which equality testing has been semantically removed [15].

We defer the proof of Theorem 18 until the next section.

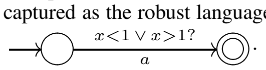
Although Theorem 18 implies that constructing full interior automata is in general impossible, one can still obtain bounded approximations of interior automata, as we demonstrate next.

We first need an auxiliary definition. For $T \subseteq \mathbf{TT}$ a set of timed traces and $n \in \mathbb{N}$ a non-negative integer, we define $T \upharpoonright n$ to be the set of all timed traces in T of length (i.e., number of events) no greater than n.

Proposition 19: Let A be a timed automaton, and let $n \in \mathbb{N}$ be given. Then one can construct an open timed automaton A^{int}_n such that $[\![A^{\mathrm{int}}_n]\!] = [\![A]\!]^{\mathrm{int}} \upharpoonright n$.

Proof: Consider an untimed trace $s = a_1 a_2 \dots a_m$ of length $m \leqslant n$. By Proposition 12, the set $T_s \subseteq [\![A]\!]$ of timed traces of A that have untimed projection s is a finite union of

[2] In [15], the notation A^{int} is used instead of A^{op}; we prefer the latter since one can have the *strict* inclusion $[\![A^{\mathrm{op}}]\!] \subsetneq [\![A]\!]^{\mathrm{int}}$.

[3] It is obvious that there are (open, closed, mixed) timed automata whose precise languages cannot be captured as the robust languages of any timed automata; one such example is 

.

trace regions, which can be effectively enumerated. The interior $U_s \subseteq [\![A]\!]^{\mathrm{int}}$ of T_s is therefore clearly an open union of trace regions, the list of which is again effectively enumerable.

For every region $r \subseteq U_s$, construct an open convex set O_r with $r \subseteq O_r \subseteq U_s$, as follows. First write r as a conjunction of linear constraints, as per Proposition 7. Next, replace every linear equality constraint of the form $\chi = k$ by the pair of strict linear inequalities $k-1 < \chi < k+1$ (and in case $k = K$, dispense with the constraint $\chi < k+1$). In other words, O_r consists of all the regions whose closures contain r. Proposition 8 (3) implies that $O_r \subseteq U_s$.

For each region $r \subseteq U_s$, we build an open timed automaton $A_{s,r}$ that accepts exactly those timed traces having untimed projection s that lie in O_r. $A_{s,r}$ comes equipped with m clocks and resets a distinct clock every time it performs an event. The set of strict inequalities which define O_r easily yields a corresponding set of clock constraints for $A_{s,r}$.

The required automaton A_n^{int} is then simply the union of all the automata $A_{s,r}$, for every untimed trace s of length at most n and region $r \in U_s$. ∎

We note that, while it is straightforward to decide whether a particular timed trace is *precisely* accepted by a given timed automaton, the situation might seem a priori very different when one considers *robust* acceptance. Indeed, whether a timed automaton A robustly accepts some timed trace u depends on whether A precisely accepts infinitely many traces neighbouring u. Fortunately, the question remains decidable:

Corollary 20: The set of robust timed traces of a timed automaton is recursive. In other words, for any timed automaton A, there is an algorithm which, given a timed trace $u \in \mathbf{TT}$, decides whether $u \in \widetilde{[\![A]\!]}$.

Proof: Follows directly from the chain

$$\widetilde{[\![A]\!]} \upharpoonright |u| = \overline{[\![A]\!]}^{\mathrm{int}} \upharpoonright |u| = [\![\overline{A}]\!]^{\mathrm{int}} \upharpoonright |u| = [\![\overline{A}_{|u|}^{\mathrm{int}}]\!] ,$$

where $|u|$ denotes the length of u. ∎

5. Universality

Universality is a fundamental problem in the language-theoretic study of computational models [19]. Indeed, one of the original motivations for introducing the robust semantics for timed automata was the hope that it may have a decidable universality problem. This matter was finally settled negatively in [18], but the analysis underlying this result relied on a strongly monotonic model of time (no two events are allowed to share the same timestamp). Recent work on the universality problem within the closely related formalism of open timed automata showed that, while the problem remains undecidable for open timed automata over strongly monotonic time, it becomes decidable over weakly monotonic time [22]. In view of the tight relationship between open and robust timed automata (cf. Proposition 19 and Theorem 34), it seemed plausible that the robust universality problem for timed automata might be decidable over weakly monotonic time. Unfortunately, as we demonstrate shortly, this is not the case. We also show that the robust integral languages of timed automata are in general not regular, and that they too give rise to an undecidable (integral) universality problem.

We begin by introducing a few preliminaries. 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 non-zero) one of the counters, and subsequently jumps nondeterministically to one of two possible next instructions, 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.

Lemma 21—[22]: Let M be a two-counter machine. The set of halting computations of M can be encoded as an open set of timed traces $L(M) \subseteq \mathbf{TT}$ over alphabet $\Sigma = \{b_0, \ldots, b_k, c, d\}$. Moreover, one can construct a closed timed automaton A such that $[\![A]\!] = \mathbf{TT} - L(M)$.

We refer the reader to [22] for the proof, as well as for greater details concerning two-counter machines and the various encodings mentioned here. We note that the relevant construction invoked in [22] builds on an idea originally introduced in [18].

Theorem 22: The robust universality problem for timed automata is undecidable. In other words, given a timed automaton A, it is undecidable whether $\widetilde{[\![A]\!]} = \mathbf{TT}$.

Proof: Let A be a closed timed automaton. We have $\widetilde{[\![A]\!]} = \mathbf{TT}$ iff $[\![A]\!]^{\mathrm{int}} = \mathbf{TT}$ iff $[\![A]\!] = \mathbf{TT}$. Thus the robust universality problem reduces to the universality problem for closed timed automata, which in turn (by Lemma 21) reduces to the emptiness problem for two-counter machines. The latter is well-known to be undecidable [19]. ∎

Digitization techniques (cf. Section 6) rely crucially on decision algorithms for integral sublanguages. Unfortunately, the robust universality problem is no more tractable in the discrete world:

Theorem 23: The robust integral universality problem for timed automata is undecidable. In other words, given a timed automaton A, it is undecidable whether $\mathbb{Z}\widetilde{[\![A]\!]} = \mathbb{Z}\mathbf{TT}$.

Proof: Let M be a two-counter machine, and let $L(M)$ and A be as in Lemma 21, so that $[\![A]\!] = \mathbf{TT} - L(M)$. It is plain that if $L(M) = \emptyset$, then $\mathbb{Z}\widetilde{[\![A]\!]} = \mathbb{Z}\mathbf{TT}$.

Suppose, on the other hand, that $L(M) \neq \emptyset$. Since $L(M)$ is an open set of timed traces, it must meet some open trace region r (where the set of trace regions is determined by A as per Section 3). By Proposition 12, $r \cap [\![A]\!] = \emptyset$. Let $v \in \overline{r}$ be an integral timed trace belonging to the closure of r (easily obtained thanks to Proposition 7). It is immediate that $v \notin \widetilde{[\![A]\!]}$, and therefore that $\mathbb{Z}\widetilde{[\![A]\!]} \neq \mathbb{Z}\mathbf{TT}$.

Since the question whether $\mathbb{Z}\widetilde{[\![A]\!]} = \mathbb{Z}\mathbf{TT}$ reduces to the emptiness problem two-counter machines, it must be undecidable. ■

Remark 24: We do not know whether the robust integral emptiness problem for timed automata (i.e., whether $\mathbb{Z}\widetilde{[\![A]\!]} = \emptyset$) is decidable or not.

The following result, to be contrasted with Proposition 10, highlights the fact that the robust semantics for timed automata is markedly less tractable than its standard counterpart.

Theorem 25: There exists a timed automaton A such that $\mathbb{Z}\widetilde{[\![A]\!]}^{\checkmark}$ is not regular.

Proof: Let M be a two-counter machine which accepts precisely those computations in which instruction b_1 is executed exactly as often as instruction b_2. As per Lemma 21, let A be a timed automaton with $[\![A]\!] = \mathbf{TT} - L(M)$. The argument invoked in the proof of Theorem 23 shows that, for $v \in \mathbb{Z}\mathbf{TT}$ an integral timed trace, if $v \notin \mathbb{Z}\widetilde{[\![A]\!]}$ then v contains as many b_1's as b_2's, and conversely for any non-negative integer n, there exists some integral timed trace v containing n b_1's and n b_2's and such that $v \notin \mathbb{Z}\widetilde{[\![A]\!]}$. In other words, the set $B = (\mathbb{Z}\mathbf{TT}^{\checkmark} - \mathbb{Z}\widetilde{[\![A]\!]}^{\checkmark}) \restriction \{b_1, b_2\}$ consists of all strings over $\{b_1, b_2\}$ that contain as many b_1's as b_2's.

A straightforward application of the pumping lemma [19] shows that B is not regular. It follows that neither is $\mathbb{Z}\widetilde{[\![A]\!]}^{\checkmark}$, since regular sets are closed under complementation and hiding [19]. ■

We note that Theorem 23 does *not* in itself imply Theorem 25; it merely shows that a purported construction of an untimed automaton accepting the robust integral language of a timed automaton A could not in general be effective.

We are now able to give the proof of Theorem 18, to the effect that there are timed automata whose robust languages cannot be captured as the precise languages of any timed automata.

Proof: For A and B two timed automata, we note that $\widetilde{[\![A]\!]} = [\![B]\!]$ implies that $\mathbb{Z}\widetilde{[\![A]\!]}^{\checkmark} = \mathbb{Z}[\![B]\!]^{\checkmark}$. Since, by Proposition 10, the set $\mathbb{Z}[\![B]\!]^{\checkmark}$ is always regular, the previous proposition shows that there exists some timed automaton A for which there can be no timed automaton B with $\widetilde{[\![A]\!]} = [\![B]\!]$. ■

Again, we note that this result does not follow from Theorem 23 alone.

6. Digitization and Decidability

As explained in the introduction, digitization techniques [17] are a very efficient and successful tool to analyze properties of timed automata, including key questions such as the emptiness, universality, and language inclusion problems. In this section, we study the decidability of the prerequisites to the application of digitization techniques, as well as their applicability to timed automata under both the standard and robust semantics.

We begin by introducing some preliminaries. 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 $[t]_\varepsilon \mathrel{\widehat{=}} \lceil t \rceil$. The $[\cdot]_\varepsilon$ operator therefore shifts the value of a real number t to the preceding or following integer, depending on whether the fractional part of t is less than ε or not.

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 26: Let $T \subseteq \mathbf{TT}$ 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 $u \in \mathbf{TT}$ is such that $[u]_\varepsilon \in T$ for all $0 \leqslant \varepsilon \leqslant 1$, then $u \in T$.

The main digitization result is as follows:

Theorem 27—[17]: 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 $\mathbb{Z}T \subseteq \mathbb{Z}T'$.

Proof: The left-to-right implication is trivial. For the other 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. ■

Corollary 28: 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 $[\![A]\!] \subseteq [\![B]\!]$ is decidable.

Proof: Follows directly from Theorem 27 and the fact that the sets $\mathbb{Z}[\![A]\!]^{\checkmark}$ and $\mathbb{Z}[\![B]\!]^{\checkmark}$ are regular (Proposition 10). ■

Note that the emptiness and universality problems are special cases of the language inclusion problem. We also remark that, in the realm of formal verification, the automaton A would represent an implementation, whereas the automaton B would stand for a specification.

Theorem 27 and Corollary 28 highlight the importance of (i) being able to determine whether a timed automaton is closed under digitization and/or closed under inverse digitization, and (ii) being able to decide integral language inclusion questions. We consider these matters in the remainder of this section.

We first record the following useful result:

Lemma 29: Let $s \in \mathbf{TT}$ be a timed trace of length n, and let r be its corresponding trace region (where we assume an arbitrarily large value of K). The set $\{[s]_\varepsilon \mid 0 \leqslant \varepsilon \leqslant 1\}$ of valid digitizations of s is in one-to-one correspondence with the set of $\overline{r} \cap (\mathbb{R}^+)^n$ of integral singletons (closed regions) bordering r.

Proof: Follows straightforwardly from Proposition 7. ■

Proposition 30: Let A be a timed automaton.

1) If A is an open timed automaton, then $[\![A]\!]$ is closed under inverse digitization.
2) If A is a closed timed automaton, then $[\![A]\!]$ is closed under digitization.

Proof: (These two results were already implicitly present in [17].)

Let A be an open timed automaton, and let $u \in \mathbf{TT}$ be an arbitrary timed trace. We establish the stronger result that, if *any* ε-digitization of u belongs to $\llbracket A \rrbracket$, then so does u. Indeed, let $v = [u]_\varepsilon \in \llbracket A \rrbracket$. If r is the trace region corresponding to u, then by Lemma 29 $v \in \overline{r}$. Since $v \in \llbracket A \rrbracket$ and $\llbracket A \rrbracket$ is open (thanks to Proposition 13), $r \cap \llbracket A \rrbracket \neq \emptyset$. But then Proposition 12 immediately yields that $r \subseteq \llbracket A \rrbracket$, whence $u \in \llbracket A \rrbracket$ as required.

Let us now consider the case in which A is a closed timed automaton. Since $\llbracket A \rrbracket$ is closed (Proposition 13), Lemma 29 immediately entails that $\llbracket A \rrbracket$ is closed under digitization. ■

Proposition 30 holds the key to the practical success of digitization techniques: since arbitrary timed automata can be infinitesimally over-approximated by closed timed automata (Proposition 14), and since the latter are closed under digitization, the requirement of Corollary 28 that implementations be closed under digitization is no hardship. On the other hand, it turns out that the most common specifications on timed systems can be expressed as open timed automata, or at least as timed automata that are closed under inverse digitization [17], [7], [21].

Note, unfortunately, that the converse of Proposition 30 does not hold, hence the wish for a decision procedure for closure under digitization and inverse digitization.

Theorem 31: The problem of closure under digitization is decidable for timed automata. In other words, there is an algorithm which, given a timed automaton A, decides whether $\llbracket A \rrbracket$ is closed under digitization.

Proof: Let A be a timed automaton, and let $\overline{A}$ be a closed automaton with $\llbracket \overline{A} \rrbracket = \overline{\llbracket A \rrbracket}$ as per Proposition 14. Let $[\llbracket A \rrbracket] \mathrel{\widehat{=}} \bigcup \{ [\llbracket A \rrbracket]_\varepsilon \mid 0 \leqslant \varepsilon \leqslant 1 \}$ be the set of digitized timed traces of A. By definition, $\llbracket A \rrbracket$ is closed under digitization if and only if $[\llbracket A \rrbracket] \subseteq \llbracket A \rrbracket$.

By Lemma 29, $\mathbb{Z}\llbracket \overline{A} \rrbracket = [\llbracket A \rrbracket]$. We have therefore reduced the question of whether $\llbracket A \rrbracket$ is closed under digitization to whether $\llbracket \overline{A} \rrbracket$ and $\llbracket A \rrbracket$ have the same integral timed traces. But this is decidable by Proposition 10, which completes the proof. ■

Unfortunately, when it comes to the robust trace semantics, closure under digitization is no longer decidable:

Theorem 32: The problem of closure under digitization is undecidable for timed automata under the robust trace semantics. In other words, given a timed automaton A, it is undecidable whether $\widetilde{\llbracket A \rrbracket}$ is closed under digitization.

Proof: Suppose that, given a timed automaton A, one could decide whether $\widetilde{\llbracket A \rrbracket}$ were closed under digitization. Then, as we now demonstrate, one could decide whether $\widetilde{\llbracket A \rrbracket}$ were universal, contradicting Theorem 22.

First note that if $\widetilde{\llbracket A \rrbracket}$ is not closed under digitization, then $\widetilde{\llbracket A \rrbracket} \neq \mathbf{TT}$. Assume therefore that $\widetilde{\llbracket A \rrbracket}$ is closed under digitization. Let s be some untimed trace of length n, and consider the (possibly empty) set $\widetilde{\llbracket A \rrbracket}_s \subseteq \widetilde{\llbracket A \rrbracket}$ of robust timed traces of A having untimed projection s. We view $\widetilde{\llbracket A \rrbracket}_s$ as a subset of $(\mathbb{R}^+)^n$. Thanks to Propositions 19 and 12, $\widetilde{\llbracket A \rrbracket}_s$ is a union of trace regions.

$\widetilde{\llbracket A \rrbracket}_s$ is clearly open. To see that it is also closed, consider a region $r \subseteq \widetilde{\llbracket A \rrbracket}_s$, and any region $r' \in \overline{r}$. By Propositions 7 and 8, $\overline{r}$ and $\overline{r'}$ must share at least one integral singleton v, which belongs to $\widetilde{\llbracket A \rrbracket}_s$ since $\widetilde{\llbracket A \rrbracket}$ is assumed closed under digitization. But $\widetilde{\llbracket A \rrbracket}_s$ is open, and therefore it follows that $r' \subseteq \widetilde{\llbracket A \rrbracket}_s$. The closure of $\widetilde{\llbracket A \rrbracket}_s$ is now immediate.

Thus $\widetilde{\llbracket A \rrbracket}_s$ is both open and closed. As a subset of a connected set (the set of all timed traces having untimed projection s, which is convex hence path-connected), $\widetilde{\llbracket A \rrbracket}_s$ is either empty or contains every possible timed trace having untimed projection s. Thus whether $\widetilde{\llbracket A \rrbracket} = \mathbf{TT}$ or not reduces to asking whether $\mathsf{untime}(\widetilde{\llbracket A \rrbracket}) = \Sigma^*$. Since the latter is decidable (Proposition 17), we have reached a contradiction. ■

We now investigate closure under digitization. This time, we find that the decidability results are reversed!

Theorem 33: The problem of closure under inverse digitization is undecidable for closed timed automata. In other words, given a closed timed automaton A, it is undecidable whether $\llbracket A \rrbracket$ is closed under inverse digitization. (Naturally, this implies that the same is true of mixed timed automata.)

Proof: We reduce the problem of deciding closure under inverse digitization to the universality problem for closed timed automata; the result then follows from Lemma 21. Suppose indeed that the former were decidable. Then, given a timed automaton A, determine first whether $\llbracket A \rrbracket$ is closed under inverse digitization. If it isn't, then clearly $\llbracket A \rrbracket \neq \mathbf{TT}$. On the other hand, if $\llbracket A \rrbracket$ is closed under inverse digitization, then, by Corollary 28, the question $\mathbf{TT} \subseteq \llbracket A \rrbracket$ is decidable. Since the latter is obviously equivalent to $\mathbf{TT} = \llbracket A \rrbracket$, a contradiction has been reached. ■

Next, we have:

Theorem 34: Timed automata are always closed under inverse digitization under the robust trace semantics. In other words, for A a timed automaton, $\widetilde{\llbracket A \rrbracket}$ is closed under inverse digitization.

Proof: Follows immediately from Propositions 19 and 30. ■

This is a somewhat surprising result, since the robust universality problem is undecidable. The reason, of course, is that the reduction from dense-time to integral-time is of no help when it comes to the robust semantics. It therefore also follows that, although the robust languages of timed automata are closed under inverse digitization, they are in general not suitable to express specifications, since their integral restrictions are not well-behaved.

7. Summary

We summarize in tabular form the main results discussed in this paper.

The emptiness and robust emptiness problems are decidable:

A	$\llbracket A \rrbracket = \emptyset$?	$\widetilde{\llbracket A \rrbracket} = \emptyset$?
Mixed	Decidable	Decidable

The integral emptiness problem is decidable, but whether the robust emptiness problem is decidable is still open:

A	$\mathbb{Z}[\![A]\!] = \emptyset\,?$	$\mathbb{Z}\widetilde{[\![A]\!]} = \emptyset\,?$
Mixed	Decidable	?

The universality problem is only decidable for open timed automata under the standard semantics:

A	$[\![A]\!] = \mathbf{TT}\,?$	$\widetilde{[\![A]\!]} = \mathbf{TT}\,?$
Mixed	Undecidable	Undecidable
Open	Decidable	Undecidable
Closed	Undecidable	Undecidable

The integral emptiness problem is only decidable under the standard semantics:

A	$\mathbb{Z}[\![A]\!] = \mathbb{Z}\mathbf{TT}\,?$	$\mathbb{Z}\widetilde{[\![A]\!]} = \mathbb{Z}\mathbf{TT}\,?$
Mixed	Decidable	Undecidable

Closure under digitization is only decidable under the standard semantics:

A	$[\![A]\!]$ CUD ?	$\widetilde{[\![A]\!]}$ CUD ?
Mixed	Decidable	Undecidable
Open	Decidable	Undecidable
Closed	Yes	Undecidable

On the other hand, closure under inverse digitization is undecidable for mixed and closed timed automata under the standard semantics, and otherwise holds:

A	$[\![A]\!]$ CUID ?	$\widetilde{[\![A]\!]}$ CUID ?
Mixed	Undecidable	Yes
Open	Yes	Yes
Closed	Undecidable	Yes

The robust integral languages of timed automata are in general not regular, unlike the case for the standard semantics. The languages of timed automata are recursive under either semantics:

Property	$\mathbb{Z}[\![A]\!]^{\checkmark}$	$\mathbb{Z}\widetilde{[\![A]\!]}^{\checkmark}$
Regular ?	Yes	Not in general
Recursive ?	Yes	Yes

We also note that the standard and robust semantics differ in expressiveness, but that both give rise to regular untimed languages.

We conclude by noting that the study of robustness for timed and hybrid automata is a rich subject which will undoubtedly lead to much further research. The works of [23], [13], [6], [14], among others, suggest that witnessing or introducing robustness usually improves the tractability of the various decision problems we have considered. Since our results point in the other direction, the conclusion must be that the *semantic* introduction of robustness studied here is counterproductive. Indeed, as open timed automata bear witness [22], it is much preferable to incorporate robustness *syntactically*, in other words to model systems with explicit safety margins.

References

[1] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proceedings of LICS 90*, pages 414–425. IEEE Computer Society Press, 1990.

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

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

[4] R. Alur and T. A. Henzinger. Back to the future: Towards a theory of timed regular languages. In *Proceedings of FOCS 92*, pages 177–186. IEEE Computer Society Press, 1992.

[5] R. Alur and R. P. Kurshan. Timing analysis in COSPAN. In *Proceedings of Hybrid Systems III*, volume 1066, pages 220–231. Springer LNCS, 1996.

[6] E. Asarin and A. Bouajjani. Perturbed Turing machines and hybrid systems. In *Proceedings of LICS 01*, pages 269–278. IEEE Computer Society Press, 2001.

[7] 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.

[8] A. Bouajjani, R. Echahed, and R. Robbana. Verifying invariance properties of timed systems with duration variables. In *Proceedings of FTRTFT 94*, volume 863, pages 193–210. Springer LNCS, 1994.

[9] A. Bouajjani, R. Echahed, and J. Sifakis. On model checking for real-time properties with durations. In *Proceedings of LICS 93*. IEEE Computer Society Press, 1993.

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

[11] D. Bošnački and D. Dams. Integrating real time in spin: A prototype implementation. In *Proceedings of FORTE/PSTV 98*, pages 423–438. Kluwer, 1998.

[12] E. M. Clarke and S. Campos. Real-time symbolic model checking for discrete time models. In *AMAST Series in Computing: Theories and Experiences for Real-Time System Development*. World Scientific, 1995.

[13] 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.

[14] M. Fränzle. What will be eventually true of polynomial hybrid automata? In *Proceedings of TACS 01*, volume 2215, pages 340–359. Springer LNCS, 2001.

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

[16] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of LICS 96*. IEEE Computer Society Press, 1996.

[17] 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.

[18] 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.

[19] J. E. Hopcroft and J. Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesley, New York, NY, 1979.

[20] J. Ouaknine. Digitisation and full abstraction for dense-time model checking. In *Proceedings of TACAS 02*, volume 2280, pages 37–51. Springer LNCS, 2002.

[21] J. Ouaknine and J. B. Worrell. Timed CSP = closed timed ε-automata. *Nordic Journal of Computing (to appear)*, 2003.

[22] J. Ouaknine and J. B. Worrell. Universality and language inclusion for open and closed timed automata. In *Proceedings of HSCC 03 (to appear)*. Springer LNCS, 2003.

[23] A. Puri. Dynamical properties of timed automata. In *Proceedings of FTRTFT 98*, volume 1486, pages 210–227. Springer LNCS, 1998.

[24] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In *Proceedings of CAV 94*, volume 818, pages 95–104. Springer LNCS, 1994.

[25] C. Bui Thanh, H. Klaudel, and F. Pommereau. Petri nets with causal time for system verification. In *Proceedings of MTCS 02*. ENTCS, 2002.