Robust Undecidability of Timed and Hybrid Systems* **

Thomas A. Henzinger[1] Jean-François Raskin[1,2]

[1] Department of Electrical Engineering and Computer Sciences
University of California at Berkeley, CA 94720-1770, USA
[2] Département d'Informatique, Faculté des Sciences
Université Libre de Bruxelles, Belgium
{tah,jfr}@eecs.berkeley.edu

Abstract. The algorithmic approach to the analysis of timed and hybrid systems is fundamentally limited by undecidability, of universality in the timed case (where all continuous variables are clocks), and of emptiness in the rectangular case (which includes drifting clocks). Traditional proofs of undecidability encode a single Turing computation by a single timed trajectory. These proofs have nurtured the hope that the introduction of "fuzziness" into timed and hybrid models (in the sense that a system cannot distinguish between trajectories that are sufficiently similar) may lead to decidability. We show that this is not the case, by sharpening both fundamental undecidability results. Besides the obvious blow our results deal to the algorithmic method, they also prove that the standard model of timed and hybrid systems, while not "robust" in its definition of trajectory acceptance (which is affected by tiny perturbations in the timing of events), is quite robust in its mathematical properties: the undecidability barriers are not affected by reasonable perturbations of the model.

1 Introduction

The main limitations of the algorithmic method for analyzing timed and hybrid systems find their precise expression in two well-publicized undecidability results. First, the universality problem for timed automata (does a timed automaton accept all timed words?) is undecidable [AD94]. This implies that timing requirements which are expressible as timed automata cannot be model checked. Consequently, more restrictive subclasses of timing requirements have been studied (e.g., Event-Clock Automata [AFH94], Metric Interval Temporal

* A preliminary version of this paper appeared in the *Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control* (HSCC 00), *Lecture Notes in Computer Science* **1790**, Springer-Verlag, 2000, pp. 145–159.

** This research was supported in part by the DARPA (NASA) grant NAG2-1214, the DARPA (Wright-Patterson AFB) grant F33615-C-98-3614, the ARO MURI grant DAAH-04-96-1-0341, and the NSF CAREER award CCR-9501708.

Logic [AFH96], Event-Clock Logic [RS99]). Second, the emptiness/reachability problem for rectangular automata (does a rectangular automaton accept any timed word, or equivalently, can a rectangular automaton reach a given location?) is undecidable [HKPV95]. While several orthogonal undecidability results are known for hybrid systems, it is the rectangular reachability problem which best highlights the essential limitations of the algorithmic approach to systems with continuous dynamics. This is because the rectangular automaton model is the minimal generalization of the timed automaton model capable of approximating continuous dynamics (using piecewise linear envelopes). It follows that rectangularity as an abstraction is insufficient for checking invariants of hybrid systems, and further loss of information is necessary (e.g., initialization [HKPV95], discretization [HK97]).

Both central undecidability results have been proved by encoding each computation of some Turing-complete machine model as a trajectory of a timed or hybrid system. The encodings are quite fragile: given a deterministic Turing machine M with empty input, one constructs either a timed automaton that rejects the single trajectory which encodes the halting computation of M (rendering universality undecidable), or a rectangular automaton that accepts that single trajectory (rendering emptiness/reachability undecidable). However, if the specified trajectory is perturbed in the slightest way, it no longer properly encodes the desired Turing computation. This has led researchers to conjecture [Fra99] that undecidability is due to the ability of timed and hybrid automata to differentiate real points in time with infinite precision. Consequently, one might hope that a more realistic, slightly "fuzzy" model of timed and hybrid systems might not suffer from undecidability.[1] In a similar vein, in [GHJ97] it is conjectured that unlike timed automata, robust timed automata, which do not accept or reject individual trajectories but bundles ("tubes") of closely related trajectories, can be complemented.

In this paper, we refute these conjectures. In doing so, we show that the sources of undecidability for timed and hybrid systems are structural, robust, and intrinsic to mixed discrete-continuous dynamics, rather than an artifact of a particular syntax or of the ability to measure time with arbitrary precision. We redo both undecidability proofs by encoding each Turing computation not as a single trajectory but as a trajectory tube of positive diameter. This requires considerable care and constitutes the bulk of this paper. As corollaries we obtain the following results:

Robust timed and rectangular automata Robust automata introduce "fuzziness" semantically, by accepting tubes rather than trajectories [GHJ97]. We prove that universality is undecidable for robust timed automata (since emptiness is decidable, it follows that they are not

[1] Note that "fuzziness," as meant here, is fundamentally distinct from "discretization," which is known to lead to decidability in many cases. Intuitively, fuzziness preserves the density of the time domain, while discretization does not. Mathematically, discretization is performed with respect to a fixed real $\epsilon > 0$ representing finite precision, while fuzziness quantifies over $\epsilon > 0$ existentially.

complementable), and that emptiness/reachability is undecidable for robust rectangular automata.

Open rectangular automata Open automata introduce "fuzziness" syntactically, by restricting all guard and differential-inclusion intervals to open sets. We prove that emptiness/reachability is undecidable for open rectangular automata. The universality problem for open timed automata is, to our knowledge, still open.

A main impact of these results is, of course, negative: they deal a serious blow to our ability for analyzing timed and hybrid systems automatically, much more so than the previously known results, which rely on questionable, "fragile" modeling assumptions (one trajectory may be accepted even if all slightly perturbed trajectories are rejected, and vice versa). There is, however, also a positive interpretation of our results: they show that the "standard" model for timed and hybrid systems, with its fragile definition of trajectory acceptance, does not give rise to a fragile theory but, on the contrary, is very robust with respect to its mathematical properties (such as decidability versus undecidability). For further decidability/undecidability results about the standard model of hybrid systems, we refer the reader to [AMP95,BT99].

2 Trajectories, Tubes, and Hybrid Automata

In this paper, we consider finite trajectories only. A *trajectory* over an alphabet Σ is an element of the language $(\Sigma \times \mathbb{R}^+)^*$, where $\mathbb{R}^+$ stands for the set of positive reals excluding 0. Thus, a trajectory is a finite sequence of pairs from $\Sigma \times \mathbb{R}^+$. We call the first element of each pair an *event*, and the second element the *time-gap* of the event. The time-gap of an event represents the amount of time that has elapsed since the previous event of the trajectory. For a trajectory τ, we denote its length (i.e., the number of pairs in τ) by $\mathrm{len}(\tau)$, and its projection onto Σ^* (i.e., the sequence of events that results from removing the time-gaps) by $\mathrm{untime}(\tau)$. We assign time-stamps to the events of a trajectory: for the i-th event of τ, the *time-stamp* is defined to be $t_\tau(i) = \sum_{1 \le j \le i} \delta_j$, where δ_j is the time-gap associated with the j-th event of τ.

Metrics on trajectories. Let the set of all trajectories be denoted Traj. Assuming that trajectories cannot be generated and recorded with infinite precision, in order to get an estimate of the amount of error in the data that represents a trajectory, we need a metric on Traj. Here we define, as an example, one particular metric d; in [GHJ97], it is shown that all reasonable metrics define the same topology on trajectories. Given two trajectories τ and τ', we define:

- $d(\tau, \tau') = \infty$ if $\mathrm{untime}(\tau) \neq \mathrm{untime}(\tau')$;
- $d(\tau, \tau') = \max\{|t_\tau(i) - t_{\tau'}(i)| : 1 \le i \le \mathrm{len}(\tau)\}$ if $\mathrm{untime}(\tau) = \mathrm{untime}(\tau')$.

Thus, only two trajectories with the same length and the same sequence of events have a finite distance, and finite errors may occur only in measuring time. The

metric measures the maximal difference in the time-stamps of any two corresponding events: two timed words are close to each other if they have the same events in the same order, and the times at which these events occur are not very different. For instance, for $\tau_1 = (a,1)(a,1)(a,1)$ and $\tau_2 = (a,0.9)(a,1.2)(a,1.2)$, we have $d(\tau_1, \tau_2) = 0.3$.

Given a metric, we use the standard definition of open sets. Formally, for the metric d, a trajectory τ, and a positive real $\epsilon \in \mathbb{R}^+$, define the *$d$-tube around τ of diameter ϵ* to be the set $T(\tau, \epsilon) = \{\tau' : d(\tau, \tau') < \epsilon\}$ of all trajectories at a d-distance less than ϵ from τ. A d-open set O, called a *d-tube*, is any subset of Traj such that for all trajectories $\tau \in O$, there is a positive real $\epsilon \in \mathbb{R}^+$ with $T(\tau, \epsilon) \subseteq O$. Thus, if a d-tube contains a trajectory τ, then it also contains all trajectories in some neighborhood of τ. Let the set of all d-tubes be denoted Tube.

From trajectory languages to tube languages. A *trajectory language* is any subset of Traj; a *tube language* [GHJ97] is any subset of Tube. Every trajectory language L induces a tube language $[L]$, which represents a "fuzzy" rendering of L. In $[L]$ we wish to include a tube iff sufficiently many of its trajectories are contained in L. We define "sufficiently many" as any dense subset, in the topological sense.

For this purpose we review some simple definitions from topology. A set S of trajectories is closed if its complement $S^c = \mathsf{Traj} - S$ is open. The closure $\overline{S}$ of a set S of trajectories is the least closed set containing S, and the interior S^{int} is the greatest open set contained in S. The set S' of trajectories is dense in S iff $S \subseteq \overline{S'}$. Formally, given a trajectory language L, the corresponding tube language is defined as $[L] = \{O \in \mathsf{Tube} : O \subseteq \overline{L}\}$. Thus, a tube O is in $[L]$ if for each trajectory $\tau \in O$ there is a sequence of trajectories with limit τ such that all elements of this sequence are in L. Equivalently, L must be dense in O; that is, for every trajectory $\tau \in O$ and for every positive real $\epsilon \in \mathbb{R}^+$, there is a trajectory $\tau' \in L$ such that $d(\tau, \tau') < \epsilon$. Since the tubes in $[L]$ are closed under subsets and union, the tube language $[L]$ can be identified with the maximal tube in $[L]$, which is the interior $\overline{L}^{int}$ of the closure of L.

We will define the semantics of a robust hybrid automaton with trajectory set L to be the tube set $[L]$. This has the effect that a robust hybrid automaton cannot generate (or accept) a particular trajectory when it refuses to generate (rejects) sufficiently many surrounding trajectories. Neither can the automaton refuse to generate a particular trajectory when it may generate sufficiently many surrounding trajectories.

Timed and rectangular automata. An *interval* has the form (a,b), $[a,b]$, $(a,b]$, or $[a,b)$, where $a \in \mathbb{Q} \cup \{-\infty\}$, $b \in \mathbb{Q} \cup \{\infty\}$, and $a \leq b$ if I is of the form $[a,b]$, and $a < b$ otherwise. We say that the interval I is *open* if it is of the form (a,b), and *closed* if it is of the form $[a,b]$. We write Rect for the set of intervals.

A *rectangular automaton* [HKPV95] is a tuple $A = \langle \Sigma, Q, Q_0, Q_f, C, E, \mathsf{Ev}, \mathsf{Init}, \mathsf{Pre}, \mathsf{Reset}, \mathsf{Post}, \mathsf{Flow} \rangle$[2], where (i) Σ is a finite alphabet of events; (ii) Q is a finite set of locations; (iii) $Q_0 \subseteq Q$ is a set of start locations; (iv) $Q_f \subseteq Q$ is a set of accepting locations; (v) C is a finite set of real-valued variables; (vi) $E \subseteq Q \times Q$ is a finite set of edges; (vii) $\mathsf{Ev} : E \to \Sigma$ is a function that associates with each edge e a letter of the alphabet Σ; (viii) $\mathsf{Init} : Q_0 \to C \to \mathsf{Rect}$ is a function that associates with each start location $q_0 \in Q_0$ and variable $x \in C$ an interval I that contains the possible initial values of this variable when the control of the automaton starts in location q_0; (ix) $\mathsf{Pre} : E \to C \to \mathsf{Rect}$ is a function that associates with each edge e and variable x an interval I such that the value of x must lie in I before crossing the edge e; (x) $\mathsf{Post} : E \to C \to \mathsf{Rect}$ is a function that associates with each edge e and each variable x an interval I such that the value of x must lie in I after crossing the edge e; (xi) $\mathsf{Reset} : E \to 2^C$ is a function that associates with each edge e a subset of variables that are reset when crossing e; if a variable x belongs to the set $\mathsf{Reset}(e)$ then the value, after crossing the edge e, of x is taken nondeterministically from the interval $\mathsf{Post}(e, x)$; (xii) $\mathsf{Flow} : Q \to C \to \mathsf{Rect}$ is a function that associates with each location q and variable x an interval I such that the first derivative of x when the control is in location q lies within I.

Timed automata are a syntactic subset of rectangular automata. A rectangular automaton A is a *timed automaton* [AD94] if the function Flow of A is such that for all locations $q \in Q$, and for all variables $x \in C$, we have $\mathsf{Flow}(q, x) = [1, 1]$; that is, every continuous variable is a clock. The timed automaton A is *open* if all intervals used in the functions Init, Pre, and Post are open. Similarly, a rectangular automaton A is *open* if all intervals used in the functions Init, Pre, Post, and Flow are open.

A rectangular automaton A defines a labeled transition system with an infinite state space S, the infinite set of labels $\mathbb{R}^+ \cup \Sigma$, and the transition relation R. Each transition with label σ correspond to an edge step whose event is $\sigma \in \Sigma$. Each transition with label $\delta \in \mathbb{R}^+$ corresponds to a time step of duration δ. The states and transitions of A are defined as follows. A *state* $(q, \mathbf{x})$ of A consists of a discrete part $q \in Q$ and a continuous part $\mathbf{x} \in \mathbb{R}^n$. The *state space* $S \subset Q \times \mathbb{R}^n$ is the set of all states of A. The state $(q, \mathbf{x})$ is an *initial state* of A if $q \in Q_0$ and $\mathbf{x} \in \mathsf{Init}(q)$[3]. For each edge $e = (q_1, q_2)$ of A, we define the binary relation $\to^e \subset S^2$ by $(q_1, \mathbf{x}) \to^e (q_2, \mathbf{y})$ iff $\mathbf{x} \in \mathsf{Pre}(e)$, $\mathbf{y} \in \mathsf{Post}(e)$, and for every coordinate $i \in \{1, \ldots, n\}$ with $i \notin \mathsf{Reset}(e)$, we have $\mathbf{x}_i = \mathbf{y}_i$. For each event $\sigma \in \Sigma$, we define the *edge-step relation* $\to^\sigma \subset S^2$ by $s_1 \to^\sigma s_2$ iff $s_1 \to^e s_2$ for some edge $e \in E$ with $\mathsf{Ev}(e) = \sigma$. For each positive real $\delta \in \mathbb{R}^+$, we define the binary *time-step relation* $\to^\delta \subset S^2$ by $(q_1, \mathbf{x}) \to^\delta (q_2, \mathbf{y})$ iff $q_1 = q_2$ and $\frac{\mathbf{y}-\mathbf{x}}{\delta} \in \mathsf{Flow}(q_1)$. The transition relation $R \subseteq S \times S$ is defined by $R = \{\to^e | \ e \in E\} \cup \{\to^\delta | \ \delta \in \mathbb{R}^+\}$.

[2] It is often convinient to annotate locations with variable constraints, so-called invariant conditions. Our results extend straight-forwardly to rectangular automata with invariant conditions.

[3] To simplify notations, we note $\mathbf{x} \in \mathsf{Init}(q)$ instead of $\mathbf{x} \in \mathsf{Init}(q, x)$.

Class of Automata	Emptiness/Reachability	Universality
Timed Automata [AD94]	Decidable	Undecidable
Rectangular Automata [HKPV95]	Undecidable	Undecidable

Fig. 1. Known decidability and undecidability results for timed/rectangular automata.

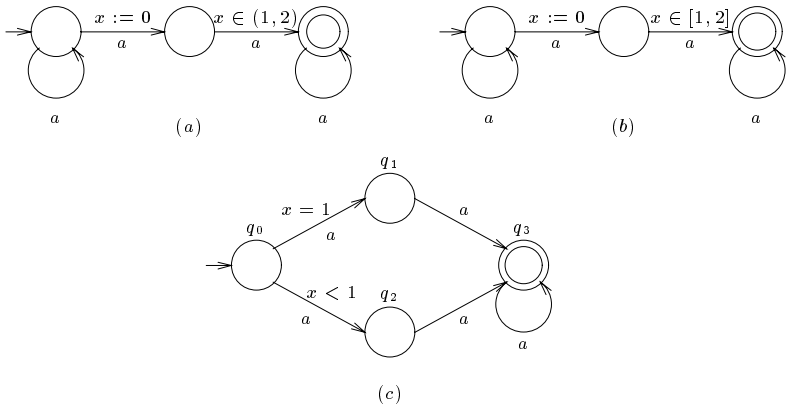

Fig. 2. The timed automata A_1, A_2, and A_3.

Trajectory acceptance and reachable locations. We now define the trajectory language and the reachable locations of a rectangular automaton A. A *run* of the automaton A is a finite path $(q_0, \mathbf{x}_0) \rightarrow^{\delta_0} (q_0, \mathbf{y}_0) \rightarrow^{\sigma_0} (q_1, \mathbf{x}_1) \rightarrow^{\delta_1} (q_1, \mathbf{y}_1) \ldots \rightarrow^{\sigma_n} (q_{n+1}, \mathbf{x}_{n+1})$ in the transition system of A that alternates between time steps and edge steps. The run is *initial* if $q_0 \in Q_0$ and $\mathbf{x}_0 \in \mathsf{Init}(q_0)$, and *accepting* if $q_n \in Q_f$. The trajectory $\tau = (\sigma_0, \delta_0)(\sigma_1, \delta_1) \ldots (\sigma_n, \delta_n)$ is *accepted* by the rectangular automaton A if A has an initial and accepting run $(q_0, \mathbf{x}_0) \rightarrow^{\delta_0} (q_0, \mathbf{y}) \rightarrow^{\sigma_0} (q_1, \mathbf{x}_1) \rightarrow^{\delta_1} \ldots \rightarrow^{\sigma_n} (q_{n+1}, \mathbf{x}_{n+1})$. The trajectory τ *leads to* location q_{n+1}. A location q of A is *reachable* if there exists an trajectory τ accepted by A that leads to q. We denote by $L(A)$ the set of trajectories accepted by A.

The *trajectory-emptiness problem* for a rectangular automaton A is to decide whether or not $L(A)$ is empty. The *trajectory-universality problem* for a rectangular automaton A is to decide whether or not $L(A)$ contains all trajectories over the alphabet Σ. The *location-reachability problem* for a rectangular automaton A is to decide if a given location of A is reachable. Note that the trajectory-emptiness problem for a class of rectangular automaton is decidable iff the location-reachability problem is decidable. The previously known results about these problems are summarized in the table of Figure 1.

Tube acceptance and robustly reachable locations. The rectangular automaton A *accepts* the set $[L(A)]$ of tubes [GHJ97]. The following examples il-

Class of Automata	Robust Emptiness/Robust Reachability	Robust Universality
Timed Automata	Decidable	Undecidable
Rectangular Automata	Undecidable	Undecidable

Fig. 3. Decidability results about robust timed and rectangular automata.

lustrate tube acceptance. First, consider the timed automaton A_1 of Figure 2(a). This automaton accepts all trajectories over the unary alphabet $\{a\}$ which contain two consecutive a events with a time-gap in the open interval $(1, 2)$. This property is invariant under sufficiently small perturbations of the time-stamps. Hence the automaton A_1 accepts precisely those tubes that consist of trajectories in $L(A_1)$, and the maximal accepted tube is $L(A_1)$ itself. In the timed automaton A_2 of Figure 2(b), the open interval $(1, 2)$ is replaced by the closed interval $[1, 2]$. This changes the set of accepted trajectories but not the set of accepted tubes: $L(A_1) \subset L(A_2)$ but $[L(A_1)] = [L(A_2)]$. Notice that the "boundary trajectories" accepted by A_2, with two consecutive a's at a time-gap of 1 or 2 but no consecutive a's at a time-gap strictly between 1 and 2, are not accepted robustly, because there are arbitrarily small perturbations that are not acceptable.

Let us now define the notion of robust reachability. A location q of a rectangular automaton A is *robustly reachable* if there exists a tube O accepted by A such that each trajectory in O leads to q. The automaton A_3 of Figure 2(c) illustrates this notion: the locations q_0, q_2, and q_3 are robustly reachable, while the location q_1 is not robustly reachable.

The *robust-emptiness problem* for a rectangular automaton A is to decide whether or not $[L(A)]$ is empty. The *robust-universality problem* for a rectangular automaton A is to decide whether or not $[L(A)]$ contains all tubes over Σ. The *robust-reachability problem* for a rectangular automaton A is to decide, given a location q of A, if q is robustly reachable. In the following sections of this paper, we will sharpen the known undecidability results about timed and hybrid systems. We will show that the introduction of fuzziness into timed and hybrid models via the notion of tubes (this fuzziness can be intuitively seen as the semantic removal of equality) does not change the undecidability results. Our results are summarized in the table of Figure 3; only the positive result was previously known [GHJ97].

Some properties of robust timed automata. We recall some results presented in [GHJ97]. We will need these notions to establish our results. The first proposition tells us that when we consider tube acceptance, we can restrict our attention either to closed or to open timed automata.

Proposition 1. *For every timed automaton A, we can construct a timed automaton $\overline{A}$, called the* closure *of A, whose* Pre, Post, Init *functions use only closed intervals, such that $L(\overline{A}) = \overline{L(A)}$. Furthermore, we can construct an open timed automaton A^{int}, called the* interior *of A, such that $[L(A)] = [L(A^{int})] = [L(\overline{A})]$.*

The following proposition shows that for open timed automata, tube emptiness coincides with trajectory emptiness.

Proposition 2. *For every open timed automaton A and every trajectory τ, if τ is accepted by A along some path, then there is a positive real $\epsilon \in \mathbb{R}^+$ such that all trajectories in the tube $T(\tau, \epsilon)$ are accepted by A along the same path.*

Before defining the tube complement of a timed automaton, we observe an important property of the trajectory languages that can be defined by timed automata.

Proposition 3. *For every timed automaton A, there is no tube O such that both $L(A)$ and its complement $L(A)^c$ are both dense in O.*

It follows that a tube cannot be accepted by both a timed automaton A and a trajectory complement of A.

For defining the tube complement of a timed automaton A, it is not useful to consider the boolean complement $\mathsf{Tube} - [L(A)]$ of the tube language $[L(A)]$. For $[L(A)]$ is closed under subsets and union. Therefore, unless $[L(A)] = \emptyset$ or $[L(A)] = \mathsf{Tube}$, the boolean complement $\mathsf{Tube} - [L(A)]$ cannot be induced by any trajectory language and, hence, cannot be accepted by any timed automaton. Thus, for every tube language $\mathcal{L} \subseteq \mathsf{Tube}$, we define the *tube complement* of $\mathcal{L}$ to be the set

$$\mathcal{L}^c = \{O \in \mathsf{Tube} : O \cap \bigcup \mathcal{L} = \emptyset\}$$

of tubes that are disjoint from the tubes in $\mathcal{L}$. The following proposition shows that for every timed automaton A, the tube complement $[L(A)]^c$ is induced by the trajectory complement $L(A)^c$; that is, $[L(A)^c] = [L(A)]^c$.

Proposition 4. *If L is a trajectory language and there is no tube O such that both L and L^c are dense in O, then $[L]^c = [L^c]$.*

For two timed automata A and B, we say that B is a *tube complement* of A iff B accepts precisely the tubes that do not intersect any tube accepted by A; that is, $[L(B)] = [L(A)]^c$. From Propositions 3 and 4, it follows that every trajectory complement of a timed automaton is also a tube complement (the converse is generally not true). Since $[L(A)]^c = [L(A^{int})]^c = [L(A^{int})^c]$, in order to construct tube complements, it would suffice to construct trajectory complements of open timed automata.[4] This, however, is not possible, as we show in the next section.

3 The Robust-Universality Problem for Timed Automata

In this section, we show that the halting problem for two-counter machines can be reduced to the robust-universality problem for timed automata. A *two-counter*

[4] Similarly, since $[L(A)]^c = [L(\overline{A})]^c = [L(\overline{A})^c]$, it would suffice to construct trajectory complements of closed timed automata. This, however, is known to be impossible [AD94].

machine M is a triple $\langle\{b_1,\ldots,b_n\},C,D\rangle$, where $\{b_1,\ldots,b_n\}$ are n instructions, and C and D are two counters ranging over the natural numbers. Each instruction b_i, $0 \leq i \leq n$, has one of the three possible forms: (i) a conditional jump instruction tests if a counter is 0 and then jumps conditionally to the next instruction; (ii) an increment/decrement instruction increments or decrements the value of one of the two counters and then jumps nondeterministically to one of two possible next instructions; (iii) a stop instruction puts an end to the machine execution. A *configuration of a two-counter machine* M is a triple $\gamma = \langle i,c,d\rangle$, where i is the program counter indicating the current instruction, and c and d are the values of the counters C and D. A *computation* of M is a finite or infinite sequence $\overline{\gamma} = \gamma_0\gamma_1\ldots$ of configurations such that $\gamma_0 = \langle 0,0,0\rangle$, i.e. the first instruction is b_0, and the initial value of the two counters C and D is 0, and for every γ_{i+1} is a M-successor configuration of γ_i, for every $i \geq 0$. If $\overline{\gamma}$ is finite then its last configuration contains a stop instruction. The *halting problem* for a two-counter machine M is to decide whether or not the execution of M has at least one computation that ends in a stop instruction. The problem of deciding if a two-counter machine has a halting computation is undecidable.

Trajectory encoding of a two-counter machine computation. We review how the undecidability of the universality problem for timed automata was established by Alur and Dill [AD94] and explain why their proof does not translate to the robust-universality problem. Given a two-counter machine M, the set $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Traj}}(M)$ of trajectories is defined as follows: $(\sigma,\delta) \in \mathsf{L}^{\mathsf{Undec}}_{\mathsf{Traj}}(M)$ iff (i) $\sigma = b_{i_0}c^{c_0}d^{d_0}b_{i_1}c^{c_1}d^{d_1}\ldots b_{i_m}c^{c_m}d^{d_m}$ such that $\langle i_0,c_0,d_0\rangle,\langle i_1,c_1,d_1\rangle,\ldots\langle i_m,c_m,d_m\rangle$ is a halting computation of M; (ii) for all $j \geq 0$, the time-stamp of b_{i_j} is j; (iii) for all $j \geq 1$, (a) if $c_{j+1} = c_j$, then for every c with time-stamp t in the interval $(j,j+1)$ there is a c with time-stamp $t+1$; (b) if $c_{j+1} = c_j + 1$, then for every c with time-stamp t in the interval $(j+1,j+2)$, except the last one, there is a c with time-stamp $t-1$; (c) if $c_{j+1} = c_j - 1$, then for every c with time-stamp t in the interval $(j,j+1)$, except the last one, there is a c with time-stamp $t+1$; and (iv) the same requirements hold for d's. Then $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Traj}}(M)$ is nonempty iff M has a halting computation. Furthermore, there exists a timed automaton that accepts exactly the trajectories not in $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Traj}}(M)$. It follows that the universality problem for timed automata is undecidable.

Note that the i-th configuration is encoded in the interval $[i,i+1)$. To enforce the requirement that the number of c events in two successive configurations is the same, every c in the first interval has a matching c at the exact distance 1, and vice versa. This use of punctuality constraints has the following consequence.

Proposition 5. *Let M be a two-counter machine, there is no tube $O \in \mathsf{Tube}$ such that O is dense in $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Traj}}(M)$; that is, $[\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Traj}}(M)] = \emptyset$.*

This has nurtured some hope that, by removing the possibility to specify punctuallity constraints, timed automata might have a decidable robust-universality problem. Unfortunately this is not the case. We next show that we can define a set $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Tube}}(M)$ of trajectories which forms a tube and encodes halting computations of the given two-counter machine M. Furthermore the tube complement

of this tube language can be defined by a robust (open) timed automaton. The undecidability of the robust-universality problem and the nonclosure under complement of robust timed automata will follow.

Tube encoding of a two-counter machine computation. To facilitate the definition of $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Tube}}(M)$, the undecidable tube language, we first introduce some new notions. We call an *open (closed) slot* an open (closed) interval of the real numbers. We define the open (closed) slot between t_1 and t_2 as the set $\{t \mid t_1 < t < t_2\}$ (respectively, $\{t \mid t_1 \leq t \leq t_2\}$). Given two real numbers t_1 and t_2 with $t_1 < t_2$, we say that (t_3, t_4) (respectively $[t_3, t_4]$) is the open (closed) slot *generated* by t_1 and t_2 if both $t_1 + 1 = t_3$ and $t_2 + 1 = t_4$.

The main idea of $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Tube}}(M)$ is that we encode the configuration i within the open interval $(i, i+1)$, and the next configuration $i+1$ will be encoded in the open slot generated by the time of the beginning and the end of configuration i. For the encoding of the elements of a configuration and their relation with the next configuration we also use open slots. For instance, we use the triple $\mathsf{B}^{\mathsf{Inst}} \cdot b_{j_i} \cdot \mathsf{E}^{\mathsf{Inst}}$ to encode that b_{j_i} is the instruction executed in the i-th configuration; the letters $\mathsf{B}^{\mathsf{Inst}}$ and $\mathsf{E}^{\mathsf{Inst}}$ are used as delimiters of the instruction, and to generate the slot for the next instruction. Let us assume that t_1 and t_2 are the time-stamps of $\mathsf{B}^{\mathsf{Inst}}$ and $\mathsf{E}^{\mathsf{Inst}}$, respectively. Then the encoding of the next instruction has to take place in the open slot $(t_1 + 1, t_2 + 1)$ generated by the slot for the current instruction. As we use a dense time domain, this constraint can always be satisfied. We will proceed in the same way for the encoding of the values of the two counters. The value of the counters C and D are encoded as follows: if the value of the counter C is u in configuration i, then the pair $\mathsf{b}^{\mathsf{c}} \cdot \mathsf{e}^{\mathsf{c}}$ is repeated u times in the encoding of the configuration i. If the counter C is unchanged from configuration i to configuration $i+1$, we verify that the $\mathsf{b}^{\mathsf{c}} \cdot \mathsf{e}^{\mathsf{c}}$ sequences in configuration $i+1$ appear exactly in the open slots defined by the $\mathsf{b}^{\mathsf{c}} \cdot \mathsf{e}^{\mathsf{c}}$ sequences in configuration i.

Having the intuition underlying the language $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Tube}}(M)$, we now define it and establish that the set of trajectories in $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Tube}}(M)$ correspond to a non empty set of tubes iff the machine M has a halting computation. The set of events that we will use in the encoding is the following: (i) $\mathsf{B}^{\mathsf{Conf}}$ and $\mathsf{E}^{\mathsf{Conf}}$ are the delimiters for the beginning and end of the encoding of a configuration; (ii) $\mathsf{B}^{\mathsf{Inst}}$ and $\mathsf{E}^{\mathsf{Inst}}$ are the delimiters for the begin and end of the encoding of the instruction executed in a configuration; (iii) $b_1, b_2, \ldots, b_n$ are used to represent the n instructions; (iv) B^{C} and E^{C} are the delimiters for the encoding of the value of the counter C in a configuration; (v) B^{D} and E^{D}, for the counter D; (vi) b^{c} and e^{c} are used to encode the value of the counter C; (vii) b^{d} and e^{d}, for D. The trajectories of $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Tube}}(M)$ agree with the following regular expression: $(\mathsf{B}^{\mathsf{Conf}} \cdot \mathsf{B}^{\mathsf{Inst}} \cdot (b_1 \mid b_2 \mid \ldots \mid b_n) \cdot \mathsf{E}^{\mathsf{Inst}} \cdot \mathsf{B}^{\mathsf{C}} \cdot (\mathsf{b}^{\mathsf{c}} \cdot c \cdot \mathsf{e}^{\mathsf{c}})^* \cdot \mathsf{E}^{\mathsf{C}} \cdot \mathsf{B}^{\mathsf{D}} \cdot (\mathsf{b}^{\mathsf{d}} \cdot d \cdot \mathsf{e}^{\mathsf{d}})^* \cdot \mathsf{E}^{\mathsf{D}} \cdot \mathsf{E}^{\mathsf{Conf}})^*$. Furthermore, if the configuration i contains the sequence $\mathsf{B}^{\mathsf{Inst}} \cdot b_{j_i} \cdot \mathsf{E}^{\mathsf{Inst}}$, then the configuration $i+1$ contains the sequence $\mathsf{B}^{\mathsf{Inst}} \cdot b_{j_{i+1}} \cdot \mathsf{E}^{\mathsf{Inst}}$, where $b_{j_{i+1}}$ is a valid next instruction of b_{j_i}. The first configuration is encoded in the open interval $(0, 1)$; that is, if the event $\mathsf{B}^{\mathsf{Conf}}$ occurs at time t_1 and the event $\mathsf{E}^{\mathsf{Conf}}$ occurs at time t_2, then $0 < t_1 < t_2 < 1$. The configuration $i+1$ is always encoded in the open slot defined by the configuration i; that is, if the event $\mathsf{B}^{\mathsf{Conf}}$ of configuration

i occurs at time t_1 and the event $\mathsf{E}^{\mathsf{Conf}}$ occurs at time t_2, then the encoding of the configuration $i+1$ takes place in the open slot (t_1+1, t_2+1). The encoding of the instruction executed during the configuration $i+1$ takes place in the slot defined by the encoding of the instruction executed in configuration i; that is, if $\mathsf{B}^{\mathsf{Inst}}$ and $\mathsf{E}^{\mathsf{Inst}}$ appear at times t_1 and t_2 in encoding of configuration i, then $\mathsf{B}^{\mathsf{Inst}}$ and $\mathsf{E}^{\mathsf{Inst}}$ appear at times t_3 and t_4 in the encoding of configuration $i+1$ with the following (open) real-time constraint: $t_1+1 < t_3 < t_4 < t_2+1$. We only explain in details the case when the counter C is incremented from configuration i to configuration $i+1$. The other operations are left to the reader. If in configuration i the events B^{C} and E^{C} occur at times t_1 and t_2, respectively, then the events B^{C} and E^{C} appear for configuration $i+1$ within the open slot (t_1+1, t_2+1). For each $\mathsf{b}^{\mathsf{c}} \cdot \mathsf{e}^{\mathsf{c}}$ sequence, such that b^{c} occurs at time t_1 and e^{c} occurs at time t_2, in the encoding of configuration i, there is exactly one sequence $\mathsf{b}^{\mathsf{c}} \cdot \mathsf{e}^{\mathsf{c}}$ sequence in the encoding of configuration $i+1$ that takes place in the open slot (t_1+1, t_2+1). Conversely, each $\mathsf{b}^{\mathsf{c}} \cdot \mathsf{e}^{\mathsf{c}}$ that appears in the encoding of the configuration $i+1$, with the exception of the last, must lie in the open slot defined by the $\mathsf{b}^{\mathsf{c}} \cdot \mathsf{e}^{\mathsf{c}}$ sequence of configuration i. This requirement is noted RT_3^c. Finally, the last $\mathsf{b}^{\mathsf{c}} \cdot \mathsf{e}^{\mathsf{c}}$ sequence in the encoding of configuration $i+1$ appears in the slot generated by the two events B^{C} and E^{C} if $C = 0$ in configuration i, and appears in the slot generated by the last e^{c} event and E^{C} event of configuration i if $C > 0$ in that configuration.

The following proposition is a direct consequence of the use of strict inequalities in the definition of the language $\mathsf{L}_{\mathsf{Tube}}^{\mathsf{Undec}}(M)$.

Proposition 6. *Let M be a two-counter machine, for every trajectory τ_1 that belongs to $\mathsf{L}_{\mathsf{Tube}}^{\mathsf{Undec}}(M)$, there exists a real $\epsilon > 0$ such that for every trajectory τ_2, if $d(\tau_1, \tau_2) < \epsilon$ then $\tau_2 \in \mathsf{L}_{\mathsf{Tube}}^{\mathsf{Undec}}(M)$.*

Corollary 1. *For every two-counter machine M with a halting computation, $[\mathsf{L}_{\mathsf{Tube}}^{\mathsf{Undec}}(M)]$ is a nonempty tube language.*

Corollary 2. *There is no tube O that is dense both in $\mathsf{L}_{\mathsf{Tube}}^{\mathsf{Undec}}(M)$ and in $(\mathsf{L}_{\mathsf{Tube}}^{\mathsf{Undec}}(M))^c$.*

Note also that by Proposition 6 and Corollary 2, we know that the tube semantics of a timed automaton that accepts the complement of the trajectories of $\mathsf{L}_{\mathsf{Tube}}^{\mathsf{Undec}}(M)$, is exactly the complement of the tube language $[\mathsf{L}_{\mathsf{Tube}}^{\mathsf{Undec}}(M)]$. The following lemma shows that it is possible to construct such a timed automaton.

Lemma 1. *There exists a timed automaton A_M that accepts exactly the trajectories that are not in $\mathsf{L}_{\mathsf{Tube}}^{\mathsf{Undec}}(M)$.*

Proof. It is sufficient to show that for each of the requirements defining $\mathsf{L}_{\mathsf{Tube}}^{\mathsf{Undec}}(M)$, we can construct a timed automaton that accepts exactly the trajectories that violate the requirement. The union of these automata is exactly what we are looking for: the timed automaton that accepts the trajectory complement of $\mathsf{L}_{\mathsf{Tube}}^{\mathsf{Undec}}(M)$. Due to the lack of space, we just give here the automaton for the complement of requirement RT_3^c; the other requirements can be found in [HR99].

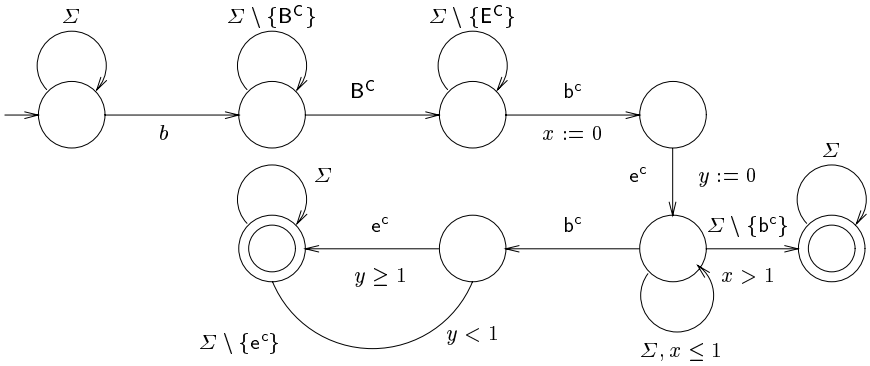

Fig. 4. A timed automaton for the negation of requirement RT_3^c

The timed automata for requirement RT_3^c is shown in Figure 4. This automaton accepts exactly the trajectories which contain two adjacent configurations i and $i+1$ such that (i) the instruction executed in configuration i increments the counter C, that is $b \in I^C$, where I^C is the subset of instructions that increment the counter C; (ii) there is a sequence $\mathsf{b}^c \cdot \mathsf{e}^c$ in configuration i that defines an open slot in configuration $i+1$ which does not contain the sequence $\mathsf{b}^c \cdot \mathsf{e}^c$. □

Combining Lemma 1 and Proposition 4, we obtain the following theorem.

Theorem 1. *For every two-counter machine M, there exists a timed automaton A_M that accepts every tube iff the two-counter machine M has no halting computation.*

Corollary 3. *The robust-universality problem for timed automata is undecidable.*

As the robust-emptiness problem for timed automata is decidable, we obtain the following:

Corollary 4. *There is a tube language definable by a timed automaton whose tube-complement is not definable by a timed automaton.*

From these results we can derive the following result about the trajectory languages of *open* timed automata (already established in [Her98]):

Theorem 2. *There is a trajectory language definable by an open timed automaton which trajectory-complement is not definable by a timed automaton (open or not).*

Proof. By reductio ad absurdum. We have constructed a timed automaton A_M that accepts the complement of the trajectories contained in $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Tube}}(M)$. This automaton A_M defines a set $L(A_M)$ of trajectories such that $[L(A_M)]$ is exactly the tube complement of $[\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Tube}}(M)]$. By Proposition 1, there exists an open timed

automaton, namely, the interior of A_M, denoted A_M^{int}, such that $[L(A_M^{int})] = [L(A_M)] = [\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Tube}}(M)]^c$. By Lemma 4, if we were able to complement the open automaton A_M^{int}, then we could obtain an automaton whose tube semantics would be $[\mathsf{L}^{\mathsf{Undec}}_{\mathsf{Tube}}(M)]$. This, however, is impossible, as the robust-emptiness problem of timed automata is decidable, which would allow us to decide the halting problem for two-counter machines. □

4 The Robust-Reachability Problem for Rectangular Automata

In this section we investigate undecidable reachability problems and show that they remain undecidable even when we remove equality from the specification formalism. In [HKPV95], it is shown that the formalism of rectangular automata lies at the boundary between decidable hybrid formalisms and undecidable ones. We show here that this boundary stays valid if we do not use equality. For this purpose, we use another tube encoding of two-counter machines computations. With each halting computation $\langle i_0, c_0, d_0\rangle, \langle i_1, c_1, d_1\rangle, \ldots, \langle i_n, c_n, d_n\rangle$, we associate the tube

$$(b_{i_j}, t_{(j,0)}), (\mathsf{B}^{\mathsf{C}}, t_{(j,1)}), (\mathsf{b}^{\mathsf{c}}, t_{(j,2)}), (\mathsf{e}^{\mathsf{c}}, t_{(j,3)})(\mathsf{B}^{\mathsf{D}}, t_{(j,4)})(\mathsf{b}^{\mathsf{d}}, t_{(j,5)})(\mathsf{e}^{\mathsf{d}}, t_{(j,6)})$$

with $0 \leq j \leq n$ and the following timing constraints. We just give the constraints for the encoding of the value of counter C; the same requirements hold for the counter D. Initially the value of the counter C is zero. To encode $C = 0$, we require that if the events B^{C},b^{c}, and e^{c} are issued at times t_1, t_2, and t_3, then the following constraint is satisfied: $t_1 + \frac{1}{2} < t_2 < t_3 < t_1 + 1$. Let d_1 denote the distance that separates the events B^{C} and b^{c}, and let d_2 denote the distance that separates the events B^{C} and e^{c} in the encoding of the value of C in configuration i. In the same way, let d_3 and d_4 be those two distances in the encoding of the value of C in configuration $i+1$. Then we have the following requirements: (a) if C is incremented between i and $i+1$, then $\frac{d_1}{2} < d_3 < d_4 < \frac{d_2}{2}$; (b) if C is decremented between i and $i+1$, then $2d_1 < d_3 < d_4 < 2d_2$; (c) if C is unchanged between i and $i+1$, then $d_1 < d_3 < d_4 < d_2$. We denote this trajectory language $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{OpenRect}}(M)$.

Lemma 2. *The trajectory language* $\mathsf{L}^{\mathsf{Undec}}_{\mathsf{OpenRect}}(M)$ *is definable by an open rectangular automaton* A_M.

Proof. We sketch the proof by giving an open rectangular automaton to increment the counter C. The automaton is given in Figure 6. To see that the automaton checks exactly the desired constraints, we first establish bounds on the values of the variables x and y at times t_0, t_1, t_2, and t_3 represented in Figure 7. The bounds are given in the table of Figure 5. So at time t_3, we have $x \in (d_1, +\infty)$ and $y \in (-\infty, d_2)$. Now let us see the constraints that we obtain on d_3 and d_4. First, by taking into account that $x \in (d_1, +\infty)$ at t_3 and the flow of x in q_5 in included in the interval $(-2, 0)$, we can deduce that $d_3 \in (\frac{d_1}{2}, +\infty)$.

	t_0	t_1	t_2	t_3
$\mathsf{Inf}(x)$	0	d_1	d_1	d_1
$\mathsf{Sup}(x)$	1	$2 \times d_1 + 1$	$d_1 + d_2 + 1$	$+\infty$
$\mathsf{Inf}(y)$	-1	-1	-1	$-\infty$
$\mathsf{Sup}(y)$	0	d_1	d_2	d_2

Fig. 5. Inferior and superior bounds on the values of variables x and y.

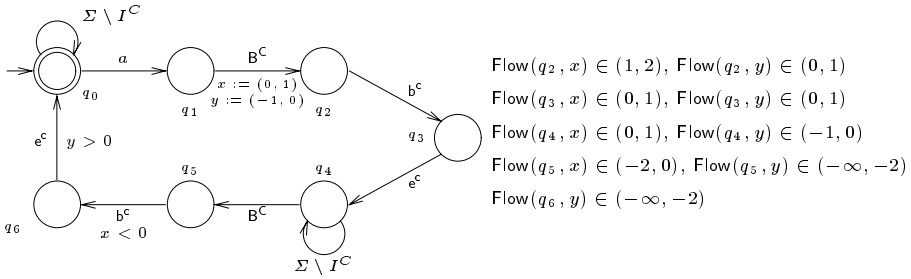

Fig. 6. Open rectangular automaton to check incrementation of counter C.

Second, by taking into account that $y \in (-\infty, d_2)$ at t_3 and that the flow of y in q_5 is included in the interval $(-\infty, -2)$, we obtain $d_3 \in (-\infty, \frac{d_2}{2})$. As b^{c} is issued before e^{c}, we have $d_3 < d_4$, and thus $\frac{d_1}{2} < d_3 < d_4 < \frac{d_2}{2}$, as desired. □

As a direct consequence of the last lemma, we have the following.

Theorem 3. *The trajectory-emptiness and location-reachability problems for open rectangular automata are undecidable.*

The following proposition is a generalization to open rectangular automata of Proposition 2.

Proposition 7. *For every open rectangular automaton A and every trajectory τ, if τ is accepted by A along some path, then there is a positive real $\epsilon \in \mathbb{R}^+$ such that all trajectories in the tube $T(\tau, \epsilon)$ are accepted by A along the same path.*

This proposition implies that tube and trajectory emptiness coincide for open rectangular automata, so we have the following theorem.

Theorem 4. *The robust-emptiness and robust-reachability problems for rectangular automata are undecidable.*

References

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

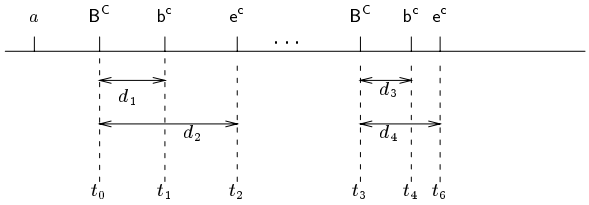

Fig. 7. Two successive encodings of the value of counter C.

[AFH94] R. Alur, L. Fix, and T.A. Henzinger. A determinizable class of timed automata, *CAV 94: Computer-aided Verification*, LNCS 818, Springer Verlag, 1–13, 1994.

[AFH96] R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[AMP95] E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138:65-66, 1995.

[BT99] V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. To appear in Automatica, 1999.

[Fra99] M. Franzle. Analysis of Hybrid Systems: An ounce of realism can save an infinity of states, *CSL'99: Computer Science Logic*. LNCS 1683, Springer Verlag, 126–140, 1999.

[GHJ97] V. Gupta, T.A. Henzinger, and R. Jagadeesan. Robust timed automata, *HART 97: Hybrid and Real-time Systems*. LNCS 1201, Springer-Verlag, 331–345, 1997.

[HK97] T.A. Henzinger and P.W. Kopke Discrete-time control for rectangular hybrid automata. *ICALP 97: Automata, Languages, and Programming*. LNCS 1256, Springer-Verlag, 582–593, 1997.

[HKPV95] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *27th Annual Symposium on Theory of Computing*, ACM Press, 373–382, 1995.

[HR99] T.A. Henzinger and J.-F. Raskin. *Robust Undecidability of Timed and Hybrid Systems*. Technical Report of the Computer Science Department of the University of California at Berkeley, UCB/CSD-99-1073, October 1999.

[Her98] P. Herrmann. Timed automata and recognizability. *Information Processing Letters*, 65(6):313-318, 1998.

[RS99] J.-F. Raskin and P.-Y. Schobbens. The Logic of Event Clocks: Decidability, Complexity, and Expressiveness. *Journal of Automata, Languages and Combinatorics*, 4(3):247-284, 1999.