Robust Timed Automata*

Vineet Gupta[1] Thomas A. Henzinger[2] Radha Jagadeesan[3]

[1] Xerox PARC, 3333 Coyote Hill Road, Palo Alto, CA 94304; vgupta@parc.xerox.com
[2] EECS Department, University of California, Berkeley, CA 94720; tah@eecs.berkeley.edu
[3] Mathematical Sciences Department, Loyola Univ.-Lake Shore Campus, Chicago, IL 60626; radha@math.luc.edu

Abstract. We define *robust timed automata*, which are timed automata that accept all trajectories "robustly": if a robust timed automaton accepts a trajectory, then it must accept neighboring trajectories also; and if a robust timed automaton rejects a trajectory, then it must reject neighboring trajectories also. We show that the emptiness problem for robust timed automata is still decidable, by modifying the region construction for timed automata. We then show that, like timed automata, robust timed automata cannot be determinized. This result is somewhat unexpected, given that in temporal logic, the removal of real-time equality constraints is known to lead to a decidable theory that is closed under all boolean operations.

1 Introduction

The formalism of timed automata [AD94] has become a standard model for real-time systems, and its extension to hybrid automata [ACHH93, ACH+95, Hen96] has become a standard model for mixed discrete-continuous systems. Yet it may be argued that the precision inherent in the formalism of timed and hybrid automata gives too much expressive power to the system designer. For example, while there is a timed automaton A that issues an event a at the exact real-numbered time t, such a system cannot be realized physically. This is because for every physical realization R_A of A there is a positive real ϵ, however small, so that one can guarantee at most that R_A issues the event a in the time interval $(t-\epsilon, t+\epsilon)$. The discretization of time into units of size ϵ, on the other hand, may not allow a sufficiently abstract representation of A. Among the reasons for leaving the precision ϵ parametric are the following: the actual value of ϵ may be unknown; future realizations of A may achieve a precision smaller than ϵ; if A is an open system, it may be composed with systems whose precision is smaller than ϵ; a small ϵ may cause a dramatic increase in the state space.

Similarly, consider two hybrid automata $B_{\leq}$ and $B_{<}$ for modeling the controller of a chemical plant. The two automata are identical except that $B_{\leq}$ activates a furnace iff the plant temperature falls to T degrees, and $B_{<}$ activates the furnace iff the plant temperature falls *below* T degrees. These two formal objects differ, and may have entirely different mathematical properties; for example, some plant transition may be possible only at temperatures less than T, thus causing any number of states to be reachable

* The first and third author were supported in part by grants from ARPA and ONR. The second author was supported in part by the ONR YIP award N00014-95-1-0520, by the NSF CAREER award CCR-9501708, by the NSF grant CCR-9504469, by the AFOSR contract F49620-93-1-0056, by the ARO MURI grant DAAH-04-96-1-0341, by the ARPA grant NAG2-892, and by the SRC contract 95-DC-324.036. The third author was also supported by the NSF.

in $B_<$ but not in $B_\leq$. Yet the difference between the two automata cannot be realized physically, because every physical thermometer has a positive error ϵ, however small, and cannot reliably distinguish between T and $T-\epsilon$ degrees. Again, the discretization of temperature into ϵ-units may not be adequate for reasons given above.

We remove the "excessive" expressive power of timed and hybrid automata without discretization, by having automata define (i.e., generate or accept) not individual trajectories, but bundles of closely related trajectories. A bundle of very similar trajectories is called a *tube*. For example, while a single trajectory τ may have event a at time t, every tube containing τ also contains some trajectories with event a very close to, but not exactly at time t. Formally, we suggest several metrics on the trajectories of timed and hybrid automata, and define a tube to be an *open* set of trajectories. This definition is shown to be independent of the choice of metric, because the "reasonable" metrics all induce the same topology. Then, a tube is accepted by a timed or hybrid automaton iff the accepted trajectories form a *dense* subset of the tube. Accordingly, while "isolated" accepted trajectories do not belong to any accepted tube, isolated rejected trajectories are added to accepted tubes, as motivated by the observation that an automaton ought not be able to accept or reject individual trajectories.

Timed and hybrid automata with tube acceptance are called *robust*, because they are insensitive against small input perturbations (and they may produce small output perturbations). In this paper, we look at some theoretical implications of robustness. First, we solve the emptiness problem for robust timed automata: given a timed automaton A, does A accept any tube? Our emptiness check for tube acceptance is derived from the region method of [AD94] for trajectory acceptance, but is somewhat more efficient, because only open regions need be considered. The emptiness check leads, in the usual way, to algorithms for verifying requirements of robust timed automata that are specified in a linear-time logic such as MITL [AFH96], in a branching-time logic such as TCTL [ACD93], or by event-clock automata [AFH94].

Second, we study the complementation problem for robust timed automata. Complementation is instrumental for using automata as a requirements specification language: abstract requirements of trajectories are often specified naturally using nondeterministic automata; then, in order to check that all trajectories that are generated by an implementation automaton A are accepted by the specification automaton B, the latter needs to be complemented (before checking the product of A and $\neg B$ for emptiness). While timed automata with trajectory acceptance are not closed under complement [AD94] (i.e., there is a timed automaton whose rejected trajectories are not the accepted trajectories of any other timed automaton), one may harbor some hope that robust timed automata can be complemented (i.e., for every timed automaton B there may be a timed automaton $\neg B$ that accepts precisely the tubes which are disjoint from the tubes accepted by B). This hope stems from the following observations:

1. In the case of linear-time temporal logic, the removal of all timing constraints that enforce exact real-numbered time differences between events leads to a decidable theory, called MITL, which is closed under all boolean operations [AFH96]. It is therefore not unreasonable to expect that in the case of timed automata, the removal of individual trajectories, which express exact real-numbered time differences between events, leads likewise to a decidable and boolean-closed theory.
2. The impossibility of complementation for timed automata follows from the fact that while the emptiness problem is decidable, the universality problem (i.e., given a timed automaton, does it accept all trajectories?) is not [AD94]. Undecidabil-

ity proofs for real-time problems, however, typically depend on an encoding of Turing-machine computations which uses the exact real-numbered times available in individual trajectories. These proofs do not straight-forwardly extend to tubes.
3. Since the complement of an open set is closed, the definition of complementation for timed automata with tube acceptance does not coincide with the definition of complementation for timed automata with trajectory acceptance.

We considerably dampen the hope that robust timed automata can be complemented by proving that, like ordinary timed automata, robust timed automata cannot be determinized (which is the usual first step in complementation). Indeed, the theory of ordinary timed automata turns out to be remarkably *robust* (pun intended) against perturbations in the definition of the automata: our results show that neither the syntactic removal of equality from timing constraints (open timing constraints only) nor the semantic removal of equality (tube acceptance) alter the theory of timed automata qualitatively.

2 Trajectories and Tubes

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 (the first time-gap can be thought of representing the amount of time that has elapsed since the "beginning of time"). For a trajectory τ, we denote its length (i.e., the number of pairs in τ) by $\text{len}(\tau)$, and its projection onto Σ^* (i.e., the sequence of events that results from removing the time-gaps) by $\text{untime}(\tau)$. For $1 \leq i \leq \text{len}(\tau)$, we denote the i-th event of τ by $a_\tau(i)$, and the i-th time-gap by $\delta_\tau(i)$. We also assign time-stamps to the events of a trajectory: for the i-th event of τ, the *time-stamp* is defined to be $t_\tau(i) = \sum_{1 \leq j \leq i} \delta_\tau(j)$.

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**. We will not choose a specific metric, but give some examples of "reasonable" metrics, and then state a condition on "reasonableness" that will be sufficient for all later results.

For all metrics d we consider, given two trajectories τ and τ', we define $d(\tau, \tau') = \infty$ if $\text{untime}(\tau) \neq \text{untime}(\tau')$. Thus, only two trajectories with the same sequence of events have a finite distance, and finite errors may occur only in measuring time. In the following examples, assume that $\text{untime}(\tau) = \text{untime}(\tau')$.

Example 1. Define

$$d_{max}(\tau, \tau') = \max\{|t_\tau(i) - t_{\tau'}(i)| : 1 \leq i \leq \text{len}(\tau)\}.$$

This 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 $d_{max}(\tau_1,\tau_2) = 0.3$. □

Example 2. The following metric considers the sum of all differences in the time-stamps:

$$d_{sum}(\tau,\tau') \;=\; \sum\{|t_\tau(i) - t_{\tau'}(i)| : 1 \le i \le \mathrm{len}(\tau)\}.$$

For instance, $d_{sum}(\tau_1,\tau_2) = 0.5$. □

Example 3. Another metric considers the pairwise time-differences between any two events of a trajectory:

$$d_{allpair}(\tau,\tau') \;=\; \max\{|\sum_{i<k\le j} (\delta_\tau(k) - \delta_{\tau'}(k))| : 0 \le i < j \le \mathrm{len}(\tau)\}.$$

This metric is based on the intuition that a clock may constrain or measure the distance between any two events in a trajectory. For instance, $d_{allpair}(\tau_1,\tau_2) = 0.4$. □

Example 4. An alternate metric measures only the differences in the time-gaps between consecutive events:

$$d_{sucpair}(\tau,\tau') \;=\; \max\{|\delta_\tau(i) - \delta_{\tau'}(i)| : 1 \le i \le \mathrm{len}(\tau)\}.$$

For instance, $d_{sucpair}(\tau_1,\tau_2) = 0.2$. □

Example 5. The drift metric assumes that the clocks for measuring time-stamps may drift, and their deviation from a correct clock is a percentage of the total elapsed time. For example, if x is a correct clock, and x' is a clock with maximal drift 0.1, then always $x/1.1 \le x' \le 1.1x$. Thus we can define the distance between two trajectories as

$$d_{drift}(\tau,\tau') \;=\; \max\{\max\left(\frac{t_{\tau'}(i)}{t_\tau(i)}, \frac{t_\tau(i)}{t_{\tau'}(i)}\right) : 1 \le i \le \mathrm{len}(\tau)\} - 1.$$

It follows that $d_{drift}(\tau,\tau') = \epsilon$ iff $t_{\tau'}(i)/(1+\epsilon) \le t_\tau(i) \le (1+\epsilon)t_{\tau'}(i)$ for all $1 \le i \le \mathrm{len}(\tau)$. For instance, $d_{drift}(\tau_1,\tau_2) = 0.111\ldots$ Note that if all time-gaps in both τ and τ' were doubled, the distance $d_{drift}(\tau,\tau')$ would remain the same. This is not true of the other metrics. It is also possible to define sum, all-pairs, and successive-pairs versions of the drift metric. □

Example 6. Finally, we have the discrete metric, with $d_{disc}(\tau,\tau') = 1$ if $\tau \ne \tau'$. This metric with the tube acceptance conditions (defined below) would give us ordinary timed automata (with trajectory acceptance). We will not use this metric any further. □

Given a metric, we use the standard definition of open sets. The listed metrics, with the exception of the discrete metric, all define the same topology on trajectories. Formally, for a 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 $T_d(\tau,\epsilon) = \{\tau' : d(\tau,\tau') < \epsilon\}$ 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 $T_d(\tau,\epsilon) \subseteq O$. 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 $\mathbf{Tube}(d)$.

Proposition 1. *The five metrics $d = d_{max}, d_{sum}, d_{allpair}, d_{sucpair}, d_{drift}$ all define the same set* $\mathbf{Tube}(d)$ *of tubes.*

Proof. We show $\mathbf{Tube}(d_{max}) \subseteq \mathbf{Tube}(d_{sum}) \subseteq \mathbf{Tube}(d_{allpair}) \subseteq \mathbf{Tube}(d_{sucpair}) \subseteq \mathbf{Tube}(d_{max}) \subseteq \mathbf{Tube}(d_{drift}) \subseteq \mathbf{Tube}(d_{max})$.

Consider a tube $O \in \mathbf{Tube}(d_{max})$ and a trajectory $\tau \in O$. Then there is a positive real $\epsilon \in \mathbb{R}^+$ such that $T_{d_{max}}(\tau, \epsilon) \subseteq O$. Since $d_{sum}(\tau, \tau') \leq \mathrm{len}(\tau) \times d_{max}(\tau, \tau')$, it follows that $T_{d_{sum}}(\tau, \epsilon/\mathrm{len}(\tau)) \subseteq T_{d_{max}}(\tau, \epsilon) \subseteq O$. Hence $O \in \mathbf{Tube}(d_{sum})$.

The other proofs are similar. In each case we need to relate two metrics. The following relations suffice, and can be easily proved from the definitions:

$$
\begin{aligned}
d_{allpair}(\tau, \tau') &\leq d_{sum}(\tau, \tau') \\
d_{sucpair}(\tau, \tau') &\leq d_{allpair}(\tau, \tau') \\
d_{max}(\tau, \tau') &\leq \mathrm{len}(\tau) \times d_{sucpair}(\tau, \tau') \\
d_{drift}(\tau, \tau') &\leq d_{max}(\tau, \tau')/t_\tau(1) \\
d_{max}(\tau, \tau') &\leq d_{drift}(\tau, \tau') \times t_\tau(\mathrm{len}(\tau)) \ \square
\end{aligned}
$$

We say that a metric d on trajectories is *reasonable* if $\mathbf{Tube}(d)$ is equal to $\mathbf{Tube}(d_{max})$. Henceforth we assume to be given a reasonable metric d. A d-tube will be simply called a tube, and the set of tubes will be denoted $\mathbf{Tube}$.

From trajectory languages to tube languages

A *trajectory language* is any subset of $\mathbf{Traj}$; a *tube language* is any subset of $\mathbf{Tube}$. 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 $S^c = \mathbf{Traj} - S$ 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

$$[L] = \{O \in \mathbf{Tube} : O \subseteq \overline{L}\}.$$

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 timed automaton with trajectory set L to be the tube set $[L]$. This has the effect that a robust timed 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. Our definition of "sufficiently many" as "dense subset" does not seem all that strong, because every tube O, while uncountable, has dense subsets that are countable (such as the set of trajectories in O all of whose time-gaps are rationals). However, when we define timed automata below, we will see that the syntax of timed automata will not allow us to specify very strange trajectory languages L. In particular, we will not be able to specify a trajectory language L such that both L and L^c are dense in some tube O. Thus, for timed automata, a tube will be accepted iff all but finitely many of its trajectories are accepted, and it will be rejected iff all but finitely many of its trajectories are rejected.

3 Robust Timed Automata

We define a variant of Alur-Dill timed automata [AD94]. While the variant makes several aspects of our presentation easier, we will show that it is equivalent in expressive power to Alur-Dill timed automata.

A *timed automaton* is a 6-tuple $A = \langle \Sigma, Q, Q_0, Q_f, C, E \rangle$:

- Σ is a finite alphabet of events;
- Q is a finite set of locations;
- $Q_0 \subseteq Q$ is a set of start locations;
- $Q_f \subseteq Q$ is a set of accepting locations;
- C is a finite set of real-valued clock variables;
- $E \subseteq Q \times Q \times \Sigma \times \Phi(C) \times 2^C \times \Phi(C)$ is a finite set of transitions. Each transition $(q, q', a, \phi, \rho, \phi')$ consists of a source location q, a target location q', an event a, a constraint ϕ on the clocks in C, a set ρ of clock variables, and a constraint ϕ' on the clocks in ρ. The *clock constraints* $\Phi(C')$ on a set C' of clock variables are generated by the grammar
$$\phi \ ::= \ x \leq c \mid x \geq c \mid \neg\phi \mid \phi \wedge \phi,$$
where c is a rational number and $x \in C'$ is a clock variable. We call ϕ the *precondition*, ρ the *update set*, and ϕ' the *postcondition* of the transition. Intuitively, the transition is enabled when the clock values satisfy the precondition. When the transition is taken, the clock variables in the update set are assigned new values so that the postcondition is satisfied, and all other clock values remain unchanged.

Instead of precondition, update set, and postcondition, we often write guarded commands. For example, if x and y are clock variables, then the nondeterministic guarded command $x > 1 \rightarrow y := (0, 1)$ corresponds to the precondition $x > 1$, the update set $\{y\}$, and the postcondition $0 < y < 1$. It is often convenient to annotate locations with clock constraints, so-called *invariant conditions* [HNSY94]. Our results extend straight-forwardly to timed automata with invariant conditions.

A *clock-valuation function* $\gamma\colon C \rightarrow \mathbb{R}_0^+$ assigns to each clock variable a nonnegative real in $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$. The clock-valuation function γ satisfies the clock constraint ϕ iff ϕ evaluates to true when each clock x is replaced by the value $\gamma(x)$. For a positive real δ, the clock-valuation function $\gamma + \delta$ assigns to each clock x the value $\gamma(x) + \delta$.

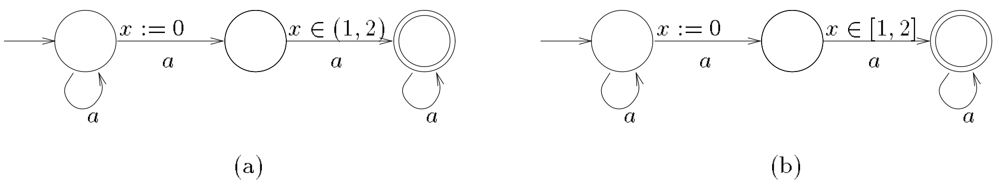

Fig. 1. The timed automata A_1 and A_2

Trajectory acceptance

A trajectory τ is *accepted* by the timed automaton A iff there is a sequence $r = \langle q_i, \gamma_i \rangle_{0 \le i \le \mathrm{len}(\tau)}$ of locations $q_i \in Q$ and clock-valuation functions $\gamma_i : C \to \mathbb{R}_0^+$ such that

1. Initialization: $q_0 \in Q_0$.
2. Consecution: for all $1 \le i \le \mathrm{len}(\tau)$, there is a transition in E of the form $(q_{i-1}, q_i, a_\tau(i), \phi_i, \rho_i, \phi'_i)$ such that $\gamma_{i-1} + \delta_\tau(i)$ satisfies ϕ_i, $\gamma_i(x) = \gamma_{i-1}(x) + \delta_\tau(i)$ for all $x \in C - \rho_i$, and γ_i satisfies ϕ'_i.
3. Acceptance: $q_k \in Q_f$ for $k = \mathrm{len}(\tau)$.

The sequence r is called a *run* of τ through the timed automaton A, and τ is said to be accepted along the *path* $\langle q_i \rangle_{0 \le i \le \mathrm{len}(\tau)}$ of locations. We write $L(A)$ for the set of trajectories accepted by A.

In an *Alur-Dill timed automaton*, if ρ is the update set of a transition, then the corresponding postcondition must have the form $\bigwedge_{x \in \rho} x = 0$; that is, the clock variables in the update set are always reset to 0.

Proposition 2. *For each timed automaton there is an Alur-Dill timed automaton that accepts the same set of trajectories.*

Proof. Every time a clock variable is updated and assigned, nondeterministically, a new value in the interval I_1, and later tested against the interval I_2, we replace the update with a reset to 0, and the test with a test against the interval $I_2 - I_1 = \{\delta_2 - \delta_1 : \delta_1 \in I_1, \delta_2 \in I_2\}$. If a clock variable is updated into different intervals on different transitions, these updates are handled using new clocks. □

Tube acceptance

The timed automaton A accepts the set $[L(A)]$ of tubes. That is, a tube O is accepted by A iff there is a set $O' \subseteq O$ of trajectories such that O' is dense in O and all trajectories in O' are accepted by A.

The following examples illustrate tube acceptance. First, consider the timed automaton A_1 of Figure 1(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

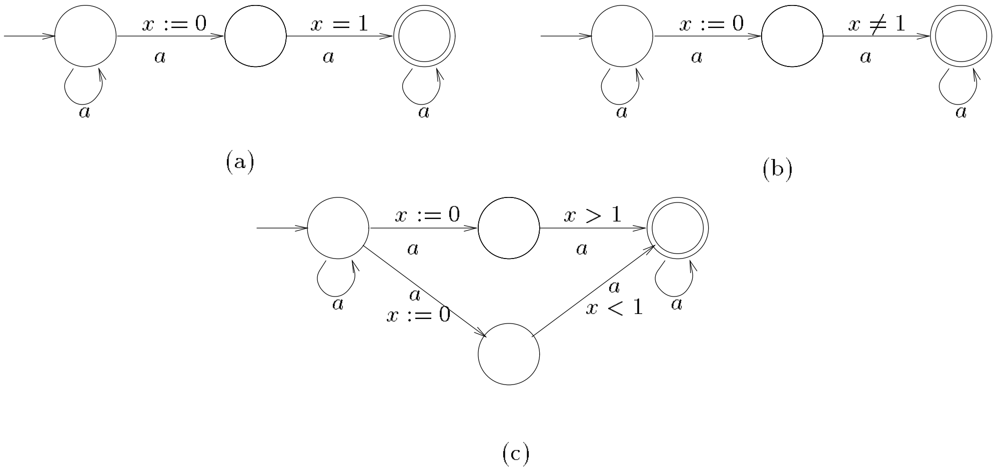

Fig. 2. The timed automata A_3, A_4, and A_5

trajectories in $L(A_1)$, and the maximal accepted tube is $L(A_1)$ itself. In the timed automaton A_2 of Figure 1(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.

Next, consider the timed automaton A_3 of Figure 2(a). This automaton accepts all trajectories of a's in which some a is followed by the subsequent a exactly 1 time unit later. The automaton A_3 accepts no tubes, because by perturbing the pair of a's that causes a trajectory to be accepted, however slightly, the trajectory becomes unacceptable. By contrast, the timed automaton A_4 of Figure 2(b) accepts all trajectories of a's such that there is a pair of consecutive a's that are not 1 time unit apart. Thus, by perturbing an unacceptable trajectory slightly, it will be accepted. As a result, the automaton A_4 accepts all tubes. The timed automaton A_5 of Figure 2(c) accepts the same trajectories as A_4, but along different paths of locations. Since our definition of tube acceptance depends only on the accepted trajectories, and not on the structure of the automaton, the automaton A_5 still accepts all tubes. Note, however, that some tubes, such as the tube of all trajectories, are not accepted along any single path through the automaton.

4 Verification of Robust Timed Automata

The basic verification problem for automata is the emptiness problem. In this section, we give an algorithm for checking if a timed automaton accepts any tube. For this purpose, we relate tube acceptance and trajectory acceptance by considering syntactic subclasses of timed automata.

Closed and open timed automata

A timed automaton A is *closed* iff all preconditions and postconditions of A are generated by the grammar

$$\phi \ ::= \ x \leq c \mid x \geq c \mid \phi \wedge \phi \mid \phi \vee \phi,$$

where c is a rational number and c is a clock variable. Dually, the timed automaton A is *open* iff all preconditions and postconditions of A are generated by the grammar

$$\phi \ ::= \ x < c \mid x > c \mid \phi \wedge \phi \mid \phi \vee \phi.$$

For each timed automaton A, we define the closure automaton and the interior automaton as follows. First, write every precondition and postcondition of A in negation-free form, using both logical-and and logical-or and both strict and nonstrict comparison operators. Then, the *closure automaton* $\overline{A}$ is the timed automaton that results from replacing each strict comparison operator by the corresponding nonstrict operator (replace $<$ by $\leq$, and replace $>$ by $\geq$). For example, $\overline{A_1} = A_2$ for the timed automata of Figure 1.

The interior automaton A^{int} cannot be formed by the reverse of the above process, as some of the postconditions would be replaced by "false." Instead, we relax the postconditions and tighten up the preconditions. For example, we obtain the interior automaton of the timed automaton A_2 of Figure 1(b) by replacing the constraint $x := 0$ by $x :\in (0, 0.5)$, and replacing the constraint $x \in [1, 2]$ by $(1.5, 2)$. This construction, however, is not sufficient if more than one test is performed. Consider the automaton shown on the left in Figure 3. Its interior is shown on the right in Figure 3. The two clocks x_1 and x_2 are necessary to capture the maximal and minimal possible values of the clock x. Then, any number of tests can be satisfied.

Formally, we assume that the clock constraints of the timed automaton A contain only integer constants. This can be ensured by multiplying all constants with their least common denominator, and once the interior automaton has been constructed, dividing all constants again by the same amount. For each clock variable x of A, the *interior automaton* A^{int} uses two clock variables, x_1 and x_2. In A^{int}, each clock constraint in a postcondition of A is replaced as follows:

$x \geq c$	$x_1 > c \wedge x_2 > c$
$x > c$	$x_1 > c \wedge x_2 > c$
$x \leq c$	$x_1 < c + 0.5 \wedge x_2 < c + 0.5$
$x < c$	$x_1 < c + 0.5 \wedge x_2 < c + 0.5$

In addition, each clock constraint in a precondition of A is replaced as follows:

$x \geq c$	$x_1 > c + 0.5 \vee x_2 > c + 0.5$
$x > c$	$x_1 > c + 0.5 \vee x_2 > c + 0.5$
$x \leq c$	$x_1 < c \vee x_2 < c$
$x < c$	$x_1 < c \vee x_2 < c$

Clearly, $\overline{A}$ is a closed automaton and A^{int} is open. Moreover, if A is closed, then $\overline{A} = A$, and if A is open, it is easy to check that $L(A^{int}) = L(A)$. The following proposition shows that as far as tube languages are concerned, we can either restrict our attention to closed automata or to open automata.

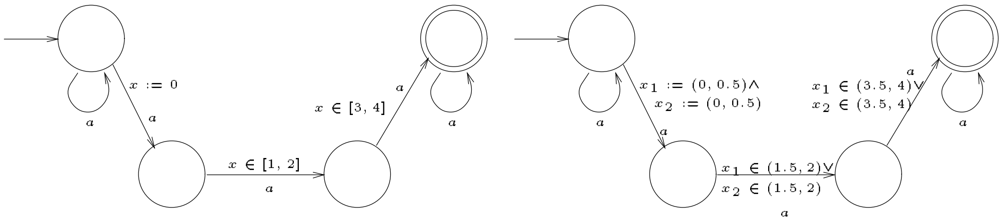

Fig. 3. A timed automaton and its interior automaton

Proposition 3. *For every timed automaton A, we have $L(\overline{A}) = \overline{L(A)}$ and $[L(A)] = [L(\overline{A})] = [L(A^{int})]$.*

Proof. Consider a trajectory $\tau \in L(\overline{A})$. By the construction of $\overline{A}$, in every ϵ-neighborhood of τ, there must exist a trajectory that is accepted by A along the same path. Hence $L(\overline{A}) \subseteq \overline{L(A)}$. Conversely, since $L(\overline{A})$ is closed, $\overline{L(A)} \subseteq L(\overline{A})$. Since $\overline{L(\overline{A})} = \overline{L(A)}$, we have $[L(\overline{A})] = [L(A)]$.

We now show that $L(A^{int}) \subseteq L(A)$, from which it follows that $[L(A^{int})] \subseteq [L(A)]$. By the construction of A^{int}, each postcondition $x \in (a, b)$ is changed to $x_1 \in (a, b + 0.5) \wedge x_2 \in (a, b+0.5)$, and each precondition $x \in (c, d)$ is changed to $x_1 \in (c+0.5, d) \vee x_2 \in (c+0.5, d)$. Thus the amount of time that may elapse between the setting and the testing of any of the clocks x, x_1, and x_2 is characterized by the interval $(c - b, d - a)$.

Conversely, consider a tube $O \in [L(A)]$ and a trajectory $\tau \in O$. Then there exists a positive real $\epsilon > 0$ such that $T(\tau, \epsilon) \subseteq O$. Since $T(\tau, \epsilon) \subseteq \overline{L(A)}$, there exists a trajectory $\tau' \in T(\tau, \epsilon)$ such that τ' is accepted strictly by A; that is, τ' is accepted along a path of A such that none of the nonstrict preconditions are satisfied at the boundary. This is because if $L_N(A)$ is the set of trajectories in $L(A)$ that are not accepted strictly by A, then the interior of $\overline{L_N(A)}$ is empty. Now τ' is accepted along the same path in A^{int}: when the clock x is set to t, let $x_1 = t$ and let $x_2 = t + 1/2$. Thus we can construct a sequence of trajectories accepted by A^{int} with limit τ, for each $\tau \in O$. Hence $O \in [L(A^{int})]$. □

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

Proposition 4. *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.*

Proof. It suffices to consider the metric d_{max}, because any d_{max}-tube contains a d-tube for every reasonable metric d. Consider a run $r = \langle q_i, \gamma_i \rangle_{0 \leq i \leq \mathrm{len}(\tau)}$ of A that accepts the trajectory τ. Since all clock constraints are open, for each $0 \leq i \leq \mathrm{len}(\tau)$, there is a real $\epsilon_i > 0$ such that substituting $\gamma_i + \epsilon_i$ or $\gamma_i - \epsilon_i$ for γ_i in r still gives a run through A. Now let $\epsilon = \min\{\epsilon_i : 0 \leq i \leq \mathrm{len}(\tau)\}$. □

From the proof it follows that in the case of $d = d_{max}$, if a trajectory τ is accepted by an open timed automaton A whose clock-constraint constants are all integers, then τ

belogs to a d-tube of diameter $\epsilon = 1/2$ which is accepted by A. It should also be noticed that for every closed timed automaton A, we have $\bigcup[L(A)] \subseteq L(A)$, and for every open timed automaton B, we have $L(B) \subseteq \bigcup[L(B)]$. The latter follows from Proposition 4.

Checking emptiness

By Proposition 3 we can reduce the problem of checking if a timed automaton A accepts any tube to the problem of checking if the interior automaton A^{int} accepts any tube. Moreover, by Proposition 4, the open automaton A^{int} accepts any tube iff it accepts any trajectory. The latter problem can be solved using the region construction of [AD94]. In fact, for checking the emptiness of open timed automata such as A^{int}, only open regions need be considered.

Theorem 5. *The problem of deciding whether a timed automaton accepts any tube is complete for PSPACE.*

Proof. Given an open timed automaton A, we construct an open-region automaton $\text{reg}(A)$, which is a finite-state machine that accepts a string s iff A accepts a trajectory τ with $\text{untime}(\tau) = s$. First, we multiply all clock-constraint constants in A with their least common denominator, so that all resulting constants are integers. Let $c_{\max}$ be the largest of these integers. An *open clock region* is a satisfiable conjunction of formulas that contains

- for each clock x of A, either the conjunct $x > c_{\max}$ or a conjunct of the form $c < x < c+1$, for an integer $0 \leq c \leq c_{\max}$, and
- for each pair of clocks x and y of A, either the conjunct $x - \lfloor x \rfloor < y - \lfloor y \rfloor$ or the conjunct $y - \lfloor y \rfloor < x - \lfloor x \rfloor$.

For two open clock regions R and R', the region R' is a *successor region* of R iff there is a clock-valuation function γ and a real $\delta \in \mathbb{R}^+$ such that R satisfies γ and R' satisfies $\gamma + \delta$.

The input alphabet Σ of the finite-state machine $\text{reg}(A)$ is the same as for A. Each state of $\text{reg}(A)$ is a pair $\langle q, R\rangle$ that consists of a location q of A and an open clock region R. The state $\langle q, R\rangle$ is a start state of $\text{reg}(A)$ iff q is a start location of A, and $\langle q, R\rangle$ is an accepting state of $\text{reg}(A)$ iff q is an accepting location of A. For each $a \in \Sigma$, there is an a-transition from the state $\langle q, R\rangle$ to the state $\langle q', R'\rangle$ in $\text{reg}(A)$ iff A has a transition of the form $(q, q', a, \phi, \rho, \phi')$ such that R implies ϕ and R' implies both ϕ' and ϕ'', which results from ϕ by replacing with "true" all comparisons that involve clocks in ρ. In addition, there is an ε-transition from the state $\langle q, R\rangle$ to the state $\langle q', R'\rangle$ in $\text{reg}(A)$ iff $q = q'$ and R' is a successor region of R.

PSPACE-completeness follows from the corresponding proof in [AD94]. □

5 Nondeterminizability of Robust Timed Automata

The previous section shows that timed automata yield a decidable theory of tubes. In this section, we present evidence that the resulting theory of tubes is not closed under all boolean operations. Using trajectory-based methods, for every two timed automata A and B, we can construct a product automaton C with $[L(C)] = [L(A)] \cap [L(B)]$[4]

[4] From the results of this section it will follow that $[L(A)] \cap [L(B)] = [L(A) \cap L(B)]$.

and a union automaton D with $[L(D)] = ([L(A)] \cup [L(B)])^* = [L(A) \cup L(B)]$, where $\mathcal{L}^*$ denotes the closure of a tube language $\mathcal{L}$ under union (i.e., $\mathcal{L}^*$ is the least set of tubes containing $\mathcal{L}$ which is closed under union). As in ordinary timed automata, however, complementation presents a problem.

Complementation

The timed automaton B is a *trajectory complement* of the timed automaton A iff B accepts precisely the trajectories that are not accepted by A; that is, $L(B) = L(A)^c$. Before defining the tube complements of a timed automaton, we observe an important property of the trajectory languages that can be defined by timed automata.

Proposition 6. *For every timed automaton A, there is no tube O such that both $L(A)$ and $L(A)^c$ are both dense in O.*

Proof. Suppose that $L(A)$ is dense in O. Then $O \subseteq \overline{L(A)}$, and $O \in [L(A)] = [L(A^{int})]$. By Proposition 4, for each trajectory $\tau \in O$ there is a positive real $\epsilon \in \mathbb{R}^+$ such that $T(\tau, \epsilon) \subseteq L(A^{int}) \subseteq L(A)$. Hence $L(A)^c$ is not dense in O. □

It follows that a tube cannot be accepted by both a timed automaton A and a trajectory complement of A. This observation will allow us to relate the tube complements of a timed automaton to its trajectory complements.

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

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

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

Proof. Let O be a tube in $[L]$, and let τ be a trajectory in O. Then there is an $\epsilon > 0$ and a tube around τ of diameter ϵ whose trajectories are all in O. Suppose that $\tau \in O'$ for some tube $O' \in [L^c]$. Then there is an $\epsilon' > 0$ and a tube around τ of diameter ϵ' whose trajectories are all in O'. Without loss of generality, assume that $\epsilon \leq \epsilon'$. Thus the trajectories of the ϵ-tube around τ are contained in both $\overline{L}$ and $\overline{L^c}$. This contradicts the fact that there is no tube in which both L and L^c are dense. Hence the tubes in $[L^c]$ are pairwise disjoint from the tubes in $[L]$. □

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 6 and 7, it follows that every trajectory complement of a timed automaton is also a tube complement (the converse is generally not true). Since

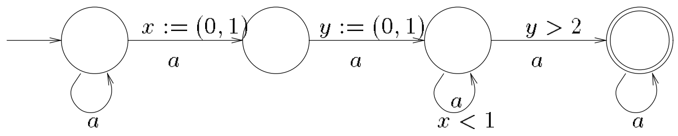

Fig. 4. A nondeterminizable open timed automaton

$[L(A)]^c = [L(A^{int})]^c = [L(A^{int})^c]$, in order to construct tube complements, it would suffice to construct trajectory complements of open timed automata.[5] This, however, does not seem feasible, because we now show that open timed automata cannot be determinized, which is the usual first step in automaton complementation.

Nondeterminizability of open timed automata

A timed automaton A is *tube-determinizable* iff there is a deterministic Alur-Dill timed automaton that accepts the set $[L(A)]$ of tubes. An Alur-Dill timed automaton is *deterministic* iff for all locations, every two outgoing transitions contain either different events or mutually exclusive preconditions. Note that trajectory-determinizability implies tube-determinizability, but not vice versa.

Theorem 8. *The open timed automaton A of Figure 4 is not tube-determinizable.*

Proof. The automaton A accepts a trajectory over the unary alphabet $\{a\}$ iff there is some consecutive pair of a's with time-stamps t and t' such that there are no a's with time-stamps in the interval $[t+1, t'+1]$. Every such trajectory is accepted robustly, as part of an accepted tube; that is, $L(A) = \bigcup[L(A)]$. To accept any tube in $[L(A)]$ with sufficiently small diameter deterministically, an automaton would have to remember the time-stamps of all a's within the last 1 time unit, which is not possible with a finite number of clock variables.

Formally, suppose there is a deterministic Alur-Dill timed automaton B with n clock variables and $[L(B)] = [L(A)]$. For simplicity, assume that all clock-constraint constants of B are integers, and assume the metric $d_{sucpair}$ on trajectories. For $\delta_0 = 0$, consider a trajectory τ of the form $(a, \delta_1) \ldots (a, \delta_{n+2})(a, (\delta_0+\delta_1)/2) \ldots (a, (\delta_{n+1}+\delta_{n+2})/2)$ with $t_\tau(n+2) = 1$. The trajectory τ is rejected robustly by A, as part of a rejected tube. Hence we can choose a positive real $\epsilon \in \mathbb{R}^+$ such that $O = T(\tau, \epsilon) \in [L(A)]^c$ and $\epsilon < \delta_i/8$ for all $1 \leq i \leq n+2$. Then $O \in [L(B)]^c = [L(B)^c]$. Since B is deterministic and has at most n clock variables, for each trajectory $\tau' \in O \cap L(B)^c$, there is at most one run of B over τ'. After reading the first $n+2$ events of this run, there is at least one $1 \leq i \leq n+1$ such that no clock variable of B has a value in the interval $(1 - t_\tau(i) - \epsilon, 1 - t_\tau(i) + \epsilon)$. We partition the trajectories in $O \cap L(B)^c$ into $n+1$ sets, corresponding to the possible values for i. At least one of these sets must be dense in $O \cap L(B)^c$. Let this be the k-th set. Now consider the set O' of trajectories obtained from $O \cap L(B)^c$ by reducing the time-gap of each $(n+k+3)$-rd event by $\delta_{k+1}/2 + \delta_k/4$,

[5] 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].

increasing the time-gap of each $(n+k+4)$-th event by the same amount, and truncating each sequence after $n+k+4$ events. All trajectories in O' are rejected by B, because they follow the same paths as the corresponding trajectories in O, which, if truncated after $n+k+4$ events, are also rejected. But all trajectories in O' are accepted by A. Since O' is dense in some tube, $[L(B)] \neq [L(A)]$. $\square$

We suspect that the open timed automaton A has no tube complement. For, a tube complement of A would have to accept all trajectories of a's such that every consecutive pair of a's with time-stamps t and t' is followed by another a with a time-stamp in the interval $(t+1, t'+1)$. For this purpose, the automaton would have to remember the time-stamps of an unbounded number of a's, which does not seem possible (however, we know of no formal proof, as the above proof depends on the determinism of B).

6 Robust Hybrid Automata

The definitions of tube acceptance can be extended to hybrid automata. If **HTraj** is the set of hybrid trajectories, then each hybrid automaton accepts a subset of **HTraj** [ACH+95]. Given a metric on **HTraj**, we again define tubes as the open sets of the corresponding topology. Now, following our definition for timed automata, a tube is accepted by a hybrid automaton iff a dense subset of trajectories in the tube are accepted by the automaton. Several metrics on hybrid trajectories can be defined similar to the corresponding metrics for timed trajectories (Section 2). Here we propose three additional metrics.

A *hybrid trajectory* σ over a given set of real-valued variables V is a piecewise smooth function $\sigma : I \to (V \to \mathbb{R})$ from a bounded interval $I \subset \mathbb{R}_0^+$ of the nonnegative real line to valuation functions for the variables in V. By piecewise smooth we mean that the domain of σ can be partitioned into a finite sequence $I = I_1 \cup \ldots \cup I_m$ of intervals such that for each $1 \leq j \leq m$ and each variable $x \in V$, the real-valued function $\sigma(x)$ restricted to the domain I_j is infinitely differentiable (i.e., $(\sigma \upharpoonright I_j)(x) \in \mathbf{C}^\infty$).

Suppose that $V = \{x_1, \ldots, x_n\}$. Each valuation function for V is a point in $\mathbb{R}^n$. For two points p and p' in $\mathbb{R}^n$, let $d_{euc}(p,p')$ be the euclidean distance between p and p':

$$d_{euc}(p,p') \;=\; \sqrt{(p_1-p_1')^2+\cdots+(p_n-p_n')^2}.$$

The timed metric on hybrid trajectories compares the values of all variables at each point in time: if two hybrid trajectories σ and σ' have different domains, $d_{time}(\sigma,\sigma') = \infty$; otherwise, if $\mathrm{dom}(\sigma)$ is the domain of both σ and σ', then

$$d_{time}(\sigma,\sigma') \;=\; \sup\{d_{euc}(\sigma(t),\sigma(t)) : t \in \mathrm{dom}(\sigma)\}.$$

Alternatively, each hybrid trajectory σ can be regarded as a subset of the $(n+1)$-dimensional real space $\mathbb{R}^{n+1}$, with one component representing time, and the other components representing values for the variables in V: let $(p_0, p_1, \ldots, p_n) \in \sigma$ iff $p_0 \in \mathrm{dom}(\sigma)$ and $\sigma(p_0)(x_i) = p_i$ for all $1 \leq i \leq n$. Then the distance of a point $p \in \mathbb{R}^{n+1}$ from a hybrid trajectory $\sigma' \subset \mathbb{R}^{n+1}$ can be defined using the euclidean metric on $\mathbb{R}^{n+1}$:

$$d_{euc}(p,\sigma') \;=\; \inf\{d_{euc}(p,p') : p' \in \sigma'\}.$$

Now the distance between two hybrid trajectories σ and σ' can be defined as

$$d_{euc}(\sigma,\sigma') \;=\; \max(\sup\{d(p,\sigma') : p \in \sigma\}, \sup\{d(p',\sigma) : p' \in \sigma'\}).$$

While the metric d_{euc} treats time as a data variable, one can also project away the time component and look at hybrid trajectories as subsets of the phase space $\mathbb{R}^n$. This gives us the metric d_{phase}.

Consider, for example, the two functions f_1 and f_2 with $f_1(t)(x) = t$ and $f_1(t)(x) = t+2$ for all $t \in \mathbb{R}_0^+$. For the two hybrid trajectories $\sigma_1 = (f_1 \upharpoonright [0,5])$ and $\sigma_2 = (f_2 \upharpoonright [0,5])$, we have $d_{time}(\sigma_1, \sigma_2) = 2$. For the two hybrid trajectories $\sigma_1 = (f_1 \upharpoonright [1,6])$ and $\sigma_2 = (f_2 \upharpoonright [0,5])$, we have $d_{euc}(\sigma_1, \sigma_2) = \sqrt{2}$. For the two hybrid trajectories $\sigma_1 = (f_1 \upharpoonright [2,7])$ and $\sigma_2 = (f_2 \upharpoonright [0,5])$, we have $d_{phase}(\sigma_1, \sigma_2) = 0$ and, for a hybrid extension of the metric d_{max} from Section 2, $d_{\sup}(\sigma_1, \sigma_2) = 1$.

Linear hybrid automata can be analyzed for tube acceptance as in the case of trajectory acceptance [AHH96], but only open regions (open polyhedral sets in $\mathbb{R}^n$) are needed during the computation. This significantly simplifies the algorithms that have been implemented in tools such as HyTech [HHWT95]. We conclude by posing an important open question: are there interesting classes of hybrid automata whose emptiness is undecidable under trajectory acceptance but decidable under tube acceptance?

References

[ACD93] R. Alur, C. Courcoubetis, and D.L. Dill. Model checking in dense real time. *Information and Computation*, 104(1):2–34, 1993.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[ACHH93] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, Lecture Notes in Computer Science 736, pages 209–229. Springer-Verlag, 1993.

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

[AFH94] R. Alur, L. Fix, and T.A. Henzinger. A determinizable class of timed automata. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 1–13. Springer-Verlag, 1994.

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

[AHH96] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[Hen96] T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996. Invited tutorial.

[HHWT95] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: the next generation. In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 56–65. IEEE Computer Society Press, 1995.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994. Special issue for LICS 92.