What's Decidable about Hybrid Automata?[1]

Thomas A. Henzinger[2] Peter W. Kopke[2] Anuj Puri[3] Pravin Varaiya[3]

Abstract. Hybrid automata model systems with both digital and analog components, such as embedded control programs. Many verification tasks for such programs can be expressed as reachability problems for hybrid automata. By improving on previous decidability and undecidability results, we identify the precise boundary between decidability and undecidability of the reachability problem for hybrid automata.

On the positive side, we give an (optimal) PSPACE reachability algorithm for the case of initialized rectangular automata, where all analog variables follow trajectories within piecewise-linear envelopes and are reinitialized whenever the envelope changes. Our algorithm is based on a translation of an initialized rectangular automaton into a timed automaton that defines the same timed language. The translation has practical significance for verification, because it guarantees the termination of symbolic procedures for the reachability analysis of initialized rectangular automata.

On the negative side, we show that several slight generalizations of initialized rectangular automata lead to an undecidable reachability problem. In particular, we prove that the reachability problem is undecidable for timed automata with a single stopwatch.

1 Introduction

A hybrid automaton [ACHH93] combines the discrete dynamics of a finite automaton with the continuous dynamics of a dynamical system. Hybrid automata thus provide a mathematical model for digital computer systems that interact with an analog environment in real time. Case studies indicate that the model of hybrid automata is useful for the analysis of embedded software and hardware, including distributed processes with drifting clocks, real-time schedulers, and protocols for the control of manufacturing plants, vehicles, and robots [AHH93, NOSY93, BPV94, HH94, HRP94, MV94, PV94b, ACH+94, MPS95]. Two problems that are central to the analysis of hybrid automata are the reachability problem and the more general timed language-inclusion problem[4]. These problems check if the trajectories of a given hybrid automaton meet a given safety or real-time requirement. While a scattering of previous results show that both problems are decidable in certain special cases, and undecidable in certain general cases, this paper provides a systematic identification of the boundary between decidability and undecidability.

Hybrid automata generalize timed automata. Timed automata [AD94] equip finite automata with clocks, which are real-valued variables that follow continuous trajectories with constant slope 1. Hybrid automata equip finite automata with real-valued variables whose trajectories follow more general dynamical laws. For each class of dynamical laws, we obtain a class of hybrid automata. A particularly interesting class of dynamical laws confines the set of possible trajectories to piecewise-linear envelopes. Suppose, for example, that the variable x represents the water level in a tank. Depending on the position of a control valve (i.e., the state of a finite control automaton), the water level either falls nondeterministically at any rate between 2 and 4 cm s^{-1}, or rises at any rate between 1 and 3 cm s^{-1}. We model these two situations by the dynamical laws $\dot{x} \in [-4,-2]$ and $\dot{x} \in [1,3]$—so-called rectangular activities [AHH93, PV94a]—which enforce piecewise-linear envelopes on the water-level trajectories. Rectangular-activity automata are interesting from a practical point of view, as they permit the modeling of clocks with bounded drift and the conservative approximation of arbitrary trajectory sets [OSY94, PV94b, HH95], and from a theoretical point of view, as they lie at the boundary of decidability.

Our results are twofold. First, we give an (optimal) PSPACE algorithm for the timed language-inclusion problem for rectangular-activity automata with two restrictions: (1) whenever the activity of a

[1]This research was supported in part by the National Science Foundation under grant CCR-9200794, by the Defense Advanced Research Projects Agency under grant NAG2-892, by the U.S. Air Force Office of Scientific Research under contract F49620-93-1-0056, by the U.S. Army Research Office under contracts DAAL03-91-C-0027 and DAAH04-94-G-0026, and by the California PATH program.

[2]Department of Computer Science, Cornell University, Ithaca, NY 14853.

[3]Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720.

STOC '95, Las Vegas, Nevada, USA
© 1995 ACM 0-89791-718-9/95/0005..$3.50

[4]In the case of timed language inclusion, we assume we are given the complement of the requirement.

variable changes, the value of the variable is reinitialized; (2) the values of two variables with different activities are never compared. Second, we prove that the reachability problem becomes undecidable if either restriction is relaxed, or if more general, triangular activities are admitted. To explain the significance of both results, they must be placed in the context of previous work.

Decidability [Section 3]. The first decidability result for hybrid automata was obtained for timed automata, whose reachability and timed language-inclusion problems are PSPACE-complete [AD94]. Under restrictions (1) and (2), that result was later generalized to multirate automata, with variables that run at any constant positive slopes [ACHH93, NOSY93], and to the reachability problem for rectangular-activity automata [PV94a]. The latter result, however, fell short in several respects, because its proof was based on the discretization of time. First, the result applied only to automata with closed rectangular activities. Second, the result applied only to the reachability problem and not to the timed-language inclusion problem. Third, the result did not suggest a practical verification algorithm. Fourth, and perhaps most importantly, the result did not imply the termination of existing semi-decision procedures for the reachability problem of hybrid automata, such as those implemented in the HyTech verification tool [AHH93, ACH+94]. Our main decidability result remedies all four of these shortcomings, because its proof is based on a translation of rectangular-activity automata via multirate automata into timed automata. As a corollary, we obtain a PSPACE algorithm for the timed language-inclusion problem of rectangular-activity automata under restrictions (1) and (2).

Undecidability [Section 4]. Over the past few years, there have been several ad hoc undecidability results about hybrid automata. A constant-slope variable with slope other than 0 or 1 is called a skewed clock, and a two-slope variable with slopes 0 and 1 is a stopwatch. In [ACHH93], it is shown that reachability is undecidable for timed automata with two skewed clocks. In [KPSY93], it is shown that reachability is undecidable for timed automata with two three-slope variables and restriction (2). In [Cer92], it is shown that reachability is undecidable for timed automata with three stopwatches and restriction (2). In [ACH93, BES93, KPSY93, BER94], it is shown that, under various strong side conditions, reachability is decidable for timed automata with one stopwatch, but the general problem is left open. We strengthen these results and give a uniform characterization of the undecidability frontier. First, we prove that any relaxation of restriction (1) leads to the undecidability of the reachability problem for timed automata with a single two-slope variable, such as a stopwatch. Second, we prove that any relaxation of restriction (2) leads to the undecidability of the reachability problem for timed automata with a single skewed clock. As a corollary, we obtain the undecidability of the reachability problem for triangular-activity automata, even under restrictions (1) and (2).

Other related work. In [OSY94], rectangular-activity automata are translated into more abstract timed automata. Our translation, by contrast, preserves timed languages and therefore leads to exact verification and decidability results. In [MP93], decidability and undecidability results are obtained for a deterministic model of hybrid systems with strong side conditions on the discrete dynamics. The hybrid automaton model, by contrast, is nondeterministic and its discrete dynamics is unconstrained. Finally, our results do not cover the case considered in [AHV93], where reachability is shown to be undecidable for timed automata with slope 0 variables.

2 Rectangular Automata

A hybrid automaton of dimension n is an infinite-state machine whose state has a discrete part, which ranges over the vertices of a graph, and a continuous part, which ranges over the n-dimensional euclidean space $\mathbb{R}^n$ [ACHH93]. A run of a hybrid automaton is a sequence of transition steps and time steps. During a transition step, the discrete and continuous states are updated according to a guarded command. During a time step, the discrete state remains unchanged, and the continuous state changes according to a dynamical law, say, a differential equation. In this paper, we are concerned with decidability questions about hybrid automata, and therefore consider restricted classes of guarded commands and dynamical laws. This leads us to the definition of rectangular automata.

Rectangular regions. Given a positive integer n, a subset of $\mathbb{R}^n$ is called a *region*. A closed and bounded region is *compact*. A *rectangular region* $B \subset \mathbb{R}^n$ is a cartesian product of (possibly unbounded) intervals on $\mathbb{R}$, all of whose endpoints are rational. We write B_i for the projection of B on the ith coordinate, so that $B = \prod_{i=1}^{n} B_i$. The rectangular region B is *bounded* if all of the intervals B_i are bounded. The set of all rectangular regions in $\mathbb{R}^n$ is denoted $\mathcal{B}_n$, and the set of all bounded rectangular regions is denoted $\mathcal{B}_n^{bd}$.

Definition of rectangular automata. An *n-dimensional rectangular automaton* A consists of a

directed multigraph (V_A, E_A), a finite *observation alphabet* Σ_A, two vertex labeling functions $inv_A: V_A \to \mathcal{B}_n^{bd}$ and $act_A : V_A \to \mathcal{B}_n$, and four edge labeling functions $pre_A : E_A \to \mathcal{B}_n$, $post_A : E_A \to \mathcal{B}_n$, $upd_A : E_A \to 2^{\{1,\ldots,n\}}$, and $obs_A : E_A \to \Sigma_A^\tau$, where $\Sigma_A^\tau = \Sigma_A \cup \{\tau\}$ augments Σ_A with the *null observation* $\tau \notin \Sigma_A$. We suppress the subscript A if it is clear from context.

The *preguard function pre*, the *postguard function post*, and the *update function upd* constrain the behavior of the automaton state during transition steps. An edge $e = (v, w)$ may be traversed only if the discrete state resides at v and the continuous state lies in $pre(e)$. For each $i \notin upd(e)$, the ith coordinate of the continuous state is not changed and must lie in $post(e)_i$. For each $i \in upd(e)$, the ith coordinate of the continuous state is nondeterministically assigned a new value in $post(e)_i$. The *observation function obs* identifies every edge traversal with an observation from Σ^τ. The *invariant function inv* and the *activity function act* constrain the behavior of the automaton state during time steps. While the discrete state resides at vertex v, the continuous state nondeterministically follows a smooth (C^∞) trajectory within $inv(v)$, and its first derivative remains within $act(v)$.

The rectangular automaton A is *initialized* if for every edge $e = (v, w)$, and for all i with $act(v)_i \neq act(w)_i$, we have $i \in upd(e)$. It follows that whenever the ith continuous coordinate of an initialized automaton changes its dynamics, as given by the activity function, then its value is nondeterministically reinitialized according to the postguard function. The rectangular automaton A is *global* if the activity function of A is a constant function on the set of vertices; that is, for all $v, w \in V$, $act(v) = act(w)$. Every global rectangular automaton is initialized.

The labeled transition system of a rectangular automaton. The rectangular automaton A defines a labeled transition system on an infinite state space. A *state* $(v, \mathbf{x})$ of A consists of a discrete part $v \in V$ and a continuous part $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x} \in inv(v)$. The *state space* $Q_A \subset V \times \mathbb{R}^n$ of A is the set of all states of A. A subset of Q_A is called a *zone*. Each zone $R \subset Q_A$ can be uniquely decomposed into a collection $\bigcup_{v \in V} \{v\} \times R_v$ of regions R_v, one for each vertex v. The zone R is *rectangular* if each region R_v is rectangular.

For each observation $\sigma \in \Sigma^\tau$, we define the *transition-step relation* $\overset{\sigma}{\Rightarrow}$ on Q_A by $(v, \mathbf{x}) \overset{\sigma}{\Rightarrow} (w, \mathbf{y})$ iff there is an edge $e = (v, w)$ in E such that $obs(e) = \sigma$, $\mathbf{x} \in pre(e)$, $\mathbf{y} \in post(e)$, and for each $i \notin upd(e)$, $x_i = y_i$. Hence $\mathbf{x}$ and $\mathbf{y}$ differ only at coordinates in the update set $upd(e)$. The *stutter closure* $\overset{\sigma}{\to} \subset Q_A^2$ of the transition-step relation $\overset{\sigma}{\Rightarrow}$ is defined as $\overset{\tau}{\Rightarrow}{}^* \circ \overset{\sigma}{\Rightarrow} \circ \overset{\tau}{\Rightarrow}{}^*$ if $\sigma \neq \tau$, and $\overset{\tau}{\to} = \overset{\tau}{\Rightarrow}{}^*$, where $\Rightarrow^*$ is the reflexive-transitive closure of $\Rightarrow$.

For each nonnegative real $t \in \mathbb{R}_{\geq 0}$, we define the *time-step relation* $\overset{t}{\Rightarrow}$ on Q_A by $(v, \mathbf{x}) \overset{t}{\Rightarrow} (w, \mathbf{y})$ iff (1) $v = w$ and (2) either $t = 0$ and $\mathbf{x} = \mathbf{y}$, or $t > 0$ and $\frac{\mathbf{y}-\mathbf{x}}{t} \in act(v)$. Notice that due to the convexity of rectangular regions, $(v, \mathbf{x}) \overset{t}{\Rightarrow} (w, \mathbf{y})$ iff there is a smooth function $f : [0, t] \to inv(v)$ with $f(0) = \mathbf{x}$, $f(t) = \mathbf{y}$, and for all $s \in (0, t)$, $\dot{f}(s) \in act(v)$. Hence the continuous state may evolve from $\mathbf{x}$ to $\mathbf{y}$ via any smooth trajectory satisfying the constraints imposed by $inv(v)$ and $act(v)$. The *stutter closure* $\overset{t}{\to} \subset Q_A^2$ of the time-step relation $\overset{t}{\Rightarrow}$ is defined as follows: $q \overset{t}{\to} q'$ iff there are nonnegative reals $t_1, \ldots, t_m$ and states $q_1, \ldots, q_{2m-2}$ such that $q \overset{t_1}{\Rightarrow} q_1 \overset{\tau}{\Rightarrow} q_2 \overset{t_2}{\Rightarrow} q_3 \overset{\tau}{\Rightarrow} \cdots \overset{\tau}{\Rightarrow} q_{2m-2} \overset{t_m}{\Rightarrow} q'$ and $t = \sum_{i=1}^m t_i$.

The *transition relation* $\Rightarrow_A \subset Q_A^2$ of A is the union of all transition-step and time-step relations: $\Rightarrow = \bigcup\{\overset{\sigma}{\Rightarrow} \mid \sigma \in \Sigma^\tau\} \cup \bigcup\{\overset{t}{\Rightarrow} \mid t \geq 0\}$. Given a zone $R \subset Q_A$, let $Post_A(R) = \{q \in Q_A \mid \exists r \in R.\, r \Rightarrow_A q\}$ be the zone of states that are reachable in one step from R, and let $Pre_A(R') = \{q \in Q_A \mid \exists r \in R'.\, q \Rightarrow_A r\}$ be the zone of states from which R' is reachable in one step. We define $Post_A^*(R) = \bigcup_{i \in \mathbb{N}} Post_A^i(R)$ and $Pre_A^*(R) = \bigcup_{i \in \mathbb{N}} Pre_A^i(R)$. Then $Post_A^*(R)$ is the zone of all states reachable from R. For two zones $R, R' \subset Q_A$, define $R \Rightarrow_A^* R'$ iff $Post_A^*(R) \cap R' \neq \emptyset$.

The language of a rectangular automaton. Let A be a rectangular automaton, and let $R \subset Q_A$ be a zone. A *timed word* for A is a finite or infinite sequence over the alphabet $\mathbb{R}_{\geq 0} \times \Sigma_A$ of time-stamped observations. An *R-run* ρ of A is a finite sequence of the form $q_0 \overset{t_1}{\to} q_1 \overset{\sigma_1}{\to} q_2 \overset{t_2}{\to} q_3 \overset{\sigma_3}{\to} \cdots, \overset{\sigma_m}{\to} q_{2m}$, where $q_0 \in R$, and for all i, $q_i \in Q_A$, $t_i \in \mathbb{R}_{\geq 0}$, and $\sigma_i \in \Sigma_A$. The run ρ *accepts* the timed word $(t_1, \sigma_1), (t_2, \sigma_2), (t_3, \sigma_3), \ldots\ldots, (t_m, \sigma_m)$. The *timed R-language* of A, denoted $Lang_A(R)$, is the set of all timed words that are accepted by R-runs of A.

Example. In examples, we refer to a coordinate of the continuous state as a *variable*, and we name variables $a, b, c, \ldots$ instead of $x_1, x_2, x_3, \ldots$ If the variable a corresponds to the ith coordinate, we write $act(v)(a)$ for $act(v)_i$, etc. In figures, we annotate each vertex with its activity function. For example, if $act(v)(a) = [3, 5]$ and $act(v)(b) = [4, 4]$, we write "$\dot{a} \in [3, 5]$" and "$\dot{b} = 4$" inside of v. Edges are annotated with observations and guarded commands. A guarded command ψ defines regions $pre(\psi)$ and $post(\psi)$, and an update set $upd(\psi)$, in a natural manner. For example, if ψ is the guarded command

$$a \leq 5 \wedge b = 4 \;\to\; b := 7;\; c :\in [0, 5],$$

then $pre(\psi)(a) = (-\infty, 5]$, $pre(\psi)(b) = [4, 4]$, $pre(\psi)(c) = (-\infty, \infty)$, $upd(\psi) = \{b, c\}$, $post(\psi)(a) = (-\infty, 5]$, $post(\psi)(b) = [7, 7]$, and $post(\psi)(c) = [0, 5]$.

Consider, for instance, the 2D rectangular automaton $\hat{A}$ of Figure 1. The observation alphabet of $\hat{A}$ is $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, and the invariant function of $\hat{A}$ is the constant function $\lambda v.\, [-20, 20]^2$ (not shown in the figure). The automaton $\hat{A}$ is initialized, as the values of the two variables c and d are reinitialized whenever their activities change. Figure 4 shows a sample trajectory of $\hat{A}$ from the zone $\hat{R} = \{(v_1, (0, 1))\}$. Each arc is labeled with a vertex giving the discrete state, while the continuous state follows the arc. The discontinuities between the arcs labeled v_2 and v_3 correspond to the update of variable d from -5 to -4 upon traversal of the edge from v_2 to v_3. A timed word $\hat{R}$-accepted by the automaton $\hat{A}$ is $((4, \sigma_1), (1, \sigma_2), (1, \sigma_3), (1, \sigma_4))$, with the corresponding state sequence

$$\big((v_1, (0, 1)), (v_1, (5, -10)), (v_2, (4, -10)),$$
$$(v_2, (0, -12.5)), (v_3, (0, -4)), (v_3, (-3, -2)),$$
$$(v_4, (-1, -2)), (v_4, (0, 0)), (v_1, (0, 1))\big).$$

CNF edge families. We sometimes annotate edges of rectangular automata with positive boolean combinations of guarded commands. Consider the two guarded commands ψ_1 and ψ_2. First, the edge label $\psi_1 \wedge \psi_2$ stands for a guarded command ψ_3 with $pre(\psi_3) = pre(\psi_1) \cap pre(\psi_2)$, $post(\psi_3) = post(\psi_1) \cap post(\psi_2)$, and $upd(\psi_3) = upd(\psi_1) \cup upd(\psi_2)$. Second, an edge with the label $\psi_1 \vee \psi_2$ stands for two edges that share source vertex, target vertex, and observation; one labeled with ψ_1 and the other with ψ_2. These conventions generalize to DNF expressions of guarded commands. An edge labeled with a CNF expression of guarded commands is interpreted by first converting the expression into DNF. A *CNF edge family*, then, consists of a pair (v, w) of vertices, an observation σ, and a CNF expression of guarded commands. Consider, for instance, the CNF edge family with the vertex pair (v, w), the observation σ, and the CNF expression

$$((x_1 < k \to x_1 := k) \vee (k \le x_1 \le k')) \wedge$$
$$((x_2 > k' \to x_2 := k') \vee (k \le x_2 \le k')).$$

This edge family corresponds to four edges from v to w, each labeled with the observation σ and one of the following guarded commands:

$$(1)\ x_1 < k \wedge x_2 > k' \to x_1 := k;\ x_2 := k',$$
$$(2)\ x_1 < k \wedge k \le x_2 \le k' \to x_1 := k,$$
$$(3)\ k \le x_1 \le k' \wedge x_2 > k' \to x_2 := k',$$
$$(4)\ k \le x_1 \le k' \wedge k \le x_2 \le k'.$$

In this way, an n-dimensional rectangular automaton may be specified by a set of vertices, an observation alphabet, invariant and activity functions, and a set of CNF edge families.

The reverse automaton. Let A be an n-dimensional rectangular automaton. The *reverse automaton* $-A$ is an n-dimensional rectangular automaton that defines the same state space as A, but with the transition relation reversed. The components of $-A$ are those of A, except for the following: for each vertex v, $act_{-A}(v) = \{\mathbf{x} \in \mathbb{R}^n \mid -\mathbf{x} \in act_A(v)\}$; for each edge $e = (v, w)$ of A, $-A$ has the edge $-e = (w, v)$ with $pre_{-A}(-e) = post_A(e)$, $upd_{-A}(-e) = upd_A(e)$, and $post_{-A}(-e) = pre_A(e)$.

Proposition 2.1 *For every rectangular automaton A, and every zone $R \subset Q_A$, $Pre_A(R) = Post_{-A}(R)$ and $Post_A(R) = Pre_{-A}(R)$.*

We consider the following two problems.

Reachability. Given a rectangular automaton A, and two rectangular zones $R, R' \subset Q_A$, is some state in R' reachable from some state in R (i.e., $R \Rightarrow_A^* R'$)?

Language inclusion. Given a rectangular automaton A, a rectangular zone $R \subset Q_A$, and a determinizable timed automaton T [AD94, AFH94] (or a formula T of the temporal logic MITL [AFH91]), is the timed language $Lang_A(R)$ contained in the timed language defined by T? This problem is more general than the reachability problem, and a solution permits the verification of timed and untimed safety requirements for systems that are modeled as rectangular automata. When measuring the complexity of the language-inclusion problem, we assume that T is given as a deterministic timed automaton.

For initialized automata, we provide a decision procedure for the language-inclusion problem (and therefore for reachability). We then show that the reachability problem (and therefore language inclusion) is undecidable for very restricted classes of uninitialized rectangular automata, and also for initialized automata with slightly generalized invariant, activity, preguard, postguard, or update functions.

3 Decidability

We translate initialized rectangular automata into timed automata [AD94]. The translation preserves timed languages, and therefore reduces the language-inclusion problem for initialized rectangular automata to the language-inclusion problem for timed automata, which is well understood. The translation proceeds in two steps, which are presented in the following two subsections.

3.1 Multirate To Timed Automata

Multirate and timed automata. The variable c is a *memory cell* of the rectangular automaton A if c has slope 0 in every vertex of A; that is, $act(v)(c) = [0,0]$ for all $v \in V$. The variable c is a *clock* if c has slope 1 in every vertex. The variable c is a *skewed clock* if there is a rational $k \in \mathbb{Q} \setminus \{0,1\}$ such that c has slope k in every vertex. The variable c is a *two-slope clock* if there is a rational $k \in \mathbb{Q}$ such that for each vertex v, either $act(v)(c) = [k,k]$ or $act(v)(c) = [1,1]$. A *stopwatch* is a two-slope clock with $k = 0$.

The rectangular automaton A is a *multirate automaton* if $act(v)$ is a singleton for all vertices v of A. The multirate automaton A is a *timed automaton* if each variable of A is either a clock or a memory cell. Every timed automaton is global, and therefore initialized. The reachability problem and the language-inclusion problem are PSPACE-complete for timed automata [AD94].[1]

Clock translation of initialized multirate automata. We reduce problems for initialized multirate automata to problems for timed automata, by translating a given initialized multirate automaton M into a timed automaton T_M such that M and T_M are timed bisimilar.[5]

Let M be an n-dimensional initialized multirate automaton. For each vertex $v \in V$, assuming $act_M(v) = \prod_{i=1}^{n}[k_i, k_i]$, define $\alpha_v : \mathbb{R}^n \to \mathbb{R}^n$ by $\alpha_v(x_1, \ldots, x_n) = (\frac{x_1}{m_1}, \ldots, \frac{x_n}{m_n})$, where $m_i = k_i$ if $k_i \neq 0$, and $m_i = 1$ if $k_i = 0$. The maps α_v are extended to regions in the natural way. The components of the timed automaton T_M are those of M, except for the following: for each $v \in V$, and all i, $inv_{T_M}(v) = \alpha_v(inv_M(v))$ and $act_{T_M}(v)_i = [l_i, l_i]$, where $l_i = 0$ if $k_i = 0$, and $l_i = 1$ if $k_i \neq 0$; and for each edge $e = (v,w)$, $pre_{T_M}(e) = \alpha_v(pre_M(e))$ and $post_{T_M}(e) = \alpha_w(post_M(e))$. Define $\alpha_M : Q_M \to Q_{T_M}$ such that $\alpha_M(v, \mathbf{x}) = (v, \alpha_v(\mathbf{x}))$.

Lemma 3.1 *Let M be an initialized multirate automaton. Then for every pair of states $q, q' \in Q_M$, and for every label $\pi \in \Sigma_M \cup \mathbb{R}_{\geq 0}$, $q \xrightarrow{\pi} q'$ in M iff $\alpha_M(q) \xrightarrow{\pi} \alpha_M(q')$ in T_M.*

It follows that the relation $\{(q, \alpha_M(q)) \mid q \in Q_M\}$ is a timed bisimulation between M and T_M.

Theorem 3.2 *Let M be an initialized multirate automaton. For every zone $R \subset Q_M$, $Post^*_M(R) = Post^*_{T_M}(\alpha_M(R))$, $Pre^*_M(R) = Pre^*_{T_M}(\alpha_M(R))$, and $Lang_M(R) = Lang_{T_M}(\alpha_M(R))$.*

[1] The [AD94] definition of timed automata does not allow memory cells, but their results can be extended to our case.

[5] We use the notions of timed and time-abstract (bi)-simulations as defined in [ACH94] for timed automata.

Corollary 3.3 [ACHH93, NOSY93] *The reachability problem and the language-inclusion problem for initialized multirate automata are PSPACE-complete.*

3.2 Rectangular to Multirate Automata

Skewed-clock translation of initialized rectangular automata. We reduce problems about initialized rectangular automata to problems about timed automata, by translating a given initialized rectangular automaton A of dimension n into an initialized multirate automaton M_A of dimension $2n$ such that M_A forward simulates A, and A backward simulates M_A.

Consider an n-dimensional initialized rectangular automaton A. We restrict our attention to the case where all values of inv_A, act_A, pre_A, and $post_A$ are compact. The generalization to arbitrary rectangular regions is straightforward and detailed in the full paper. Without loss of generality, we assume that for each edge $e = (v,w)$ of A, $pre_A(e) \subset inv_A(v)$, $post_A(e) \subset inv_A(w)$, and for each $i \notin upd_A(e)$, $pre_A(e)_i = post_A(e)_i$.

The idea is to replace each variable a with two variables a^l and a^u such that when $act_A(v)(a) = [k,k']$, then $act_{M_A}(v)(a_l) = [k,k]$ and $act_{M_A}(v)(a_u) = [k',k']$. Consider Figure 5. With each time step, the activity of a creates an envelope, whose boundaries are tracked by a_l and a_u. With each transition step, the values of a_l and a_u are updated so that the interval $[a_l, a_u]$ is precisely the range of possible values of a. In Figure 5, at time t a transition is taken along an edge e with $pre_A(e)(a) = [m, \infty)$. Since the value of a_l is below m at time t, a_l is updated to the new value m. In the following formal definition of the multirate automaton M_A, if the variable a corresponds to the ith coordinate of $\mathbb{R}^n$, we encode a_l by coordinate $2i-1$, and a_u by coordinate $2i$, for $1 \leq i \leq n$.

The multirate automaton M_A has dimension $2n$. It has the same observation alphabet as A, and uses four copies of each vertex of A, so $V_{M_A} = V_A \times \{0,1\}^2$. The invariant function inv_{M_A} is inherited from A: for each vertex $v \in V_A$, for all $\mu, \nu \in \{0,1\}$, and for all $1 \leq i \leq n$, $inv_{M_A}(v, \mu, \nu)_{2i-1} = inv_{M_A}(v, \mu, \nu)_{2i} = inv_A(v)_i$. On the vertices in $V_A \times \{0\} \times \{0\}$, the activity function act_{M_A} is defined as outlined above, the vertices in $V_A \times \{1\} \times \{0,1\}$ are used to prevent x_{2i-1} from moving below $\min inv_A(v)_i$, and the vertices in $V_A \times \{0,1\} \times \{1\}$ are used to prevent x_{2i} from moving above $\max inv_A(v)_i$: if $act_A(v)_i = [k,k']$, then for all $\mu, \nu \in \{0,1\}$, $act_{M_A}(v,0,\nu)_{2i-1} = [k,k]$, $act_{M_A}(v,\mu,0)_{2i} = [k',k']$, and $act_{M_A}(v,1,\nu)_{2i-1} = act_{M_A}(v,\mu,1)_{2i} = [0,0]$.

We are left with defining a set of CNF edge families for M_A. For each edge $e = (v, w)$ of A, the multirate automaton M_A has the CNF edge family $((v,0,0),(w,0,0), obs_A(e), \psi_e)$ that shares the observation of e. The CNF expression ψ_e is a conjunction $\bigwedge_{i=1}^{n} \psi_e^i$ of n CNF expressions ψ_e^i. Suppose that $pre_A(e)_i = [k, k']$ and $post_A(e)_i = [m, m']$. If $i \in upd_A(e)$, then the CNF expression ψ_e^i is the guarded command

$$x_{2i-1} \leq k' \wedge x_{2i} \geq k \;\rightarrow\; x_{2i-1} := m;\; x_{2i} := m'.$$

The values of x_{2i-1} and x_{2i} satisfy the guard iff $[x_{2i-1}, x_{2i}]$ intersects $pre_A(e)_i$. Since $i \in upd_A(e)$, the range of values of x_i after traversal of e in A is exactly $post_A(e)_i$, and hence x_{2i-1} is set to the minimum of this interval, and x_{2i} is set to the maximum. If $i \notin upd_A(e)$, then by assumption $[k, k'] = [m, m']$, and the CNF expression ψ_e^i is

$$((x_{2i-1} < k \;\rightarrow\; x_{2i-1} := k) \vee (k \leq x_{2i-1} \leq k')) \wedge$$
$$((x_{2i} > k' \;\rightarrow\; x_{2i} := k') \vee (k \leq x_{2i} \leq k')).$$

The idea is that if the edge e is traversed in A, new information becomes available about the value of x_i, namely, that it lies within the interval $[k, k']$. Therefore, if $x_{2i-1} < k$, it must be updated to k, and if $x_{2i} > k'$, it must be updated to k', in order to keep $[x_{2i-1}, x_{2i}]$ in M_A the range of possible values of x_i in A.

If the invariant function of A is trivial, the construction of M_A is complete. Otherwise, the multirate automaton M_A has additional τ edges to stop its skewed clocks from leaving the regions imposed by inv_A. Suppose that $act_A(v)_i = [k, k']$ and $inv_A(v)_i = [m, m']$. If $k < 0$, then there is an edge e from $(v, 0, \nu)$ to $(v, 1, \nu)$ with $obs_{M_A}(e) = \tau$, $pre_{M_A}(e)_{2i-1} = [m, m]$, and $pre_{M_A}(e)_j = \mathbb{R}$ for all $j \neq 2i - 1$. If $k' > 0$, then there is an edge e from $(v, \mu, 0)$ to $(v, \mu, 1)$ with $obs_{M_A}(e) = \tau$, $pre_{M_A}(e)_{2i} = [m', m']$, and $pre_{M_A}(e)_j = \mathbb{R}$ for all $j \neq 2i$. In addition, the multirate automaton M_A has an edge e with $obs_{M_A}(e) = \tau$ and $pre_{M_A}(e) = \mathbb{R}^n$ from every vertex $(v, 1, \nu)$ to $(v, 0, \nu)$, and from every vertex $(v, \mu, 1)$ to $(v, \mu, 0)$. For all of these τ edges e, $upd_{M_A}(e) = \emptyset$ and $post_{M_A}(e) = pre_{M_A}(e)$.

This completes the definition of the multirate automaton M_A. The multirate automaton M_A has $4^{n-|upd(e)|}$ edges for each edge e of A, and $O(|V_A|)$ additional τ edges. The multirate automaton M_A is initialized. For example, Figure 2 gives the initialized multirate automaton $M_{\hat{A}}$ that corresponds to the initialized rectangular automaton $\hat{A}$ of Figure 1 (since $\hat{A}$ has the trivial invariant, only four vertices of $M_{\hat{A}}$ are relevant). Figure 3 shows the timed automaton T_{M_A} that corresponds to $M_{\hat{A}}$.

Reachability and language inclusion for initialized rectangular automata. We define the map $\beta_A : Q_{M_A} \to 2^{Q_A}$ by $\beta_A((v, \mu, \nu), \mathbf{x}) = \{v\} \times \prod_{i=1}^{n} [x_{2i-1}, x_{2i}]$. The map β_A is extended to zones in the natural way. For two zones $R, R' \subset Q_A$, and a label $\pi \in \Sigma_A \cup \mathbb{R}_{\geq 0}$, define $R \xrightarrow{\pi} R'$ iff R' is the zone of states that are reachable from R in one $\xrightarrow{\pi}$-step; that is, $R' = \{q' \in Q_A \mid \exists q \in R.\, q \xrightarrow{\pi} q'\}$. The *upper half-space* U_{M_A} of M_A is the zone of all states $((v, \mu, \nu), \mathbf{x}) \in Q_{M_A}$ such that $x_{2i-1} \leq x_{2i}$ for all $1 \leq i \leq n$.

Lemma 3.4 *Let A be an initialized rectangular automaton. Then for every pair of states $q, q' \in U_{M_A}$, and for every label $\pi \in \Sigma_A \cup \mathbb{R}_{\geq 0}$, $q \xrightarrow{\pi} q'$ in M_A iff $\beta_A(q) \xrightarrow{\pi} \beta_A(q')$ in A.*

Theorem 3.5 *Let A be an initialized rectangular automaton. For every zone $R \subset U_{M_A}$, $Post^*_A(\beta_A(R)) = \beta_A(Post^*_{M_A}(R))$, $Pre^*_A(\beta_A(R)) = \beta_A(Pre^*_{-(M-A)}(R))$, and $Lang_A(\beta_A(R)) = Lang_{M_A}(R)$.*

While the multirate automaton M_A has exponentially more edges than the rectangular automaton A, it has only four times the number of vertices, and only twice as many variables.

Corollary 3.6 *The reachability problem and the language-inclusion problem for initialized rectangular automata are PSPACE-complete.*

The verification tool HyTech contains a semi-decision procedure that attacks the reachability problem for hybrid automata by iterating *Pre* and *Post* operations on zones [AHH93]. The HyTech procedure is known to terminate on timed automata [HNSY94]. We obtain the following corollary, which asserts that the HyTech procedure terminates on all initialized rectangular automata.

Corollary 3.7 *Let A be an initialized rectangular automaton. For every rectangular zone $R \subset Q_A$, there is a natural number $i \in \mathbb{N}$ such that $Pre^*_A(R) = Pre^i_A(R)$ and $Post^*_A(R) = Post^i_A(R)$.*

Simulation relations. While the initialized rectangular automaton A and the multirate automaton M_A define the same timed languages, they are not timed bisimilar. Lemma 3.4 implies only that M_A forward simulates A, and A backward simulates M_A. To see this, we need a few definitions. Let A and A' be two rectangular automata with the state spaces Q_A and $Q_{A'}$, respectively. A relation $\gamma \subset Q_A \times Q_{A'}$ is a *forward simulation of A' by A* if for each state $q' \in Q_{A'}$, there is a state $q \in Q_A$ with $(q, q') \in \gamma$; and whenever $q' \xrightarrow{\pi} r'$ in A', and $(q, q') \in \gamma$, then there is a state $r \in Q_A$ such that $(r, r') \in \gamma$ and $q \xrightarrow{\pi} r$ in A. The relation γ is a *backward simulation of A' by A* if for each state $q' \in Q_{A'}$, there is a state $q \in Q_A$ with

$(q,q') \in \gamma$; and whenever $q' \xrightarrow{\pi} r'$ in A', and $(r,r') \in \gamma$, then there is a state $q \in Q_A$ such that $(q,q') \in \gamma$ and $q \xrightarrow{\pi} r$ in A. Notice that γ is a forward simulation of A' by A iff γ is a backward simulation of $-A'$ by $-A$.

We define the relation $\gamma_A \subset Q_{M_A} \times Q_A$ by $(q,q') \in \gamma_A$ iff $q' \in \beta_A(q)$. For backward simulations, we restrict our attention to the upper half-space of M_A (replace Q_{M_A} by U_{M_A} in the definition of backward simulation).

Proposition 3.8 *Let A be an initialized rectangular automaton. The relation γ_A is a forward simulation of A by M_A, and γ_A^{-1} is a backward simulation of M_A, restricted to its upper half-space, by A.*

Now consider a 1D rectangular automaton A. Suppose A has two edges e and e' that share the source v with $pre(e) = [k,k]$ and $pre(e') = [m,m]$, where $m > k$. While no state of A can traverse both edges, the state $((v,0,0),k,m)$ can traverse both corresponding edges in M_A.

Proposition 3.9 *There exists an initialized rectangular automaton A such that there is no forward simulation of M_A by A, and no backward simulation of A by M_A.*[6]

4 Undecidability

The n-dimensional rectangular automaton A is *simple* if the following restrictions are met:

1. The invariant function *inv* is constant and its value is compact.
2. For every edge $e \in E$, and for all $1 \le i \le n$, if $i \in upd(e)$ then $post(e)_i = [0,0]$, and if $i \notin upd(e)$ then $post(e)_i = pre(e)_i$.
3. For every edge $e \in E$, $pre(e)$ is compact.
4. At most one variable of A is *not* a clock.

The automaton A is *k-simple* if it meets restrictions 1–3, and at most k variables of A are not clocks. We use simple automata for our undecidability results. Restrictions 1 and 2 imply that invariants, postguards, and nondeterministic variable updates do not contribute to our undecidability results. Many limited decidability results are based on the digitization of real-numbered delays [HMP92, BES93, BER94, PV94a]. Since the digitization technique requires closed preguards and invariants, restrictions 1 and 3 imply that the technique does not generalize beyond very special cases. Many limited decidability results apply to automata with a single stopwatch [ACH93, BES93, KPSY93, BER94]. Restriction 4 implies that these results do not generalize beyond very special cases.

All of our undecidability proofs are reductions from the halting problem for two-counter machines to the reachability problem for rectangular automata. Each counter is modeled by a clock. Counter value k corresponds to clock value 2^{1-k}. Hence decrementing (incrementing) a counter corresponds to doubling (halving) the value of the corresponding clock. We implement halving via doubling, so the latter is the key to the proofs.

4.1 Uninitialized Automata

We show that initialization is necessary for a decidable reachability problem.

Theorem 4.1 *For every slope $k \in \mathbb{Q} \setminus \{1\}$, the reachability problem is undecidable for simple rectangular automata with one two-slope clock of slopes 1 and k.*

We first prove three lemmas that are basic to all of our undecidability proofs. In figures of simple automata, all variables whose slopes are not listed are clocks—they have slope 1. A rectangular automaton is *wrapping* if (1) for each edge $e = (v,w)$, and each variable a, $pre(e)(a) \subset [0, 4 \cdot act(v)(a)]$, and (2) for every vertex v, and every variable a, $inv(v)(a) = [0, 4k_a]$, where $k_a = \max_{w \in V} \max act(w)(a)$ is the largest slope allowed to a in any vertex. In figures we leave these wrapping conditions implicit. We use the following wrapping technique found originally in [Cer92].

Wrapping lemma. *Let $k_1, k_2 \in \mathbb{Q}_{>0}$, and consider the simple wrapping automaton fragment of Figure 6. Suppose that when edge e_1 is traversed into v_1, $c = \gamma$ and $d = \delta$, where $0 < \gamma \le 2k_1$ and $0 < \delta \le 2k_2$. Then the next time e_4 is traversed out of v_1, c has value γ and d has value δ.*

The following device allows us to test if two variables with the same slope have the same value.

Equality lemma. *Let $k_1 \in \mathbb{Q}_{>0}$, and consider the simple wrapping automaton fragment of Figure 7. Suppose that when edge e_1 is traversed into v_1, $c = \gamma$ and $d = \delta$, where $0 < \gamma, \delta \le 2k_1$. Then edge e_3 can later be traversed iff $\gamma = \delta$, and the next time e_3 is traversed, both c and d have value γ $(= \delta)$.*

Similarly, given two skewed clocks c and d with slopes k_1 and k_2, respectively, it is possible to assign to d the value of c multiplied by the ratio of their slopes.

Assignment lemma. *Let $k_1, k_2 \in \mathbb{Q}_{>0}$, and consider the simple wrapping automaton fragment of Figure 8. Suppose that when edge e_1 is traversed into v_1, $c = \gamma$, where $0 < \gamma \le 2k_1$. Then the next time e_3 is traversed, c has value γ and d has value $\frac{k_2}{k_1}\gamma$.*

[6] While we have defined *timed* simulations, our proof shows that Proposition 3.9 also holds for *time-abstract* simulations.

We now prove Theorem 4.1. Let a, b, c, and d be clocks, and let z be a two-slope clock of slopes 1 and k. We first prove the result for $k > 0$. Without loss of generality, assume $k = 2$. We encode the values of the two counters C and D using the clocks c and d, respectively. The test result $C = 0$ corresponds to an edge e with $pre(e)(c) = [2,2]$, and the test result $C \neq 0$ corresponds to an edge e with $pre(e)(c) = [0,1]$. Using the equality lemma, halving can be implemented by guessing, doubling, and checking; so once we have implemented doubling, the proof will be complete. To double the value of c while maintaining the value of d, we put two assignment-lemma fragments in series (see Figure 9). In the first, $\dot{z} = 2$; it assigns to z twice the original value of c. In the second, $\dot{z} = 1$; it assigns to c the value of z, which is twice the original value of c. The original value of d is maintained upon exit.

The case $k < 0$ can be handled similarly, and is detailed in the full paper.

We now turn to the case $k = 0$. The encoding of the two-counter machine is the same as before. In Figure 10, we give a simple wrapping automaton fragment that doubles the value of clock c while maintaining the value of clock d, using two synchronization clocks a and b, and stopwatch z (to avoid clutter, we have omitted the wrapping edges from each vertex).

4.2 Generalized Automata

A slight generalization of the invariant, activity, preguard, postguard, or update function leads to the undecidability of rectangular automata, even under the stringent restrictions of simplicity and initialization.

Assignment updates. The update function can be generalized to allow the value of one variable to be assigned to another variable. An *assignment update* assigns to each edge e both an update set $upd(e) \subset \{1,\ldots,n\}$ and an assignment function $assign(e) : \{1,\ldots,n\} \to \{1,\ldots,n\}$. The transition-step relation $\stackrel{\sigma}{\Rightarrow}$ is then redefined as follows: $(v,\mathbf{x}) \stackrel{\sigma}{\Rightarrow} (w,\mathbf{y})$ iff there is an edge $e = (v,w)$ with $obs(e) = \sigma$, $\mathbf{x} \in pre(v)$, $\mathbf{y} \in post(w)$, and for all $i \notin upd(e)$, $y_i = x_{assign(i)}$. Using assignment updates and one skewed clock, or assignment updates and one memory cell, the proof of Theorem 4.1 can be duplicated. The latter gives a new proof of a result from [Cer92].

Corollary 4.2 *For every slope $k \in \mathbb{Q} \setminus \{0,1\}$, the reachability problem is undecidable for simple initialized automata with one skewed clock of slope k (resp. one memory cell) and assignment updates.*

Triangular preguards, postguards, invariants. The preguard, postguard, and invariant functions can be generalized to allow comparisons between the values of variables. A *triangular restriction* $\leq$ is a partial order on $\{1,\ldots,n\}$. A *triangular preguard* (resp. *postguard*) assigns to each edge e both a rectangular region $pre(e)$ (resp. $post(e)$) and a triangular restriction $\leq_e$. The transition-step relation $\stackrel{\sigma}{\Rightarrow}$ is then redefined as follows: $(v,\mathbf{x}) \stackrel{\sigma}{\Rightarrow} (w,\mathbf{y})$ iff there is an edge $e = (v,w)$ with $obs(e) = \sigma$, $\mathbf{x} \in pre(v)$, $\mathbf{y} \in post(w)$, for all $i \notin upd(e)$, $x_i = y_i$, and for all i and j with $i \leq_e j$, $x_i \leq x_j$ (resp. $y_i \leq y_j$). A *triangular invariant* assigns to each vertex v both a rectangular region $inv(v)$ and a triangular restriction $\leq_v$. The set Q_A of admissible states is then redefined to contain a state $(v,\mathbf{x}) \in V \times \mathbb{R}^n$ iff $\mathbf{x} \in inv(v)$ and for all i and j with $i \leq_v j$, $x_i \leq x_j$. Using one skewed clock and any of these three types of triangular conditions, the proof of Theorem 4.1 can be duplicated.

Corollary 4.3 *For every slope $k \in \mathbb{Q} \setminus \{0,1\}$, the reachability problem is undecidable for simple initialized automata with one skewed clock of slope k and triangular preguards (resp. postguards; invariants).*

Triangular activities. The activity functions can be generalized to impose an order on the derivatives of variables. A *triangular activity* assigns to each vertex v both a rectangular region $act(v)$ and a triangular restriction $\leq_v$. For $t > 0$, the time-step relation $\stackrel{t}{\Rightarrow}$ is then redefined as follows: $(v,\mathbf{x}) \stackrel{t}{\Rightarrow} (w,\mathbf{y})$ iff $v = w$, $\frac{\mathbf{y}-\mathbf{x}}{t} \in act(v)$, and for all i and j with $i \leq_v j$, $y_i - x_i \leq y_j - x_j$. A triangular activity is *global* if the functions act and $\lambda v.\ \leq_v$ are both constant functions on the set of vertices. Using three variables and a global triangular activity, we can simulate the behavior of the two-slope clock from Theorem 4.1.

Corollary 4.4 *The reachability problem is undecidable for 3-simple automata with a global triangular activity.*

A decidable class of triangular activities. A *clock-partition activity* is a global triangular activity that assigns to each vertex v the rectangular region $[0,\infty)^n$ and a symmetric triangular restriction $\leq$. Note that $\leq$ induces a partition of $\{1,\ldots,n\}$, which can be viewed as a partition of a distributed system into individual processes. Clock-partition activities, then, model distributed systems that are composed of perfectly asynchronous processes, where the clocks within each process are perfectly synchronized.

Theorem 4.5 *The reachability problem and the language-inclusion problem for rectangular automata with clock-partition activities are PSPACE-complete.*

We prove Theorem 4.5 by showing that every rectangular automaton with a clock-partition activity has a finite time-abstract bisimulation, which is a product of region equivalences for timed automata [AD94].

Acknowledgement. We thank Howard Wong-Toi for a careful reading.

References

[ACH93] R. Alur, C. Courcoubetis, T.A. Henzinger. Computing accumulated delays in real-time systems. *CAV 93*, Springer LNCS 697, 1993, pp. 181–193.

[ACH+94] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[ACH94] R. Alur, C. Courcoubetis, T.A. Henzinger. The observational power of clocks. *CONCUR 94*, Springer LNCS 836, 1993, pp. 162–177.

[ACHH93] R. Alur, C. Courcoubetis, T.A. Henzinger, P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. *Hybrid Systems*, Springer LNCS 736, 1993, pp. 209–229.

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

[AFH91] R. Alur, T. Feder, T.A. Henzinger. The benefits of relaxing punctuality. *10th ACM PODC*, 1991, pp. 139–152.

[AFH94] R. Alur, L. Fix, T.A. Henzinger. A determinizable class of timed automata. *CAV 94*, Springer LNCS 818, 1994, pp. 1–13.

[AHH93] R. Alur, T.A. Henzinger, P.-H. Ho. Automatic symbolic verification of embedded systems. *14th IEEE RTSS*, 1993, pp. 2–11.

[AHV93] R. Alur, T.A. Henzinger, M.Y. Vardi. Parametric real-time reasoning. *25th ACM STOC*, 1993, pp. 592–601.

[BER94] A. Bouajjani, R. Echahed, R. Robbana. Verifying invariance properties of timed systems with duration variables. *FTRTFT 94*, Springer LNCS 863, 1994, pp. 193–210.

[BES93] A. Bouajjani, R. Echahed, J. Sifakis. On model checking for real-time properties with durations. *8th IEEE LICS*, 1993, pp. 147–159.

[BPV94] D. Bosscher, I. Polak, F. Vaandrager. Verification of an audio-control protocol. *FTRTFT 94*, Springer LNCS 863, 1994, pp. 170–192.

[Cer92] K. Cerāns. *Algorithmic Problems in Analysis of Real-time System Specifications.* PhD thesis, Univ. of Latvia, 1992.

[HH94] T.A. Henzinger, P.-H. Ho. Model-checking strategies for linear hybrid systems. *Workshop on Hybrid Systems and Autonomous Control* (Ithaca, NY), 1994.

[HH95] T.A. Henzinger, P.-H. Ho. Algorithmic analysis of nonlinear hybrid systems. *CAV 95*, Springer LNCS, 1995.

[HMP92] T.A. Henzinger, Z. Manna, A. Pnueli. What good are digital clocks? *ICALP 92*, Springer LNCS 623, 1992, pp. 545–558.

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

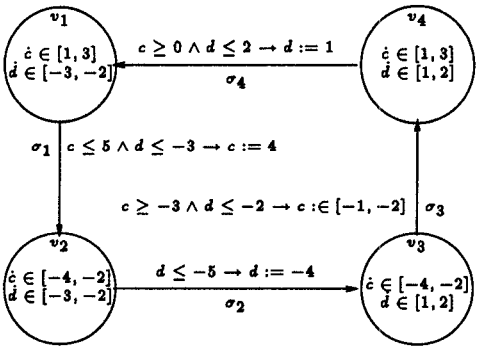

Figure 1: The initialized rectangular automaton $\hat{A}$

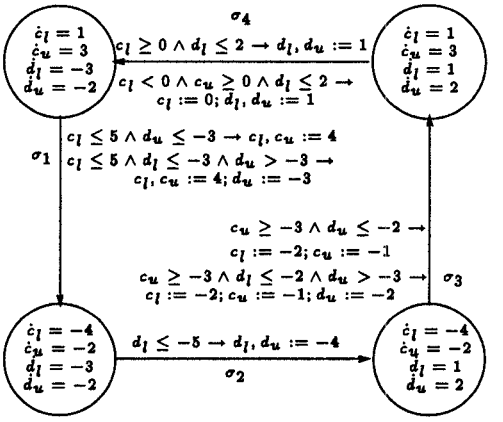

Figure 2: The initialized multirate automaton $M_{\hat{A}}$

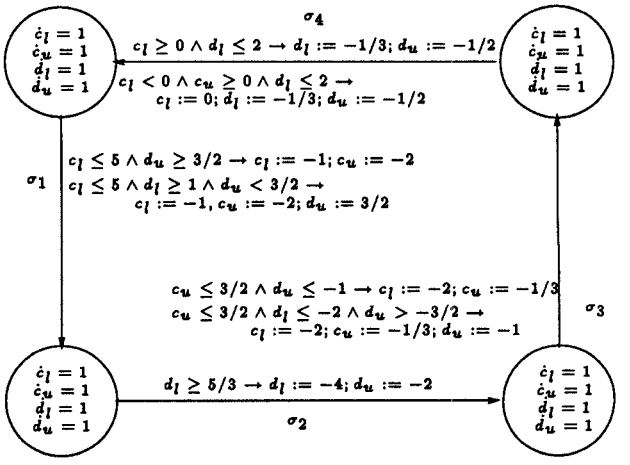

Figure 3: The timed automaton $T_{M_{\hat{A}}}$

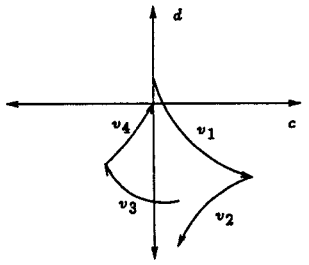

Figure 4: A sample trajectory of $\hat{A}$

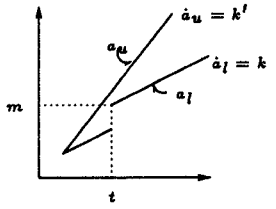

Figure 5: Envelope of the activity $act(v)(a) = [k, k']$

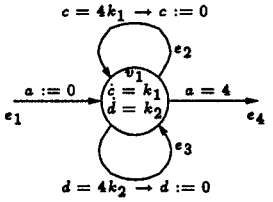

Figure 6: Wrapping lemma: both c and d retain their entry values upon exit

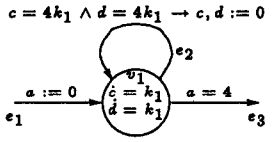

Figure 7: Equality lemma: testing $c = d$

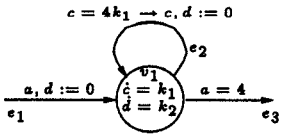

Figure 8: Assignment lemma: performing the assignment $d := \frac{k_2}{k_1} c$

[HRP94] N. Halbwachs, P. Raymond, Y.-E. Proy. Verification of linear hybrid systems by means of convex approximation. *1st Static Analysis Symp.*, 1994.

[KPSY93] Y. Kesten, A. Pnueli, J. Sifakis, S. Yovine. Integration graphs: a class of decidable hybrid systems. *Hybrid Systems*, Springer LNCS 736, 1993, pp. 179–208.

[MP93] O. Maler, A. Pnueli. Reachability analysis of planar multi-linear systems. *CAV 93*, Springer LNCS 697, 1993, pp. 194–209.

[MPS95] O. Maler, A. Pnueli, J. Sifakis. On the synthesis of discrete controllers for timed systems. *STACS 95*, Springer LNCS, 1995.

[MV94] J. McManis, P. Varaiya. Suspension automata: a decidable class of hybrid automata. *CAV 94*, Springer LNCS 818, 1994, pp. 105–117.

[NOSY93] X. Nicollin, A. Olivero, J. Sifakis, S. Yovine. An approach to the description and analysis of hybrid systems. *Hybrid Systems*, Springer LNCS 736, 1993, pp. 149–178.

[OSY94] A. Olivero, J. Sifakis, S. Yovine. Using abstractions for the verification of linear hybrid systems. *CAV 94*, Springer LNCS 818, 1994, pp. 81–94.

[PV94a] A. Puri, P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. *CAV 94*, Springer LNCS 818, 1994, pp. 95–104.

[PV94b] A. Puri, P. Varaiya. Verification of hybrid systems using abstractions. *Workshop on Hybrid Systems and Autonomous Control* (Ithaca, NY), 1994.

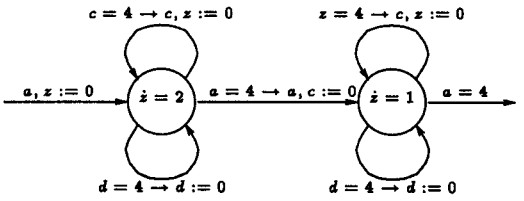

Figure 9: Doubling c using the two-slope clock z

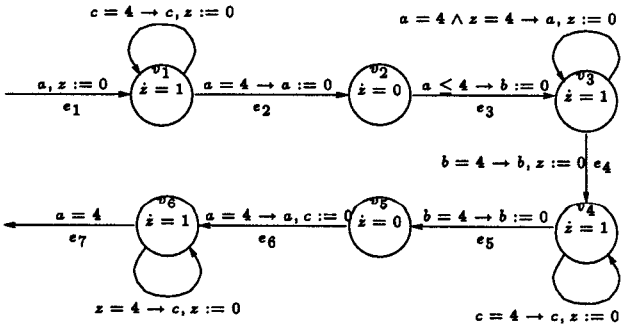

Figure 10: Doubling c using the stopwatch z