What's Decidable About Hybrid Automata?*

Thomas A. Henzinger† Peter W. Kopke‡
Computer Science Department, Cornell University
Ithaca, NY 14853, U.S.A.

Anuj Puri§ Pravin Varaiya¶
Department of Electrical Engineering and Computer Science
University of California, Berkeley, CA 94720.

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 the construction of a timed automaton that contains all reachability information about a given initialized rectangular automaton. The translation has practical significance for verification, because it guarantees the termination of symbolic procedures for the reachability analysis of initialized rectangular automata. The translation also preserves the ω-languages of initialized rectangular automata with bounded nondeterminism.

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 augmented with a single stopwatch.

1 Introduction

A hybrid automaton [ACHH93, ACH+95] combines the discrete dynamics of a finite automaton with the continuous dynamics of a dynamical system. Hybrid automata thus provide a mathe-

*This research was supported in part by the Office of Naval Research Young Investigator award N00014-95-1-0520, by the National Science Foundation CAREER award CCR-9501708, by the National Science Foundation grants CCR-9200794 and CCR-9504469, by the Air Force Office of Scientific Research contract F49620-93-1-0056, by the Advanced Research Projects Agency grant NAG2-892, by the U.S. Army Research Office through the Mathematical Sciences Institute of Cornell University, Contract Number DAAL03-91-C-0027, by the U.S. Army Research Office under contract DAAH04-94-G-0026, and by the California PATH program.

†Phone: (607) 255-3009. Fax: (607) 255-4428. Email: tah@cs.cornell.edu

‡Email: pkpk@cs.cornell.edu

§Email: anuj@eecs.berkeley.edu

¶Phone: (510) 642-5270. Fax: (510) 642-6330. Email: varaiya@eecs.berkeley.edu

matical 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, DOY94, HRP94, MV94, HH95b, HHWT95, HW95, MPS95, PV95]. Two problems that are central to the analysis of hybrid automata are the reachability problem and the more general ω-language emptiness problem. The solution of the reachability problem for a given hybrid automaton allows us to check if the trajectories of the automaton meet a given safety requirement; the solution to the ω-language emptiness problem allows us to check if the trajectories of the automaton meet a liveness requirement [VW86]. 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, PV94]—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, HH95a, PV95], and from a theoretical point of view, as they lie at the boundary of decidability.

Our results are threefold. First, we give an (optimal) PSPACE algorithm for the reachability problem for rectangular-activity automata with two restrictions: (1) whenever the activity of a variable changes, the value of the variable is reinitialized; (2) the values of two variables with different activities are never compared. Second, under the additional assumption of bounded nondeterminism (in particular, bounded activities), we obtain a PSPACE algorithm for checking ω-language emptiness of rectangular-activity automata. Third, we prove that the reachability problem becomes undecidable if either one of the restrictions (1) and (2) is relaxed, or if more general, triangular activities are admitted. The first two results are proven by translating rectangular-activity automata of dimension n into timed automata of dimension $2n$, where the dimension is the number of real-valued variables. The third result is proven by a reduction from the halting problem for two-counter machines.

We now place these results in the context of previous work.

Decidability [Section 3]. The first decidability result for hybrid automata was obtained for timed automata, whose reachability and ω-language emptiness 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 automata with closed rectangular activities [PV94]. While the latter result was based on the discretization of time, we present a reachability preserving and (in the case of bounded nondeterminism) ω-language preserving translation from rectangular-activity automata via multirate automata to timed automata. Unlike discretization-based arguments, our translation applies to all rectangular activities. Moreover, and perhaps most importantly, the translation implies that, when applied to rectangular-activity automata that meet restrictions (1) and (2), existing semi-decision procedures for the reachability problem of hybrid automata terminate. Such procedures have been implemented in the HyTech verification tool [AHH93, HHWT95].

Undecidability [Section 4]. Over the past few years, there have been several 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, BR95], 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 the undecidability 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 augmented 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 augmented 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 reachability and (in the case of bounded nondeterminism) ω-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 that compare clocks 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 edge steps and time steps. During an edge 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.

Notation. We use the symbol $\mathbb{R}_{\geq 0}$ to denote the set $\{x \in \mathbb{R} \mid x \geq 0\}$ of nonnegative reals. We use the boldface characters $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ for vectors in $\mathbb{R}^n$, and subscripts on italic characters such as x_i, y_j, and z_k for components of vectors.

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 region $B \subset \mathbb{R}^n$ is *rectangular* if it is a cartesian product of (possibly unbounded) intervals, 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 set of all rectangular regions in $\mathbb{R}^n$ is denoted $\mathcal{B}^n$.

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, three vertex labeling functions $init_A : V_A \to \mathcal{B}^n$, $inv_A : V_A \to \mathcal{B}^n$, 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 $obs_A : E_A \to \Sigma_A$. An *n-dimensional rectangular automaton with silent moves* A differs in that the function obs_A maps E_A into Σ_A^τ, 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 *initial function init* specifies a set of initial automaton states. When the discrete state begins at v, the continuous state must begin in $init(v)$. The *preguard function pre*, the *postguard function post*, and the *update function upd* constrain the behavior of the automaton state during edge steps. The 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 or 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 $act(v)$. A rectangular automaton with silent moves may traverse τ-edges during time steps.

Note that if we replace rectangular regions with arbitrary linear regions in our definition of rectangular automata, we obtain the linear hybrid automata of [ACHH93]. Thus rectangular automata are the subclass of linear hybrid automata in which all defining regions are rectangular.

Initialization and bounded nondeterminism

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 has *bounded nondeterminism* if (1) for every vertex v, $init(v)$ and $act(v)$ are bounded, and (2) for every edge e, and every $i \in \{1, \ldots, n\}$, if $i \in upd(e)$, then $post(e)$ is bounded. Note that bounded nondeterminism does not imply finite branching. It ensures that the successor of a bounded region is bounded.

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 $Z \subset Q_A$ 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 Z is *rectangular* (resp. *bounded*, *compact*), if each region R^v is rectangular (resp. bounded, compact). The zone Z is *multirectangular* if it is a finite union of rectangular zones. The state $(v, \mathbf{x})$ of A is *initial* if $\mathbf{x} \in init(v)$. The *initial zone* $Init_A \subset Q_A$ is the set of all initial states of A. Notice that both the state space Q_A and the initial zone $Init_A$ are rectangular.

We now define a labeled transition relation $\xrightarrow{\pi}$ on Q_A, where $\pi \in \mathbb{R}_{\geq 0} \cup \Sigma_A$. For each edge $e = (v, w) \in E$, we define the relation $\xrightarrow{e}$ on Q_A by $(v, \mathbf{x}) \xrightarrow{e} (w, \mathbf{y})$ iff $\mathbf{x} \in pre(e)$, $\mathbf{y} \in post(e)$, 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 $upd(e)$. For each observation $\sigma \in \Sigma^\tau$, we define the *edge-step relation* $\xrightarrow{\sigma}$ on Q_A by $(v, \mathbf{x}) \xrightarrow{\sigma} (w, \mathbf{y})$ iff $(v, \mathbf{x}) \xrightarrow{e} (w, \mathbf{y})$ for some edge $e \in E$ with $obs(e) = \sigma$.

For each nonnegative real $t \in \mathbb{R}_{\geq 0}$, we define the 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 $act(v)$. If A does not have silent moves, then we define the *time-step relation* $\xrightarrow{t}$ to be $\overset{t}{\Rightarrow}$. If A has silent moves, then the time-step relation $\xrightarrow{t}$ is defined by $q \xrightarrow{t} q'$ iff there exists an integer $m \geq 1$, nonnegative reals $t_1, \ldots, t_m$, and states $q_1, \ldots, q_{2m-2}$ such that $q \overset{t_1}{\Rightarrow} q_1 \xrightarrow{\tau} q_2 \overset{t_2}{\Rightarrow} q_3 \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_{2m-2} \overset{t_m}{\Rightarrow} q'$ and $t = \sum_{i=1}^m t_i$.

Given a zone $Z \subset Q_A$, and a label $\pi \in \mathbb{R}_{\geq 0} \cup \Sigma \cup E$, let

$$Post_A^\pi(Z) = \{q \in Q_A \mid \exists r \in Z . r \xrightarrow{\pi} q\}$$

be the zone of states that are reachable in one π-step from Z, and let $Post_A(Z) = \bigcup_{\pi \in \mathbb{R}_{\geq 0} \cup \Sigma} Post_A^\pi(Z)$ be the zone of states that are reachable in one edge or time step from Z. Similarly, let

$$Pre_A^\pi(Z) = \{q \in Q_A \mid \exists r \in Z . q \xrightarrow{\pi} r\}$$

be the zone of states from which Z is reachable in one π-step, and let $Pre_A(Z) = \bigcup_{\pi \in \mathbb{R}_{\geq 0} \cup \Sigma} Pre_A^\pi(Z)$ be the zone of states from which Z is reachable in one edge or time step. Notice that $Post_A(Z) \supset Z$ and $Pre_A(Z) \supset Z$ because of time steps of the form $\xrightarrow{0}$. We define $Post_A^{\pi_0 \pi_1 \cdots \pi_k}(Z)$ and $Pre_A^{\pi_0 \pi_1 \cdots \pi_k}(Z)$ for a finite word $\pi_0 \pi_1 \cdots \pi_k$ inductively in the usual way. We define $Post_A^*(Z) = \bigcup_{i \in \mathbb{N}} Post_A^i(Z)$ and $Pre_A^*(Z) = \bigcup_{i \in \mathbb{N}} Pre_A^i(Z)$. Then $Post_A^*(Z)$ is the zone of states that are reachable from Z in a finite number of steps. A state $q \in Q_A$ is *reachable* if $q \in Post_A^*(Init_A)$. The zone of reachable states of A is denoted $Reach_A$.

Proposition 2.1 *For every rectangular automaton A, every multirectangular zone $Z \subset Q_A$, and every label $\pi \in \mathbb{R}_{\geq 0} \cup \Sigma^\tau \cup E$, $Post_A^\pi(Z)$ and $Pre_A^\pi(Z)$ are multirectangular zones.*

Proof. We give the proof for *Post*; the result for *Pre* then follows from Proposition 2.2 below. Since each relation $\xrightarrow{\sigma}$ with $\sigma \in \Sigma^\tau$ is a finite union of relations $\xrightarrow{e}$ with $e \in E$, it suffices to prove the proposition for $\pi \in \mathbb{R}_{\geq 0} \cup E$. Call a zone *elementary* if it is of the form $\{v\} \times R$, where R is a rectangular region. Then a zone is multirectangular iff it is a finite union of elementary zones. We show that for any elementary zone $Z' = \{v\} \times R$, $Post_A^\pi(Z')$ is elementary. If $\pi = (v, w) \in E$, then $Post_A^\pi(Z') = \{w\} \times R'$, where

$$R'_i = \begin{cases} post(\pi)_i, & \text{if } i \in upd(\pi)_i, \\ R_i, & \text{if } i \notin upd(\pi)_i. \end{cases}$$

If $\pi \in \mathbb{R}_{\geq 0}$, then $Post_A^{\pi}(Z') = \{v\} \times R'$, where R' is a rectangular region satisfying

$$\inf R'_i = \max\{\inf inv(v)_i, \inf R_i + \pi \cdot \inf act(v)\}$$

and

$$\sup R'_i = \min\{\sup inv(v)_i, \sup R_i + \pi \cdot \sup act(v)\},$$

where we have used the convention that $0 \cdot \infty = 0 \cdot (-\infty) = 0$. ■

The ω-language of a rectangular automaton

Let A be a rectangular automaton, possibly with silent moves, and let $Z \subset Q_A$ be a zone. A *timed word* for A is an infinite sequence over the alphabet $\mathbb{R}_{\geq 0} \cup \Sigma_A$. A *$Z$-run* ρ of A is an infinite sequence of the form $q_0 \xrightarrow{\pi_0} q_1 \xrightarrow{\pi_1} q_2 \xrightarrow{\pi_2} \cdots$, where $q_0 \in Z$, and for all $i \geq 0$, $q_i \in Q_A$ and $\pi_i \in \mathbb{R}_{\geq 0} \cup \Sigma_A$. The Z-run ρ *accepts* the timed word $\pi_0\pi_1\pi_2\cdots$. The Z-run ρ is *divergent* if $\sum\{\pi_i \mid i \in \mathbb{N} \text{ and } \pi_i \in \mathbb{R}_{\geq 0}\} = \infty$. The *$\omega$-language of A from Z*, denoted $Lang_A(Z)$, is the set of all timed words that are accepted by divergent Z-runs of A.[1] An $Init_A$-run of A is called a *run* of A. The *ω-language* of A, denoted $Lang(A)$, is $Lang_A(Init_A)$.

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, and sometimes with its invariant. For example, if $act(v)(a) = [3,5]$, $act(v)(b) = [4,4]$, $inv(v)(a) = (1,7]$, and $inv(v)(b) = (-\infty, 0]$, we write "$\dot{a} \in [3,5]$", "$\dot{b} = 4$", "$1 < a \leq 7$", and "$b \leq 0$" inside of v. Often however, we give the invariant in the text and omit it from the figure. 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

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

then $pre(\psi)(a) = (-\infty, 5]$, $pre(\psi)(b) = [4,4]$, $pre(\psi)(c) = (-\infty, \infty)$, $upd(\psi) = \{b, c\}$, $post(\psi)(a) = (-\infty, \infty)$, $post(\psi)(b) = [7,7]$, 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 2 shows a sample trajectory of $\hat{A}$ from the initial zone $\hat{Z} = \{(v_1, (0,1))\}$. 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 accepted by a $\hat{Z}$-run of the automaton $\hat{A}$ is $(4\sigma_1 1\sigma_2 1\sigma_3 1\sigma_4)^{\omega}$, with the corresponding state sequence

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

[1] The authors have argued elsewhere [HKWT95] that time divergence is a suitable acceptance condition for ω-automata.

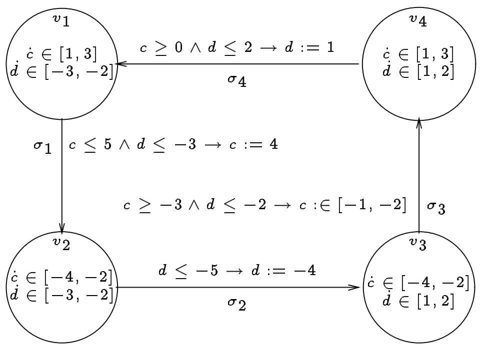

Figure 1: The initialized rectangular automaton $\hat{A}$

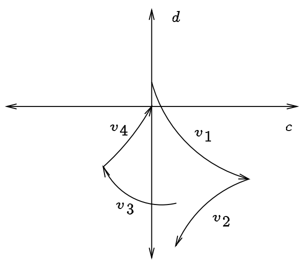

Figure 2: A sample trajectory of $\hat{A}$

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

$$[(x_1 < k \rightarrow x_1 := k) \vee (k \leq x_1 \leq k')] \wedge [(x_2 > k' \rightarrow x_2 := k') \vee (k \leq x_2 \leq k')].$$

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. $x_1 < k \wedge x_2 > k' \rightarrow x_1 := k;\ x_2 := k'$,
2. $x_1 < k \wedge k \leq x_2 \leq k' \rightarrow x_1 := k$,
3. $k \leq x_1 \leq k' \wedge x_2 > k' \rightarrow x_2 := k'$,
4. $k \leq x_1 \leq k' \wedge k \leq x_2 \leq k'$.

In this way, an n-dimensional rectangular automaton may be specified by a set of vertices, an observation alphabet, initial, invariant, and activity functions, and a set of CNF edge families. If Z is a zone of the rectangular automaton A, and Ψ is a CNF edge family, we define $Post_A^{\Psi}(Z)$ to be $\bigcup_e Post_A^e(Z)$, where the union is taken over all edges e of A that correspond to the edge family Ψ.

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, the reverse automaton $-A$ 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 2.2 follows immediately from the definitions.

Proposition 2.2 *For every rectangular automaton A, all states $q, q' \in Q_A$, and every label $\pi \in \mathbb{R}_{\geq 0} \cup E_A$, $q \xrightarrow{\pi} q'$ in A iff $q' \xrightarrow{\pi} q$ in $-A$.*

It follows that for every zone Z of A, $Pre_A(Z) = Post_{-A}(Z)$ and $Post_A(Z) = Pre_{-A}(Z)$.

Two problems concerning rectangular automata

We study the following two problems about rectangular automata.

Reachability. Given a rectangular automaton A, and a rectangular zone $Z \subset Q_A$, is $Z \cap Reach_A$ nonempty? That is, is some state in Z reachable? A solution to this problem permits the verification of safety requirements for systems that are modeled as rectangular automata. If we equip rectangular automata with final zones, then the reachability problem is equivalent to the finitary language emptiness problem.

ω-language emptiness. Given a rectangular automaton A, is $Lang(A)$ nonempty? That is, does A have a divergent run? This problem is more general than the reachability problem, and a solution permits the verification of safety and liveness requirements for systems that are modeled as rectangular automata.

For initialized rectangular automata, we provide a PSPACE decision procedure for the reachability problem. For initialized rectangular automata with bounded nondeterminism, we give a PSPACE decision procedure for the ω-language emptiness problem. We then show that the reachability problem (and therefore ω-language emptiness) 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 a given initialized rectangular automaton A into a timed automaton [AD94] that contains all reachability information about A. The translation proceeds in two steps: from initialized rectangular automata to initialized multirate automata (Section 3.2), and from initialized multirate automata to timed automata (Section 3.1). For the subclass of automata with bounded nondeterminism, the translation also preserves ω-languages (Section 3.3), and therefore reduces the ω-language emptiness problem for these automata to the corresponding problem for timed automata, which is well understood. In Section 3.4, we explain our translations in terms of simulations and bisimulations of the underlying labeled transition systems. In Section 3.5, we supply practical implications of our translations, showing that the model checker HyTech terminates on initialized rectangular automata after a linear preprocessing step.

3.1 From Initialized Multirate Automata To Timed Automata

We define several types of variables and several subclasses of rectangular automata. The variable c is a *one-slope variable* if there exists a rational number k such that $act(v)(c) = [k, k]$ for all $v \in V$. A one-slope variable with slope 0 is called a *memory cell*. A one-slope variable with slope 1 is called a *clock*. A one-slope variable with any other slope is called a *skewed clock*. Notice if every variable of the rectangular automaton A is a one-slope variable, then A is initialized. The variable c is a *two-slope variable* if there exists rational numbers $k_1 \neq k_2$ such that for each vertex v, either $act(v)(c) = [k_1, k_1]$ or $act(v)(c) = [k_2, k_2]$. A *stopwatch* is a two-slope variable with $k_1 = 1$ and $k_2 = 0$. The variable c is a *multirate variable* if for every vertex v, $act(v)(c)$ is a singleton.

The rectangular automaton A has *deterministic updates* if (1) the initial zone $Init_A$ is finite, and (2) for every edge $e \in E_A$, and every $1 \leq i \leq n$, if $i \in upd(e)$, then $post(e)_i$ is a singleton. The second requirement says that along each edge step, each variable either remains unchanged or is deterministically assigned a new value. A *timed automaton* is a rectangular automaton with deterministic updates such that every variable is a clock. The reachability and ω-language emptiness problems for timed automata (with or without silent moves) are PSPACE-complete [AD94]. More precisely, the ω-language emptiness problem for an n-dimensional timed automaton T with silent moves can be solved in space $O(\log(n!|V_T|k^n))$, where k is determined by the rational constants used in the definition of T [Alu91]. If T uses only nonnegative integer constants, then k is the largest constant appearing in the definition of T. Otherwise, let K_T be the set of all finite endpoints of intervals appearing in T, and let d be the least common denominator of the elements of K_T. Then $k = \max\{|cd| : c \in K_T\}$. The reachability of a zone Z can be solved in the same amount of space, where the constant k takes into account the endpoints of the intervals of Z as well as those in the definition of T. We consider generalizations of timed automata, and so all of our PSPACE-hardness results follow from the corresponding results for timed automata.

A *stopwatch automaton* is a rectangular automaton with deterministic updates such that every variable is a stopwatch. We later show that if even one of the variables of a stopwatch automaton is not a clock, then the reachability problem is undecidable. If we require initialization, however, then stopwatch automata are no more powerful than timed automata. This is because whenever a stopwatch is stopped or started, it is reinitialized to a new value. Such stopwatches cannot be used to accumulate delays; for example, it is not possible to record the amount of time spent in a particular vertex during the course of a computation. It follows that a stopwatch z in a timed automaton T can be replaced by a clock, if the vertex set is enlarged. When z has slope 0 in T, its value is determined uniquely by the edge by which it was stopped. So by adding one bit for each stopwatch telling if it has slope 0 or 1, and a function mapping each stopwatch to a value to be used if it has slope 0, an initialized stopwatch automaton can be transformed into a timed automaton with the same behavior.

Proposition 3.1 *The reachability and ω-language emptiness problems for initialized stopwatch automata with silent moves are PSPACE-complete.*

Proof. We translate a given n-dimensional initialized stopwatch automata automaton S into a timed automaton T_S of the same dimension and using the same rational constants, but with a vertex set of size $|V_S|(k+1)^n$. This does not affect the complexity of either the reachability algorithm or ω-language emptiness algorithm, because $O(\log(n!|V_S|(k+1)^n k^n))$ is polynomial in the size of S.

Let $K_\perp = K_S \cup \{\perp\}$. The vertex set V_{T_S} is $V_S \times (K_\perp)^{\{1,\ldots,n\}}$. So a vertex of T_S is of the form (v, f), where $f : \{1, \ldots, n\} \to K_\perp$. If $f(i) = \perp$, then $\dot{x}_i = 1$ in S. If $f(i) \neq \perp$, then $\dot{x}_i = 0$ in S, and moreover the last time x_i was assigned a value, that value was $f(i)$. It is a simple matter of coding to translate the preguard and postguard of each edge $e = (v, w)$ of S into a preguard and postguard for edges from (v, f) to (w, f') for each f, f', in such a way that T_S is timed bisimilar to S (see Section 3.4). ■

Stopwatch translation of initialized multirate automata

A *multirate automaton* is a rectangular automaton with deterministic updates such that all variables are multirate variables. We reduce problems for initialized multirate automata to problems for timed automata, by translating a given initialized multirate automaton M into an initialized stopwatch automaton S_M such that M and S_M are timed bisimilar (see Section 3.4 for a formal definition of timed bisimulation).

Let M be an n-dimensional initialized multirate automaton with silent moves. 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 n-dimensional timed automaton S_M with silent moves are those of M, except for the following: for each $v \in V$, and all i, $init_{S_M}(v) = \alpha_v(init_M(v))$, $inv_{S_M}(v) = \alpha_v(inv_M(v))$, and $act_{S_M}(v)_i = [l_i, l_i]$, where $l_i = 0$ if $k_i = 0$, and $l_i = 1$ if $k_i \neq 0$; and for each edge $e = (v, w)$, $pre_{S_M}(e) = \alpha_v(pre_M(e))$ and $post_{S_M}(e) = \alpha_w(post_M(e))$. Define $\alpha_M : Q_M \to Q_{S_M}$ such that $\alpha_M(v, \mathbf{x}) = (v, \alpha_v(\mathbf{x}))$. The map α_M is extended to zones in the natural way.

The next lemma, and the ensuing theorem, follow immediately from the definitions.

Lemma 3.2 *Let M be an initialized multirate automaton with silent moves. Then $\alpha_M(Init_M) = Init_{S_M}$. Moreover, 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 S_M.*

Theorem 3.3 *For every initialized multirate automaton M with silent moves, and for every zone $Z \subset Q_M$, $\alpha_M(Post^*_M(Z)) = Post^*_{S_M}(\alpha_M(Z))$, $\alpha_M(Pre^*_M(Z)) = Pre^*_{S_M}(\alpha_M(Z))$, and $Lang_M(Z) = Lang_{S_M}(\alpha_M(Z))$.*

Corollary 3.4 *For every initialized multirate automaton M with silent moves,*

$$Reach_M = \alpha^{-1}_{M_A}(Reach_{S_M}) \text{ and } Lang(M) = Lang(S_M).$$

Corollary 3.5 [ACHH93, NOSY93] *The reachability and ω-language emptiness problems for initialized multirate automata with silent moves are PSPACE-complete.*

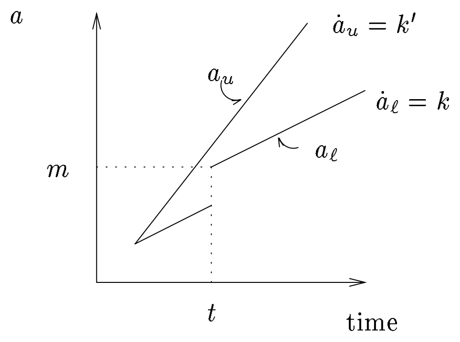

Figure 3: Envelope of the activity $act(v)(a) = [k, k']$

3.2 From Initialized Rectangular Automata to Initialized Multirate 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 a $2n$-dimensional initialized multirate automaton M_A with silent moves such that M_A contains all reachability information about A. We first give a simplified construction for the compact case, and then proceed to the general case. All of the main ideas of the construction are already present in the compact case; the general case requires a lot of additional bookkeeping. The automata A and M_A are not timed bisimilar, but A timed backward simulates M_A, and M_A timed forward simulates A (see Section 3.4).

The compact case

Let A be an n-dimensional initialized rectangular automaton. To put across the main ideas of the translation, we first restrict our attention to the case where inv_A is the trivial invariant $\lambda v \in V.\, \mathbb{R}^n$, and all values of $init_A$, act_A, pre_A, and $post_A$ are compact. In this case, we say A is *compact*. The generalization to arbitrary initialized rectangular automata is given later. Without loss of generality, we assume for the remainder of Section 3.2 that for each edge $e = (v, w)$ of A, $pre(e) \subset inv(v)$, $post(e) \subset inv(w)$, and for each $i \notin upd_A(e)$, $pre_A(e)_i = post_A(e)_i$. If this is not the case, then we replace each guard with its intersection with the appropriate invariant, and then replace each $pre_A(e)_i$ and $post_A(e)_i$ with $i \notin upd_A(e)$ by their intersection $pre_A(e)_i \cap post_A(e)_i$. In the compact case, we transform A into an initialized multirate automaton N_A with silent moves.

The idea is to replace each variable a of A with two multirate variables a_ℓ and a_u such that when $act_A(v)(a) = [k, k']$, then $act_{N_A}(v)(a_\ell) = [k, k]$ and $act_{N_A}(v)(a_u) = [k', k']$. Consider Figure 3. With each time step, the activity of a creates an envelope, whose boundaries are tracked by a_ℓ and a_u. With each edge step, the values of a_ℓ and a_u are updated so that the interval $[a_\ell, a_u]$ is precisely the range of possible values of a. In Figure 3, 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 N_A with silent moves, the variables $y_{\ell(i)}$ and $y_{u(i)}$ correspond respectively to the lower and upper bound multirate variables for variable x_i of A. For concreteness, put $\ell(i) = 2i - 1$ and $u(i) = 2i$.

The multirate automaton N_A has dimension $2n$. It has the same observation alphabet as A and the same vertex set. The initial function $init_{N_A}$ is given by $init_{N_A}(v)_{\ell(i)} = \min init_A(v)_i$ and

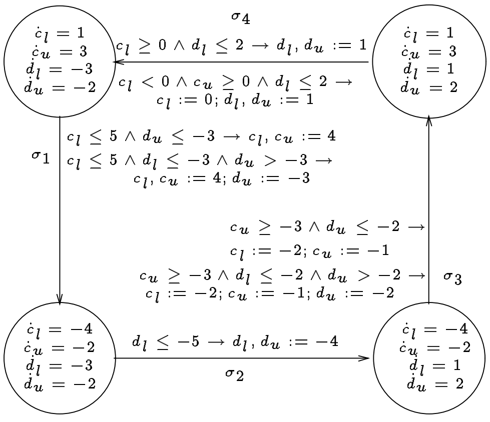

Figure 4: The initialized multirate automaton $N_{\hat{A}}$

$init_{N_A}(v)_{u(i)} = \max init_A(v)_i$. The invariant function inv_{N_A} is the trivial invariant. The activity function act_{N_A} is defined as outlined above, if $act_A(v)_i = [k, k']$, then $act_{N_A}(v)_{\ell(i)} = [k, k]$, and $act_{N_A}(v)_{u(i)} = [k', k']$.

We are left with defining a set of CNF edge families for N_A. For each edge $e = (v, w)$ of A, the multirate automaton N_A has the CNF edge family $\Psi_e = (v, w, obs_A(e), \psi_e)$, which 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

$$y_{\ell(i)} \leq k' \wedge y_{u(i)} \geq k \ \rightarrow \ y_{\ell(i)} := m; \ y_{u(i)} := m'.$$

The values of $y_{\ell(i)}$ and $y_{u(i)}$ satisfy the guard iff $[y_{\ell(i)}, y_{u(i)}]$ 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 $y_{\ell(i)}$ is set to the minimum of this interval, and $y_{u(i)}$ 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

$$[(y_{\ell(i)} < k \ \rightarrow \ y_{\ell(i)} := k) \vee (k \leq y_{\ell(i)} \leq k')] \wedge [(y_{u(i)} > k' \ \rightarrow \ y_{u(i)} := k') \vee (k \leq y_{u(i)} \leq k')].$$

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 $y_{\ell(i)} < k$, it must be updated to k, and if $y_{u(i)} > k'$, it must be updated to k', in order to keep $[y_{\ell(i)}, y_{u(i)}]$ in M_A the range of possible values of x_i in A.

This completes the definition of the multirate automaton N_A for the compact case. The multirate automaton N_A is initialized, and has $4^{n-|upd(e)|}$ edges for each edge e of A. Figure 4 gives the initialized multirate automaton $N_{\hat{A}}$ corresponding to the initialized rectangular automaton $\hat{A}$ of Figure 1. Figure 5 shows the timed automaton $T_{N_{\hat{A}}}$ corresponding to $N_{\hat{A}}$.

We introduce the map $\xi_A : Q_{N_A} \rightarrow 2^{Q_A}$ by $\xi_A(v, \mathbf{y}) = \{v\} \times \prod_{i=1}^n [y_{\ell(i)}, y_{u(i)}]$. This map formalizes the relationship illustrated in Figure 3. The map ξ_A is extended to zones by $\xi_A(Z) = \bigcup_{q \in Z} \xi_A(q)$. The *upper half-space* U_{N_A} of N_A is the zone of all states $(v, \mathbf{x}) \in Q_{M_A}$ such that $y_{\ell(i)} \leq y_{u(i)}$ for all $1 \leq i \leq n$, i.e., $U_{N_A} = \{q \in Q_{N_A} \mid \xi_A(q) \neq \emptyset\}$. Our first lemma shows that the initial zone of N_A maps to the set of initial states of A.

Lemma 3.6 *For every compact initialized rectangular automaton A, $\xi_A(Init_{N_A}) = Init_A$.*

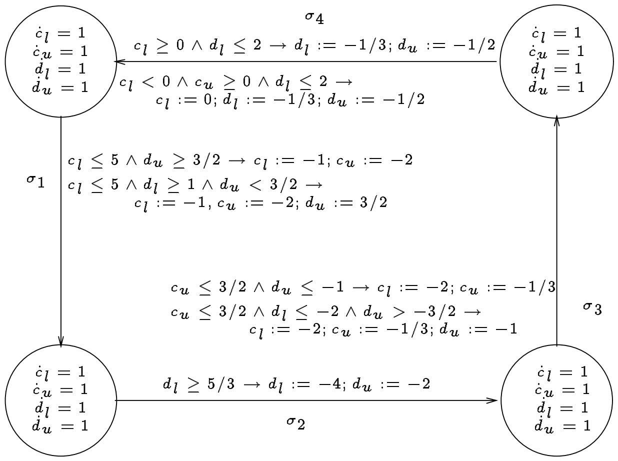

Figure 5: The initialized stopwatch automaton (and timed automaton) $S_{N_{\hat{A}}}$

The following lemma follows immediately from the proof of Lemma 2.1. It states that ξ_A commutes with time steps.

Lemma 3.7 *Let A be a compact initialized rectangular automaton. Then for every state $q \in U_{N_A}$, and every duration $t \in \mathbb{R}_{\geq 0}$,*

$$Post^t_A(\xi_A(q)) = \xi_A(Post^t_{N_A}(q)).$$

We can now prove the analogue of Lemma 3.7 for edge steps.

Lemma 3.8 *Let A be a compact initialized rectangular automaton. Then for every state $q \in U_{N_A}$, and every edge $e \in E_A$,*

$$Post^e_A(\xi_A(q)) = \xi_A(Post^{\Psi_e}_{N_A}(q)).$$

It follows that reachability problems in A are reducible to reachability problems in N_A. The next theorem follows immediately from the two lemmas.

Theorem 3.9 *Let A be a compact initialized rectangular automaton. For every zone $Z \subset U_{N_A}$, $Post^*_A(\xi_A(Z)) = \xi_A(Post^*_{N_A}(Z))$ and $Pre^*_A(\xi_A(Z)) = \xi_{-A}(Pre^*_{-(N_{-A})}(Z))$.*

Corollary 3.10 *For every compact initialized rectangular automaton A, $Reach_A = \xi_A(Reach_{M_A})$.*

It follows from Corollary 3.10 that the reachability problem for compact initialized rectangular automata is PSPACE-complete.

The general case

All of the main ideas are already present in the construction for compact automata. The extension to the general case is mostly a matter of bookkeeping. In particular, for each lower or upper bound multirate variable, one bit is used to distinguish a strict from a non-strict bound, and another bit is used to distinguish a finite from an infinite bound. The reader who is uninterested in the details can skip ahead to Theorem 3.20 without loss of continuity (in this case, the reader should know that M_A is the analogue of N_A, β_A is the analogue of ξ_A, Lemma 3.19 is the analogue of Lemma 3.7, and Lemma 3.16 is the analogue of Lemma 3.8).

We encounter the following difficulties when the rectangular automaton A has non-compact components and a nontrivial invariant. (1) For each variable x, the lower and upper bound variables x_ℓ and x_u may violate the invariant of a vertex, and so the mapping function ξ_A must be changed to include only states of A. (2) The lower and upper bounds represented by x_ℓ and x_u may be strict or weak (non-strict). To solve this problem, we introduce a bit called the *weak/strict bit* for each multirate variable. (3) Upper and lower bounds may be finite or infinite. For this, we introduce a bit called the *finite/infinite bit* for each multirate variable. (4) Unbounded activities cause discontinuous jumps in upper and lower bounds. For example suppose the variable x is assigned the value 3 by traversal of edge $e = (v, w)$, where $act(w)(x) = [1, \infty)$. Then in M_A, both $y_{\ell(i)}$ and $y_{u(i)}$ are assigned the value 3 by traversal of edge e. But after any positive amount of time passes, the upper bound on x_i should be ∞. We introduce a τ-edge called a *jump edge* which is taken before any positive time step. In this example, the jump edge sets the finite/infinit bit for $y_{u(i)}$ to *inf*. Since the result of the jump edge presupposes that some positive amount of time has passed, all edge transitions inherited from A are disabled until time passes. Implementing this restriction requires a new clock z and a bit called the *time passage bit*. (5) Strict activities can cause a weak bound to change to strict after any positive amount of time passes. For example, suppose in the above case that $act(w)(x) = [1, 5)$. Then after edge e is traversed, the upper bound on x_i is a non-strict bound of 3. But after $t > 0$ time units pass, the upper bound is a strict bound of $3 + 5t$. Once again, we use the jump edge to solve this problem. When the jump edge is traversed, the weak/strict bit for $y_{u(i)}$ is set to *str*.

The weak/strict and finite/infinite bitvectors are encoded in the discrete state. We now proceed formally to define the $(2n + 1)$-dimensional initialized multirate automaton M_A with silent moves corresponding to the initialized rectangular automaton A. We first define the continuous variables of M_A, then the vertex set, and then an analogue to ξ_A. Next the activity and invariant are defined. Then come jump edges and edges inherited from A. Last, we define the initial zone. We provide lemmas about M_A as soon as enough definitions have been made to give the proofs.

For each variable x_i of A, the multirate automaton M_A has multirate variables $y_{\ell(i)}$ and $y_{u(i)}$ corresponding respectively to the lower bound and upper bound on x_i. We add a zeroth coordinate to M_A, so that the synchronization clock z is given by y_0 in M_A. The observation alphabet of M_A is that of A. The vertex set V_{M_A} is $V_A \times (\{0,1\}^{2n})^2 \times \{0,1\}$. A state of M_A is a tuple of the form $(v, \vec{\lambda}, \vec{\nu}, tp, \mathbf{y})$, where $\vec{\lambda}$ is the vector of finite/infinite bits (*fin* $= 0$, *inf* $= 1$), $\vec{\nu}$ is the vector of weak/strict bits bits (*wk* $= 0$, *str* $= 1$), *tp* is the time passage bit, and $\mathbf{y} \in \mathbb{R}^{2n+1}$ is the continuous state.

Relating the state spaces of M_A and A. We define a map $\eta_A : Q_{M_A} \to 2^{V_A \times \mathbb{R}^n}$, which specifies how the variables of M_A give the range of possible values for the variables of A. Let $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y}) \in Q_{M_A}$. The set $\eta_A(q)$ is a rectangular region, so each component $\eta_A(q)_i$ is an interval, and hence is completely specified by its infimum, supremum, and which, if any, of the latter two it contains.

- If the finite/infinite bit $\lambda_{\ell(i)} = inf$, then $\inf \eta_A(q)_i = -\infty$.
- If the finite/infinite bit $\lambda_{\ell(i)} = fin$, then $\inf \eta_A(q)_i = y_{\ell(i)}$.
- The weak/strict bit $\nu_{\ell(i)}$ determines the strictness of the lower bound: if $\lambda_{\ell(i)} = fin$, then $\inf \eta_A(q) \in \eta_A(q)$ iff $\nu_{\ell(i)} = wk$.

We make the corresponding definitions for the upper bound.

- If the finite/infinite bit $\lambda_{u(i)} = \mathit{inf}$, then $\sup \eta_A(q)_i = \infty$.
- If the finite/infinite bit $\lambda_{u(i)} = \mathit{fin}$, then $\sup \eta_A(q)_i = y_{u(i)}$.
- The weak/strict bit $\nu_{u(i)}$ determines the strictness of the lower bound: if $\lambda_{u(i)} = \mathit{fin}$, then $\sup \eta_A(q) \in \eta_A(q)$ iff $\nu_{u(i)} = \mathit{wk}$.

Since the lower (resp. upper) bound multirate variables do not respect the lower (resp. upper) bounds of the invariant of A, $\eta_A(q) \not\subset Q_A$ for some states q of M_A. We remedy this deficiency by introducing a map $\beta_A : Q_{M_A} \to 2^{Q_A}$ defined by $\beta_A(q) = \eta_A(q) \cap Q_A$, except if $y_0 = 0$ and $\mathit{tp} = 1$, when $\beta_A(q) = \emptyset$. We make the latter adjustment because the states in which $y_0 = 0$ and $\mathit{tp} = 1$ are only reached transiently after a jump edge before a positive time step—no edge inherited from A can be traversed from such a state. The *upper half-space* U_{M_A} of M_A is the zone of all states $q \in Q_{M_A}$ such that $\beta_A(q) \neq \emptyset$. We extend η_A and β_A to functions from $2^{Q_{M_A}}$ to 2^{Q_A} by $\eta_A(Z) = \bigcup_{q \in Z} \eta_A(q)$, and similarly for β_A. The truncation of η_A to β_A is justified by the following lemma, which follows from the assumption that the preguard of every edge is contained in the invariant of the source vertex.

Lemma 3.11 *For every edge e of A, and every state $q \in U_{M_A}$, $\eta_A(q) \cap \mathit{pre}_A(e) \neq \emptyset$ iff $\beta_A(q) \cap \mathit{pre}_A(e) \neq \emptyset$.*

For the rest of this section, i ranges over $\{1, \ldots, n\}$, j ranges over $\{1, \ldots, 2n\}$, v ranges over V_A, $\vec{\lambda}$ and $\vec{\nu}$ range over $\{0,1\}^{2n}$, and tp ranges over $\{0,1\}$. So when we quantify these variables, as in "for all $v, \vec{\lambda}, \vec{\nu}, \mathit{tp}, i, j$", the quantification is over the domain just specified for each variable.

Activity and invariant of M_A. Let I be an interval. Define the *lower strictness* of I by

$$\mathit{strict}{\downarrow} I = \begin{cases} \mathit{wk}, & \text{if } \inf I \in I, \\ \mathit{str}, & \text{if } \inf I \notin I. \end{cases}$$

Define the *upper strictness* of I by

$$\mathit{strict}{\uparrow} I = \begin{cases} \mathit{wk}, & \text{if } \sup I \in I, \\ \mathit{str}, & \text{if } \sup I \notin I. \end{cases}$$

We now define the invariant function inv_{M_A}. Let $v, \vec{\lambda}, \vec{\nu}, \mathit{tp}, i$ be given. Then

$$\mathit{inv}_{M_A}(v, \vec{\lambda}, \vec{\nu}, \mathit{tp})_{\ell(i)} = \begin{cases} (-\infty, \infty), & \text{if } \lambda_{\ell(i)} = \mathit{inf}, \\ (-\infty, \sup \mathit{inv}_A(v)_i], & \text{if } \lambda_{\ell(i)} = \mathit{fin} \text{ and } \nu_{\ell(i)} = \mathit{strict}{\uparrow}\, \mathit{inv}_A(v)_i = \mathit{wk}, \\ (-\infty, \sup \mathit{inv}_A(v)_i), & \text{otherwise.} \end{cases}$$

If the finite/infinite bit $\lambda_{\ell(i)}$ is infinite, then the value of lower bound multirate variable $y_{\ell(i)}$ is irrelevant, so we do not constrain it. If the finite/infinite bit is finite, then the upper bound on this interval is strict unless both the strictness $\nu_{\ell(i)}$ of the lower bound multirate variable $y_{\ell(i)}$ and the upper strictness of the invariant are both weak. The motivation for this is that if I and J are intervals, and $\inf I = \sup J$, then $I \cap J \neq \emptyset$ iff $\mathit{strict}{\downarrow}\, I = \mathit{strict}{\uparrow}\, J = \mathit{wk}$. Here I is meant to represent the range of allowable values for x_i in A, as determined by the state of M_A, and J is meant to represent an invariant $\mathit{inv}_A(v)_i$. We have the corresponding definitions for the upper bounds. Define

$$\mathit{inv}_{M_A}(v, \vec{\lambda}, \vec{\nu}, \mathit{tp})_{u(i)} = \begin{cases} (-\infty, \infty), & \text{if } \lambda_{u(i)} = \mathit{inf}, \\ [\inf \mathit{inv}_A(v)_i, \infty), & \text{if } \lambda_{u(i)} = \mathit{fin} \text{ and } \nu_{u(i)} = \mathit{strict}{\downarrow}\, \mathit{inv}_A(v)_i = \mathit{wk}, \\ (\inf \mathit{inv}_A(v)_i, \infty), & \text{otherwise.} \end{cases}$$

The invariant of the synchronization clock is defined so that time may not pass if the time passage bit tp is 0: $inv_{M_A}(v, \vec{\lambda}, \vec{\nu}, 0)_0 = \{0\}$ and $inv_{M_A}(v, \vec{\lambda}, \vec{\nu}, 1)_0 = [0, \infty)$.

We now define the activity function act_{M_A}. For all $v, \vec{\lambda}, \vec{\nu}, tp, i$,

$$act_{M_A}(v, \vec{\lambda}, \vec{\nu}, tp)_{\ell(i)} = \begin{cases} \{\inf act_A(v)_i\}, & \text{if } \inf act_A(v)_i > -\infty, \\ \{0\}, & \text{if } \inf act_A(v)_i = -\infty. \end{cases}$$

$$act_{M_A}(v, \vec{\lambda}, \vec{\nu}, tp)_{u(i)} = \begin{cases} \{\sup act_A(v)_i\}, & \text{if } \sup act_A(v)_i < \infty, \\ \{0\}, & \text{if } \sup act_A(v)_i = \infty. \end{cases}$$

The slope of $y_{\ell(i)}$ in M_A is the infimum of the allowable slopes for x_i in A, unless that infimum is infinite. The synchronization variable is a clock: $act_{M_A}(v, \vec{\lambda}, \vec{\nu}, tp)_0 = \{1\}$.

It remains to define the edges of M_A. As in the compact case, the multirate automaton M_A has an edge family for each edge of A. We say that the edges defined by these edge families are *inherited from A*. In addition, the multirate automaton M_A has a set of τ-edges called *jump edges*. The jump edges provide changes to bound value and strictness that are caused by the passage of any positive amount of time. For example, an unbounded activity always causes a discontinuous jump in the bound value. Such a jump can only be simulated by an edge transition.

Jump edges. We proceed to define the jump edges. For all $v, \vec{\lambda}, \vec{\nu}$, there is an edge from $(v, \vec{\lambda}, \vec{\nu}, 0)$ to $(v, \vec{\lambda}', \vec{\nu}', 1)$ with observation τ. The target vertex components $\vec{\lambda}', \vec{\nu}'$ will be defined presently. The preguard and postguard are both $\{0\} \times \mathbb{R}^{2n}$, and so this edge can only be traversed when $y_0 = 0$. The update set is empty. Recall that jump edges provide changes that become necessary if any positive amount of time passes. These edges are taken proactively, before any time passes: i.e., only if the synchronization clock y_0 has value 0. To prevent spurious edges from being taken due to the changes made by these edges, no edge inherited from A may be traversed until a positive amount of time has passed, i.e., until $y_0 > 0$.

The finite/infinite bitvector must be changed to account for finite bounds that become infinite due to an unbounded activity.

$$\lambda'_{\ell(i)} = \begin{cases} inf, & \text{if } \inf act_A(v)_i = -\infty, \\ \lambda_{\ell(i)}, & \text{otherwise.} \end{cases}$$

$$\lambda'_{u(i)} = \begin{cases} inf, & \text{if } \sup act_A(v)_i = \infty, \\ \lambda_{u(i)}, & \text{otherwise.} \end{cases}$$

The weak/strict bitvector must be changed to account for weak bounds that become strict due to a strict activity.

$$\nu'_{\ell(i)} = \begin{cases} str, & \text{if } strict\!\downarrow act_A(v)_i = str, \\ \nu_{\ell(i)}, & \text{otherwise.} \end{cases}$$

$$\nu'_{u(i)} = \begin{cases} str, & \text{if } strict\!\uparrow act_A(v)_i = str, \\ \nu_{u(i)}, & \text{otherwise.} \end{cases}$$

The jump edges, just defined, play the following role in M_A. Suppose an edge inherited from A is taken. Then $tp = 0$. If no edge inherited from A is traversed, then before any time may pass (since no time may pass when $tp = 0$), a jump edge is traversed, setting tp to 1, and performing whatever bookkeeping is required for the weak/strict and finite/infinite bitvectors. The changes made by the jump edge reflect the situation after some positive amount of time has passed only. Therefore no edge inherited from A is allowed to be traversed before some time passes: if $tp = 1$ and $y_0 = 0$, then no inherited transitions are enabled.

Inherited edges. We now define the edges of M_A inherited from A. It is convenient to extended the definition of CNF edge family to allow multiple target vertices. When we use such edges, we consider the bitvectors $\vec{\lambda}$ and $\vec{\nu}$ to be discrete array variables, and so we write, for example, $\lambda_{\ell(i)} := \mathit{inf}$ to change the $\ell(i)$ component of the finite/infinite bitvector to inf. The translation of such *extended CNF edge families* into edges is a straightforward extension of the existing translation of edge families into edges, and will not be detailed. An extended CNF edge family is completely specified by a source vertex, an observation, the first component of the target vertex (an element of V_A), the time passage bit of the target vertex, and a CNF expression extended to include assignment to the discrete array variables $\vec{\lambda}$ and $\vec{\nu}$.

For each edge $e = (v, w)$ of A, all edge bitvectors $\vec{\lambda}$ and $\vec{\nu}$, and each $tp \in \{0, 1\}$, there is in M_A an extended CNF edge family $\Psi(e, \vec{\lambda}, \vec{\nu}, tp) = ((v, \vec{\lambda}, \vec{\nu}, tp), obs_A(e), w, 0, \psi(e, \vec{\lambda}, \vec{\nu}, tp))$. As in the compact case, this edge family shares the observation label of the edge e. The time passage bit tp is set to 0 along each of these edges. So every edge derived from the family has target vertex of the form $(w, \vec{\lambda}', \vec{\nu}', 0)$. The CNF expression $\psi(e, \vec{\lambda}, \vec{\nu}, tp)$ is a conjunction of CNF expressions $\theta_{tp} \wedge \bigwedge_{i=1}^{n} \psi(e, \vec{\lambda}, \vec{\nu}, tp)_i$. The guarded command θ_0 is $y_0 = 0$ and the guarded command θ_1 is $y_0 > 0 \rightarrow y_0 := 0$. Hence an edge from family $\psi(e, \vec{\lambda}, \vec{\nu}, 0)$ can be taken only if the synchronization clock has value 0, and an edge from family $\psi(e, \vec{\lambda}, \vec{\nu}, 1)$ can be taken only if the synchronization clock has value greater than 0. In the latter case the synchronization clock is reset to 0. An edge of M_A derived from the edge family $\Psi(e, \vec{\lambda}, \vec{\nu}, tp)$ may be traversed from state $q \in Q_{M_A}$ iff the range of possible values for each x_i intersects $pre(e)_i$, i.e., iff $\eta_A(q) \cap pre(e) \neq \emptyset$. It follows from the inclusion of edge preguards in source vertex invariants that $\eta_A(q) \cap pre(e) \neq \emptyset$ iff $\beta_A(q) \cap pre(e) \neq \emptyset$. If all values are finite and all bounds are weak, then the intersection is nonempty iff $y_{\ell(i)} \leq \max pre(e)_i$ and $y_{u(i)} \geq \min pre(e)_i$. This was the requirement given in the construction of N_A for compact A. Taking strictness and infinite bounds into account, we obtain the more complicated guarded commands $\ell guard(i, pre(e)_i)$ and $uguard(i, pre(e)_i)$. For an interval I, and $1 \leq i \leq n$, define

$$\ell guard(i, I) = \begin{cases} true, & \text{if } \lambda_{\ell(i)} = \mathit{inf} \\ y_{\ell(i)} \leq \sup I, & \text{if } \lambda_{\ell(i)} = \mathit{fin} \text{ and } \nu_{\ell(i)} = strict \uparrow I = wk \\ y_{\ell(i)} < \sup I, & \text{otherwise} \end{cases}$$

$$uguard(i, I) = \begin{cases} true, & \text{if } \lambda_{u(i)} = \mathit{inf} \\ y_{u(i)} \geq \inf I, & \text{if } \lambda_{u(i)} = \mathit{fin} \text{ and } \nu_{u(i)} = strict \downarrow I = wk \\ y_{u(i)} > \inf I, & \text{otherwise} \end{cases}$$

To understand this definition, consider the conditions under which an interval J intersects an interval I.

Lemma 3.12 *Let I and J be nonempty intervals, and let ϕ_ℓ and ϕ_u be defined as in Table 3.2. Then $I \cap J \neq \emptyset$ iff ϕ_ℓ and ϕ_u are true.*

The guarded command $\ell guard(i, I)$ corresponds to the predicate ϕ_ℓ and the guarded command $uguard(i, I)$ corresponds to the predicate ϕ_u. Notice ϕ_ℓ is always satisfied if $\inf J = -\infty$. Thus

$strict{\downarrow} J$	$strict{\uparrow} I$	ϕ_ℓ
wk	wk	$\inf J \leq \sup I$
wk	str	$\inf J < \sup I$
str	wk	$\inf J < \sup I$
str	str	$\inf J < \sup I$

$strict{\uparrow} J$	$strict{\downarrow} I$	ϕ_u
wk	wk	$\sup J \geq \inf I$
wk	str	$\sup J > \inf I$
str	wk	$\sup J > \inf I$
str	str	$\sup J < \inf I$

Table 1: $J \cap I \neq \emptyset$ iff $\phi_\ell \wedge \phi_u$

$\ell guard(i, I)$ is satisfied if $\lambda_{\ell(i)} = inf$. If $strict{\downarrow} J = strict{\uparrow} I = wk$, then ϕ_ℓ is $\inf J \leq \sup I$. Hence the second line of the definition of $\ell guard(i, I)$. Finally, if either $strict{\downarrow} J = str$ or $strict{\uparrow} I = str$, then ϕ_ℓ is $\inf J < \sup I$. Hence the third line of the definition of $\ell guard(i, I)$. Symmetrical remarks apply to $uguard(i, I)$ and ϕ_u.

A CNF expression for M_A with no assignments, such as θ_{tp}, $\ell guard(i, I)$, and $uguard(i, I)$, is simply a predicate over $\mathbb{R}^{2n+1}$. Therefore we say that a state $((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y})$ of M_A *satisfies* such a CNF expression ψ iff the continuous state $\mathbf{y}$ satisfies ψ regarded as a predicate over $\mathbb{R}^{2n+1}$.

Lemma 3.13 *Let e be an edge of A, and for $tp = 0, 1$, let ψ_{tp} be the CNF expression $\theta_{tp} \wedge \bigwedge_{i=1}^{n}(\ell guard(i, pre(e)_i) \wedge uguard(i, pre(e)_i))$. Then for every state $q \in U_{M_A}$, $Post_A^e(\beta_A(q)) \neq \emptyset$ iff q satisfies ψ_{tp}.*

Proof. Let $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y})$. In the discussion following Lemma 3.12, we proved that if q satisfies θ_{tp}, and $\eta_A(q) = \{v\} \times I$, then $I \cap pre_A(e) \neq \emptyset$ iff q satisfies ψ_{tp}. So we may restrict attention to those states q for which $\beta_A(q) \neq \eta_A(q)$. There are two classes of such states. First, if $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y})$ with $tp = 1$ and $y_0 = 0$, then $\beta_A(q) = \emptyset$. In this case, q does not satisfy θ_1, and so q does not satisfy ψ_1. Second, there is the class of states in which $\eta_A(q) \not\subset Q_A$, when, e.g., a lower bound multirate variable has dropped below the infimum of the invariant. In this case, the result follows from Lemma 3.11. ■

The reader may recall from the compact case that the construction for those $i \in upd_A(e)$ differs from the construction for those $i \notin upd_A(e)$. We now continue the construction for $i \in upd_A(e)$. In this case, the lower and upper bounds on x_i are assigned to the infimum and supremum of $post_A(e)_i$, with corresponding assignments made to the finite/infinite, off/on, and weak/strict bitvectors. For an interval I, and $1 \leq i \leq n$, define

$$\ell assign(i, I) = \begin{cases} true \to y_{\ell(i)} := \inf I;\ \lambda_{\ell(i)} := fin;\ \nu_{\ell(i)} := strict{\downarrow}\, post_A(e)_i & \text{if } \inf I \neq -\infty, \\ true \to y_{\ell(i)} := 0;\ \lambda_{\ell(i)} := inf;\ \nu_{\ell(i)} := str & \text{if } \inf I = -\infty. \end{cases}$$

$$uassign(i, I) = \begin{cases} true \to y_{u(i)} := \sup I;\ \lambda_{u(i)} := fin;\ \nu_{u(i)} := strict{\uparrow}\, post_A(e)_i & \text{if } \sup I \neq \infty, \\ true \to y_{u(i)} := 0;\ \lambda_{u(i)} := inf;\ \nu_{u(i)} := str & \text{if } \sup I = \infty. \end{cases}$$

The assignments to 0 above are required for M_A to be initialized. After such an assignment, the value of the multirate variable is ignored due to the finite/infinite bit set to inf.

For $i \in upd_A(e)$, the guarded command $\psi(e, \vec{\lambda}, \vec{\nu}, tp)_i$ is

$$\ell guard(i, pre_A(e)_i) \wedge uguard(i, pre_A(e)_i) \wedge \ell assign(i, post_A(e)_i) \wedge uassign(i, post_A(e)_i).$$

The remainder of the definition of $\psi(e, \vec{\lambda}, \vec{\nu}, tp)$ makes no mention of any $i \in upd(e)$. Therefore we have the following lemma.

Lemma 3.14 *For every edge e of A, every state $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y}) \in U_{M_A}$, and every $i \in upd(e)$,*

$$Post^e_A(\beta_A(q))_i = \beta_A(Post^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}_{M_A}(q))_i.$$

As before, the case of $i \notin upd(e)$ is more complicated, because the lower (resp. upper) bound is only reset if it is too small (resp. too large). Strictness also contributes some complications. Our definitions follow once again from Lemma 3.12. The extended CNF expressions $\ell adjust(i, pre_A(e)_i)$ and $uadjust(i, pre_A(e)_i)$ give the adjustments to $y_{\ell(i)}$, $y_{u(i)}$, the finite/infinite bits $\lambda_{\ell(i)}$ and $\lambda_{u(i)}$, and the weak/strict bits $\nu_{\ell(i)}$ and $\nu_{u(i)}$, for $i \notin upd_A(e)$. Let I be a nonempty interval. Define

$$\ell adjust(i, I) = \begin{cases} true & \text{if } \inf I = -\infty, \\ (\lambda_{\ell(i)} = inf \to \lambda_{\ell(i)} := fin;\ y_{\ell(i)} := \inf I;\ \nu_{\ell(i)} := strict{\downarrow} I) & \\ \vee (\lambda_{\ell(i)} = fin \wedge y_{\ell(i)} < \inf I \to y_{\ell(i)} := \inf I;\ \nu_{\ell(i)} := strict{\downarrow} I) & \\ \vee (\lambda_{\ell(i)} = fin \wedge y_{\ell(i)} = \inf I \wedge \nu_{\ell(i)} = wk \to \nu_{\ell(i)} := strict{\downarrow} I) & \\ \vee (\lambda_{\ell(i)} = fin \wedge y_{\ell(i)} = \inf I \wedge \nu_{\ell(i)} = str) & \\ \vee (\lambda_{\ell(i)} = fin \wedge y_{\ell(i)} > \inf I), & \text{if } \inf I \neq -\infty. \end{cases}$$

$$uadjust(i, I) = \begin{cases} true & \text{if } \sup I = \infty, \\ (\lambda_{u(i)} = inf \to \lambda_{u(i)} := fin;\ y_{u(i)} := \sup I;\ \nu_{u(i)} := strict{\uparrow} I) & \\ \vee (\lambda_{u(i)} = fin \wedge y_{u(i)} > \sup I \to y_{u(i)} := \sup I;\ \nu_{u(i)} := strict{\uparrow} I) & \\ \vee (\lambda_{u(i)} = fin \wedge y_{u(i)} = \sup I \wedge \nu_{u(i)} = wk \to \nu_{u(i)} := strict{\uparrow} I) & \\ \vee (\lambda_{u(i)} = fin \wedge y_{u(i)} = \sup I \wedge \nu_{u(i)} = str) & \\ \vee (\lambda_{u(i)} = fin \wedge y_{u(i)} < \sup I), & \text{if } \sup I \neq \infty. \end{cases}$$

Explanation of these definitions is deferred to the next paragraph. For $i \notin upd_A(e)_i$, the guarded command $\psi(e, \vec{\lambda}, \vec{\nu}, tp)_i$ is

$$\ell guard(i, pre_A(e)_i) \wedge \ell adjust(i, pre_A(e)_i) \wedge uguard(i, pre_A(e)_i) \wedge uadjust(i, pre_A(e)_i).$$

The conjuncts $\ell guard(i, pre_A(e)_i)$ and $uguard(i, pre_A(e)_i)$ ensure that the edge can be taken iff the interval defined by the lower and upper bounds and the finite/infinite and weak/strict bits intersects the preguard interval $pre_A(e)_i$. The conjuncts $\ell adjust(i, pre_A(e)_i)$ and $uadjust(i, pre_A(e)_i)$ reset the lower and upper bound values and their finite/infinite and weak/strict bits based upon the new information learned about the value of x_i if the edge e is traversed in A.

We now examine the definition of $\ell adjust(i, pre_A(e)_i)$. If edge e is traversed in A, then new information about the value of x_i is obtained, namely that it lies within $pre_A(e)_i$. Put $k = \inf pre_A(e)_i$. If $k = -\infty$, then there is no new information, and so $\ell adjust(i, pre_A(e)_i) = true$; hence the first line of the definition. If $k > -\infty$, then we have several cases. If the finite/infinite bit $\lambda_{\ell(i)} = inf$, then the present lower bound is infinite, and so $\lambda_{\ell(i)}$ must be set to fin, $y_{\ell(i)}$ must be reassigned to k, and the weak/strict bit $\nu_{\ell(i)}$ must be assigned to the lower strictness of $pre_A(e)_i$ (line two). Now suppose the finite/infinite bit $\lambda_{\ell(i)} = fin$. If $y_{\ell(i)} < k$, then again $y_{\ell(i)}$ and $\nu_{\ell(i)}$ must be reset to k and its strictness (line three). If $y_{\ell(i)} = k$ and the strictness bit $\nu_{\ell(i)} = wk$, then information is gained if the lower strictness of $pre_A(e)_i$ is str. So in this case (line four) we perform the assignment $\nu_{\ell(i)} := strict{\downarrow}\, pre_A(e)_i$. But if $y_\ell = k$ and the strictness bit $\nu_{\ell(i)} = str$, then no information is gained; and so no assignment is performed (line five). Finally, if $y_{\ell(i)} > k$, then there is no new information, and so there is no assignment (line 6). We have proven the following lemma.

Lemma 3.15 *For every edge e of A, every state $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y}) \in U_{M_A}$, and every $i \notin upd(e)$,*

$$Post^e_A(\beta_A(q))_i = \beta_A(Post^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}_{M_A}(q))_i.$$

Putting together Lemmas 3.13, 3.14, and 3.15, we have the following lemma.

Lemma 3.16 *Let A be an initialized rectangular automaton. For every state $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y}) \in U_{M_A}$, and every edge e of A,*

$$Post^{e}_{A}(\beta_A(q)) = \beta_A(Post^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}_{M_A}(q)).$$

Moreover, $|Post^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}_{M_A}(q)| \leq 1$. That is, $Post^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}_{M_A}(q)$ is either empty or a singleton.

Proof. The first statement follows from Lemmas 3.13, 3.14, and 3.15, and from the fact that $Post^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}_{M_A}(q) \neq \emptyset$ iff q satisfies the predicate ψ_{tp} from Lemma 3.13. For the second claim, notice that all of the assignments made in the guarded commands comprising $\psi(e, \vec{\lambda}, \vec{\nu}, tp)$ are deterministic (that is, $|post_{M_A}(\tilde{e})_j| = 1$ whenever $j \in upd_{M_A}(\tilde{e})$ for some $0 \leq j \leq 2n$ and some edge $\tilde{e}$ derived from $\Psi(e, \vec{\lambda}, \vec{\nu}, tp)$), the disjuncts of $\ell adjust(i, pre_A(e)_i)$ are mutually exclusive, as are the disjuncts of $uadjust(i, pre_A(e)_i)$. So each state q can execute at most one of the disjuncts of each of these guarded commands, and each guarded command makes only deterministic assignments. ■

Initial zone. It remains to define the initial zone $Init_{M_A}$ in such a way that $\beta_A(Init_{M_A}) = Init_A$. This is done by setting $y_{\ell(i)}$ and $y_{u(i)}$ at vertex v to be the infimum and supremum of the region of $Init_{M_A}$ associated with v. Let $Z \subset Q_A$ be a rectangular zone. Then there is the canonical decomposition $Z = \bigcup_{v \in V_A} \{v\} \times R^v$ for some regions $R^v \subset \mathbb{R}^n$. Define $lowhigh\, Z$ as follows. Each $\{v\} \times R^v$ contributes a singleton zone $\{(v, \vec{\lambda}(R^v), \vec{\nu}(R^v), 0, \mathbf{y}(R^v))\}$ to $lowhigh Z$. If $\inf R^v_i = -\infty$, then $\lambda(R^v)_{\ell(i)} = inf$, $\nu(R^v)_{\ell(i)} = str$, and $y(R^v)_{\ell(i)} = 0$. If $\inf R^v_i \neq -\infty$, then $\lambda(R^v)_{\ell(i)} = fin$, $\nu(R^v)_{\ell(i)} = strict{\downarrow}\, R^v_i$, and $y(R^v)_{\ell(i)} = \inf R^v_i$. Similarly, if $\sup R^v_i = \infty$, then $\lambda(R^v)_{u(i)} = inf$, $\nu(R^v)_{u(i)} = str$, and $y(R^v)_{u(i)} = 0$. If $\sup R^v_i \neq \infty$, then $\lambda(R^v)_{u(i)} = fin$, $\nu(R^v)_{u(i)} = strict{\uparrow}\, R^v_i$, and $y(R^v)_{u(i)} = \sup R^v_i$. Now define $Init_{M_A}$ to be $lowhigh\, Init_A$.

Lemma 3.17 *Let A be an initialized rectangular automaton. For every rectangular zone $Z \subset Q_A$, $\beta_A(lowhigh\, Z) = Z$. In particular, $\beta_A(Init_{M_A}) = Init_A$.*

This completes the definition of the multirate automaton M_A. Notice that M_A is initialized and has deterministic updates. The automaton M_A has exponentially many more vertices and edges than A. As in the translation from initialized stopwatch automata to timed automata, this exponential blowup does not adversely affect the complexity of reachability or ω-language emptiness.

Analysis of time steps. For the remainder of this section, it will be convenient to refer to the components of a state q of M_A generically as v, $\vec{\lambda}$, $\vec{\nu}$, tp, and $\mathbf{y}$. We say "$\nu_{\ell(i)}$ in q" or "$\nu_{\ell(i)}$ in q'" to distinguish components of different states. Lemma 3.16 gives the basic correspondence between edge steps in A and edge steps in M_A. We must now develop a correspondence for time steps. The next lemma simply says that every reachable state of M_A that is the target of a time-step has its finite/infinite and weak/strict bits set correctly.

Lemma 3.18 *For every reachable state $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y}) \in Reach_{M_A} \cap U_{M_A}$ with $y_0 > 0$ and $tp = 1$, and for every i,*

1. *if $\inf act_A(v)_i = -\infty$ then $\lambda_{\ell(i)} = inf$; if $\sup act_A(v)_i = \infty$ then $\lambda_{u(i)} = inf$.*
2. *if $strict{\downarrow}\, act_A(v) = str$, then $\nu_{\ell(i)} = str$; if $strict{\uparrow}\, act_A(v) = str$, then $\nu_{u(i)} = str$.*

The following lemma gives the time-step correspondence between A and M_A.

Lemma 3.19 *Let A be an initialized rectangular automaton. Then for every reachable state $q \in Reach_{M_A} \cap U_{M_A}$, and every duration $t \in \mathbb{R}_{\geq 0}$,*

$$Post_A^t(\beta_A(q)) = \beta_A(Post_{M_A}^t(q)).$$

Moreover, $|Post_{M_A}^t(q) \cap U_{M_A}| \leq 1$. That is, $Post_{M_A}^t(q) \cap U_{M_A}$ is either empty or a singleton.

Proof. The second claim is the determinism of the time-step relation. It follows from the fact that M_A is initialized and its continuous variables are multirate variables: they take on only one rate between initializations. Let $q \in U_{M_A}$ be a reachable state, and let $\beta_A(q) = \{v\} \times B$.

Case 1: $t = 0$. In this case $Post_A^t(\beta_A(q)) = \beta_A(q)$. Every $q' \in Post_{M_A}^t(q)$ is either q itself, when obviously $\beta_A(q') = \beta_A(q)$, or the target of a jump edge, when $\beta_A(q') = \emptyset$. Hence $Post_A^0(\beta_A(q)) = \beta_A(q) = \beta_A(Post_{M_A}^0(q))$.

Case 2: $t > 0$ and $Post_A^t(\beta_A(q)) \neq \emptyset$. Recall from Proposition 2.1 that each $Post_A^t(\beta_A(q))_i$ is a nonempty interval with

$$\inf Post_A^t(\beta_A(q))_i = \max\{\inf inv_A(v)_i, \inf B_i + t \cdot \inf act_A(v)_i\}, \tag{1}$$

and

$$\sup Post_A^t(\beta_A(q))_i = \min\{\sup inv_A(v)_i, \sup B_i + t \cdot \sup act_A(v)_i\}. \tag{2}$$

The strictness of the infimum is given as follows. Put $Inv = inv_A(v)_i$, $Act = act_A(v)_i$, and $Try = \inf B_i + t \cdot \inf act_A(v)_i$. If $\inf Inv > Try$, then $strict{\downarrow}\, Post_A^t(\beta_A(q)) = strict{\downarrow}\, Inv$. If $\inf Inv = Try$, then $strict{\downarrow}\, Post_A^t(\beta_A(q)) = wk$ iff $strict{\downarrow}\, Inv = strict{\downarrow}\, B_i = strict{\downarrow}\, Act = wk$. If $\inf Inv < Try$, then $strict{\downarrow}\, Post_A^t(\beta_A(q)) = wk$ iff $strict{\downarrow}\, B_i = strict{\downarrow}\, Act = wk$. The strictness of the supremum is given symmetrically.

There is exactly one state $q' \in Post_{M_A}^t(q)$. We show that $\beta_A(q') = Post_A^t(\beta_A(q))$.

Subcase 2a: $\beta_A(q)$ and $act_A(v)$ are bounded. The upper bound clock $y_{u(i)}$ moves at the supremum of the allowable rates for x_i in A. If this rate is positive then the upper bound reaches the upper boundary of $inv_A(v)_i$ after $\frac{\sup inv_A(v)_i - \sup B_i}{\sup act_A(v)_i}$ units of time pass, Hence $\sup \beta_A(q') = \min\{\sup inv_A(v)_i, \sup B_i + t \cdot \sup act_A(v)_i\}$, which is the same as $\sup Post_A^t(\beta_A(q))_i$. Similarly, $\inf \beta_A(q') = \max\{\inf inv_A(v)_i, \inf B_i + t \cdot \inf act_A(v)_i\}$, which is the same as $\inf Post_A^t(\beta_A(q))_i$. So the infimum and supremum of $\beta_A(q')_i$ coincide with the infimum and supremum of $post_A^t(\beta_A(q))_i$. The question of strictness remains. If $\sup B_i + t \cdot \sup act_A(v)_i > \sup inv_A(v)_i$, then $strict{\uparrow}\, \beta_A(q')_i = strict{\uparrow}\, inv_A(v)_i = strict{\uparrow}\, Post_A^t(\beta_A(q))$. The strictness is correct. Now suppose $\sup B_i + t \cdot \sup act_A(v)_i < \sup inv_A(v)_i$. Then in state q', the upper bound $y_{u(i)}$ has value $\sup B_i + t \cdot \sup act_A(v)_i$. Since q' is reachable, Lemma 3.18 implies that in q', $\nu_{u(i)} = wk$ iff $strict{\uparrow}\, \beta_A(q)_i = strict{\uparrow}\, act_A(v)_i = wk$. A glance at the discussion of strictness following Equation 2 shows that the strictness is correct: $strict{\uparrow}\, \beta_A(q')_i = strict{\uparrow}\, Post_A^t(\beta_A(q))$. Finally, suppose $\sup B_i + t \cdot \sup act_A(v)_i = \sup inv_A(v)_i$. In this case, again we have that in q', $\nu_{u(i)} = wk$ iff $strict{\uparrow}\, \beta_A(q)_i = strict{\uparrow}\, act_A(v)_i = wk$. But here the definition of β_A, which intersects the value of η_A with the invariant, comes to the fore, resulting in a strict bound if $strict{\uparrow}\, inv_A(v)_i = str$. Therefore $strict{\uparrow}\, \beta_A(q')_i = wk$ iff $strict{\uparrow}\, \beta_A(q)_i = strict{\uparrow}\, act_A(v)_i = strict{\uparrow}\, inv_A(v)_i = wk$. The discussion of strictness following Equation 2 shows that the strictness is correct: $strict{\uparrow}\, \beta_A(q')_i =$

strict$\uparrow$ $Post^t_A(\beta_A(q))$. Symmetrical remarks apply to the lower bound multirate variable, and we have completed the discussion of strictness.

Subcase 2b: $\beta_A(q)$ is not bounded from above. In this case, $inv_A(v)$ is not bounded from above, either, and the finite/infinite bit $\lambda_{u(i)}$ is *inf* in state q. The jump edges do not change this bit when $inv_A(v)$ is unbounded from above, and so in state q', still $\lambda_{u(i)}$ is *inf*, and so $\sup \beta_A(q')_i = \infty = \sup Post^t_A(\beta_A(q))$. Symmetrical remarks apply to the case of $\beta_A(q)$ not bounded from below.

Subcase 2c: $act_A(v)_i$ is not bounded from above. Since $t > 0$, $\lambda_{u(i)} = inf$ in q' by Lemma 3.18. So $\sup \beta_A(q')_i = \sup inv_A(v)_i = \sup Post^t_A(\beta_A(q))_i$, with matching strictnesses. A symmetrical argument handles the lower bound. We conclude that $\beta_A(q')_i = Post^t_A(\beta_A(q))_i$. The case of $t > 0$ and $Post^t_A(\beta_A(q)) \neq \emptyset$ is complete.

Case 3: $Post^t_A(\beta_A(q)) = \emptyset$. This means that for each coordinate i, either the lower bound $y_{\ell(i)}$ rises above the upper boundary of $inv_A(v)_i$ within time t, or the upper bound $y_{u(i)}$ drops below the lower boundary of $inv_A(v)_i$ within time t. Put $bottom(i) = \max\{\inf \inf_A(v)_i, \inf B_i + t \cdot \inf act_A(v)_i\}$ (see Equation 1) and $top(i) = \min\{\sup inv_A(v)_i, \sup B_i + t \cdot \sup act_A(v)_i\}$ (see Equation 2). That is, $bottom(i)$ (resp. $top(i)$) would equal $\inf Post^t_A(\beta_A(q))_i$ (resp. $\sup Post^t_A(\beta_A(q))_i$), if only $Post^t_A(\beta_A(q))_i$ were nonempty. The fact that $Post^t_A(\beta_A(q))_i = \emptyset$ means that either $bottom(i) > \sup inv_A(v)_i$ or $top(i) > \inf inv_A(v)_i$, or we have equality in one of these two expressions with a strictness conflict. If $bottom(i) > \sup inv_A(v)_i$, then q cannot take a $\xrightarrow{t}$ transition, because any state q' with $q \xrightarrow{t} q'$ has the lower bound clock $y_{\ell(i)} > \sup inv_A(v)_i$, which is impossible, since for any such q', $\sup inv_{M_A}(q')_{\ell(i)} = \sup inv_A(v)_i$ (see the definition of inv_{M_A}). So in this case $Post^t_{M_A}(q) = \emptyset$, and so $\beta_A(Post^t_{M_A}(q)) = \emptyset = Post^t_A(\beta_A(q))$ as desired. We have a similar deduction for $top(i) < \inf inv_A(v)$. Now suppose $bottom(i) = \sup inv_A(v)_i$. There are two possible strictness conflicts. (1) The invariant upper boundary is strict. (2) The lower bound multirate variable is strict. In either case, the invariant inv_{M_A} places $\sup inv_A(v)_i$ out of the reach of $y_{\ell(i)}$: the supremum of the invariant for $y_{\ell(i)}$ is $\sup inv_A(v)_i$, but the supremum is not contained in the invariant interval (the reader is encouraged to reread the second line of the definition of inv_{M_A}). The proof is complete. ■

Theorem 3.20 *For every initialized rectangular automaton A, and for every zone $Z \subset U_{M_A}$, $Post^*_A(\beta_A(Z)) = \beta_A(Post^*_{M_A}(Z))$ and $Pre^*_A(\beta_A(Z)) = \beta_{-A}(Pre^*_{-(M_{-A})}(Z))$.*

Proof. The first claim is immediate from Lemma 3.16 and 3.19. The second follows from Proposition 2.2 and the fact that $\beta_A = \beta_{-A}$:

$$Pre^*_A(\beta_A(Z)) = Post^*_{-A}(\beta_A(Z)) = \beta_A(Post^*_{M_{-A}}(Z)) = \beta_{-A}(Pre^*_{-(M_{-A})}(Z)). \blacksquare$$

Corollary 3.21 *For every initialized rectangular automaton A, $Reach_A = \beta_A(Reach_{M_A})$.*

Corollary 3.22 *The reachability problem for initialized rectangular automata is PSPACE-complete.*

Proof. Let A be an initialized rectangular automaton. The vertex reachability problem asks whether $Post^*_A(\{v\} \times inv(v)) \cap (\{w\} \times inv(w)) = \emptyset$. The general reachability problem from the initial zone *Init* to another zone Z may easily be reduced in polynomial time to the vertex reachability problem. So it suffices to show how to solve the latter in PSPACE. By Theorem 3.20, we may reduce the vertex reachability problem from v to w in A to a reachability problem in M_A from a set of vertices of the form $\{v\} \times (\{0,1\}^{2n})^2 \times \{0,1\}$ to a set of vertices of the form $\{w\} \times (\{0,1\}^{2n})^2 \times \{0,1\}$. Since the dimension of M_A is only twice the dimension of A, this can be solved in space $O(\log((2n+1)!|V_A|(k+1)^{2n+1}k^{2n+1}))$ by performing the search on $T_{S_{M_A}}$. The automaton $T_{S_{M_A}}$ need not be explicitly constructed to perform this search. ■

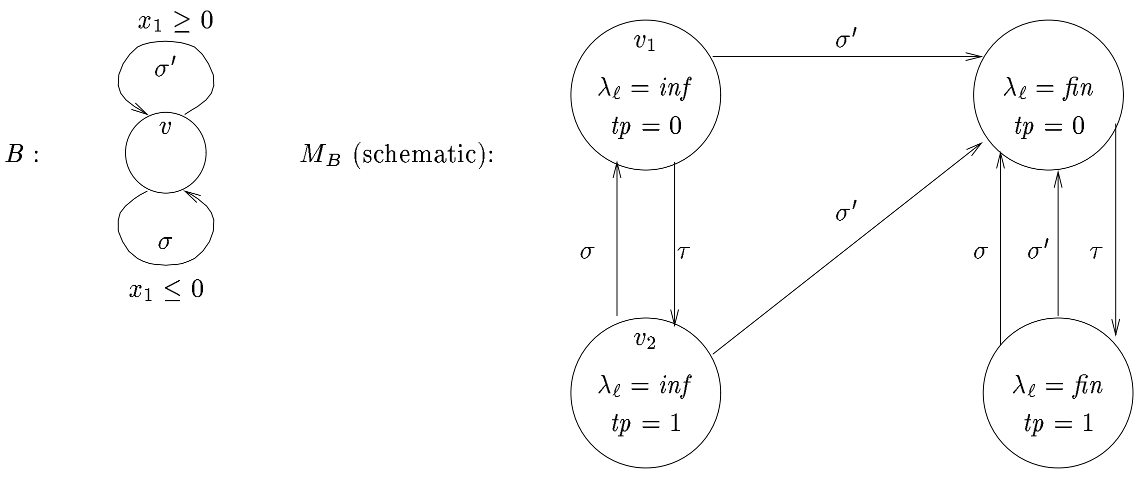

Figure 6: $Lang(B) \subsetneq Lang(M_B)$

3.3 ω-Language Emptiness

Let A be an initialized rectangular automaton. While Lemmas 3.16 and 3.19 imply that M_A and A generate the same finite timed words, the multirate automaton M_A may generate infinite timed words that are not in the ω-language of A. For example, consider the timed automaton B in Figure 6, with initial zone $Z = \{v\} \times (-\infty, 0]$. The timed word $(1\sigma)^\omega$ is not an element of $Lang(B)$. However, since Z is unbounded, in M_B the lower finite/infinite bit $\lambda_{\ell(1)}$ remains at *inf* on time steps and σ steps. Therefore $(1\sigma)^\omega$ is an element of $Lang(M_B)$. Consider the schematic picture of M_B in Figure 6, in which guarded commands, the weak/strict bits, and the finite/infinite bit for the upper bound multirate variable are suppressed. The multirate automaton M_A has the divergent run $(v_1 \xrightarrow{\tau} v_2 \overset{1}{\Rightarrow} v_2 \xrightarrow{\sigma})^\omega$, where we have suppressed the continuous state. This is due to the unboundedness of the initial zone. Similar behavior is exhibited in automata with unbounded postguards. The definition of bounded nondeterminism (see Section 2) precludes both.

We prove that if A has bounded nondeterminism, then $Lang(A) = Lang(M_A)$. The main theorem states that if A has bounded nondeterminism, then the ω-language $Lang(A)$ is limit-closed. That is, given a bounded zone Z, and given any timed word $\underline{\pi}$ such that every finite prefix of $\underline{\pi}$ is accepted by a finite Z-run, then there is an infinite Z-run accepting $\underline{\pi}$. From this it follows that $Lang(A) = Lang(M_A)$. We first give the result for A with the stronger requirement of compact nondeterminism, because the proof of the main theorem for this restricted case is a simple consequence of the fact that a decreasing sequence of nonempty compact regions has nonempty intersection. Thereafter we proceed to the general case.

Preliminary definitions. Let A be a rectangular automaton, and let $Z \subset Q_A$ be a zone. In this section, it is convenient to consider timed words over the alphabet $\mathbb{R}_{\geq 0} \cup E_A$, where the edge set replaces the observation alphabet. Each definition in this paragraph is exactly analogous to the corresponding definition for the alphabet $\mathbb{R}_{\geq 0} \cup \Sigma_A$ given in Section 2. A *timed edge word* is an infinite sequence over the alphabet $\mathbb{R}_{\geq 0} \cup E_A$. An *$Z$-edge run* ϱ of A is an infinite sequence of the form $q_0 \overset{\varpi_0}{\rightarrow} q_1 \overset{\varpi_1}{\rightarrow} q_2 \overset{\varpi_2}{\rightarrow} \cdots$, where $q_0 \in Z$, and for all i, $q_i \in Q_A$ and $\varpi_i \in \mathbb{R}_{\geq 0} \cup E_A$. The Z-edge run ϱ *accepts* the timed edge-word $\varpi_0\varpi_1\varpi_2\cdots$. Divergence of a Z-edge run is defined in the same way as for Z-runs. The *edge ω-language of A from Z*, denoted $ELang_A(Z)$, is the set of all divergent timed edge words that are accepted by Z-edge runs of A.

The case of compact nondeterminism

The first proposition gives a basic property of compact zones which is inherited from Euclidean space.

Proposition 3.23 *Let A be a rectangular automaton, and let $(Z_i)_{i\in\mathbb{N}}$ be a decreasing sequence of nonempty compact zones of A. Then the intersection $\bigcap_{i\in\mathbb{N}} Z_i$ is nonempty.*

Proof. This follows from the corresponding statement for regions (subsets of $\mathbb{R}^n$), and the fact that V_A is finite. ■

The rectangular automaton A has *compact nondeterminism* if it has bounded nondeterminism, and all rectangular regions appearing in the definition of A are closed. Formally, we say that A has *compact nondeterminism* if

- for every vertex v, $init(v)$ and $act(v)$ are compact, and $inv(v)$ is closed,
- for every edge e, $pre(e)$ and $post(e)$ are closed, and
- for every edge e, and every $i \in \{1, \ldots, n\}$, if $i \in upd(e)$, then $post(e)$ is compact.

We show that rectangular automata with compact nondeterminism define limit-closed ω-languages. The following two technical lemmas are used to establish the compactness of all zones that are used in the proof of the main theorem.

Lemma 3.24 *Let A be a rectangular automaton with compact nondeterminism. For every compact multirectangular zone $Z \subset Q_A$, and every $\varpi \in \mathbb{R}_{\geq 0} \cup E_A$, the zone $Post_A^{\varpi}(Z)$ is compact and multirectangular.*

Lemma 3.25 *Let A be a rectangular automaton with compact nondeterminism. For every pair of compact zones $Z, Z' \subset Q_A$, and every $\varpi \in \mathbb{R}_{\geq 0} \cup E_A$, the zone $Pre_A^{\varpi}(Z') \cap Z$ is compact and multirectangular.*

Note the asymmetry of the two lemmas. The intersection of $Pre_A^{\varpi}(Z')$ with the compact zone Z is required for compactness, because preguards of automata with compact nondeterminism are only required to be closed, not compact. The next lemma gives the meat of the limit-closure argument, showing that if all prefixes of a timed edge word may be generated from a given zone Z, then in fact there is an element of Z from which each prefix may be generated.

Lemma 3.26 *Let A be a rectangular automaton with compact nondeterminism, and let $Z \subset Q_A$ be a compact multirectangular zone. Suppose $\underline{\varpi} \in (\mathbb{R}_{\geq 0} \cup E_A)^{\omega}$ is a timed edge word such that for every $k \in \mathbb{N}$, $Post_A^{\varpi_0\varpi_1\cdots\varpi_k}(Z) \neq \emptyset$. Then there is a state $q \in Z$ such that for every $k \in \mathbb{N}$, $Post_A^{\varpi_0\varpi_1\cdots\varpi_k}(\{q\}) \neq \emptyset$.*

Proof. For each $k \in \mathbb{N}$, define $J_k = \{q \in Z \mid Post_A^{\varpi_0\varpi_1\cdots\varpi_k}(\{q\}) \neq \emptyset\}$. Since each $Post_A^{\varpi_0\varpi_1\cdots\varpi_k}(Z)$ is nonempty, each J_k is nonempty. Also, $J_k \supset J_{k+1}$ for each k. We claim each J_k is compact. If so, then the sequence (J_k) is a decreasing sequence of nonempty compact sets. Hence the intersection $\bigcap_{k=0}^{\infty} J_k$ is nonempty. An element of the intersection is the requirement of the lemma.

We now establish the claim that each J_k is compact. By Lemma 3.24, for each $k \in \mathbb{N}$, the zone $Post_A^{\varpi_0\varpi_1\cdots\varpi_k}(Z)$ is compact and multirectangular. The zone $Pre_A^{\varpi_0\varpi_1\cdots\varpi_k}(Post_A^{\varpi_0\varpi_1\cdots\varpi_k}(Z))$ is compact by Lemma 3.25. Hence $J_k = Z \cap Pre_A^{\varpi_0\varpi_1\cdots\varpi_k}(Post_A^{\varpi_0\varpi_1\cdots\varpi_k}(Z))$ is compact as well. ■

The following main theorem establishes the limit closure of $Lang_A(Z)$ for all rectangular automata A with compact nondeterminism, and all compact zones Z.

Theorem 3.27 *Let A be a rectangular automaton with compact nondeterminism, and let $Z \subset Q_A$ be a compact zone. Suppose $\underline{\varpi} \in (\mathbb{R}_{\geq 0} \cup E_A)^\omega$ is a timed edge word such that for every $k \in \mathbb{N}$, $Post_A^{\varpi_0\varpi_1\cdots\varpi_k}(Z) \neq \emptyset$. Then there is a state $q \in Z$ such that $\underline{\varpi} \in ELang_A(\{q\})$.*

Proof. Let $Z_0 = Z$. By Lemma 3.26, there is a state $q_0 \in Z_0$ such that $Post_A^{\varpi_0\varpi_1\cdots\varpi_k}(\{q_0\}) \neq \emptyset$ for each $k \geq 0$. Let $Z_1 = Post_A^{\varpi_0}(\{q_0\})$. Then Z_1 is compact and multirectangular, and for each $k \geq 1$, $Post_A^{\varpi_1\varpi_2\cdots\varpi_k}(Z_1) \neq \emptyset$. So by Lemma 3.26, there is a state $q_1 \in Z_1$ such that for each $k \geq 1$, $Post_A^{\varpi_1\varpi_2\cdots\varpi_k}(\{q_1\}) \neq \emptyset$. Proceed inductively in this manner, with $Z_{j+1} = Post_A^{\pi_j}(\{q_j\})$ compact and multirectangular, and $q_{j+1} \in Z_{j+1}$ given by Lemma 3.26, such that for each $k > j+1$, $Post_A^{\varpi_{j+1}\varpi_{j+2}\cdots\varpi_k}(\{q_{j+1}\}) \neq \emptyset$. Then

$$q_0 \xrightarrow{\varpi_0} q_1 \xrightarrow{\varpi_1} q_2 \xrightarrow{\varpi_2} \cdots$$

is a Z-edge run of A. ■

Corollary 3.28 *For every initialized rectangular hybrid automaton A with compact nondeterminism, $Lang(A) = Lang(M_A)$.*

Proof. Let $Z = Init_A$. The inclusion $Lang_A(Z) \subset Lang_{M_A}(lowhighZ)$ is immediate from Lemmas 3.16 and 3.19. For the reverse, suppose

$$q_0 \xrightarrow{\varpi_0} q_1 \xrightarrow{\varpi_1} q_2 \xrightarrow{\varpi_2} \cdots$$

is an *lowhigh*Z-edge run of M_A. Then there exist states $q'_k \in U_{M_A}$, $k = 0, 1, 2 \ldots$, in the upper half space of M_A such that

$$q'_0 \xrightarrow{\varpi_0} q'_1 \xrightarrow{\varpi_1} q'_2 \xrightarrow{\varpi_2} \cdots$$

is an *lowhigh*Z-edge run of M_A. Define an edge word $\underline{\varpi}'$ for A by $\varpi'_k = \varpi_k$ if $\varpi_k \in \mathbb{R}_{\geq 0}$, and $\varpi'_k = e$ if ϖ_k is an edge of M_A derived from the edge e of A by an edge family $\Psi(e, \vec{\lambda}, \vec{\mu}, \vec{\nu}, tp)$. Then by Lemmas 3.16 and 3.19, for each k,

$$Post_A^{\varpi'_0\varpi'_1\cdots\varpi'_k}(Z) = \beta_A(Post_{M_A}^{\varpi_0\varpi_1\cdots\varpi_k}(lowhighZ)) \supset \beta_A(\{q'_{k+1}\}) \neq \emptyset.$$

Hence by Theorem 3.27, there is a state $q \in Z$ such that $\underline{\varpi}' \in ELang(q)$. ■

Corollary 3.29 *The ω-language emptiness problem for initialized rectangular automata with compact nondeterminism is PSPACE-complete.*

The case of bounded nondeterminism

Let A be a rectangular automaton. Recall that the ω-language of an automaton consists of all words accepted by divergent runs. Let $CLang(A)$ be the set of infinite timed words accepted runs of A that are not necessarily divergent. Note that in the case of compact nondeterminism, $CLang(A)$ is limit-closed. This is no longer the case for bounded nondeterminism. Consider for example, the timed automaton D in Figure 7. Every finite prefix of the infinite timed word $\underline{\pi} = \sigma' \frac{1}{2} \sigma \frac{1}{4} \sigma \frac{1}{8} \sigma \cdots$ is generated by a finite run of D, and yet $\underline{\pi} \notin Lang(A)$.

However, we show that $Lang(A)$ is still limit-closed for all rectangular automata with bounded nondeterminism. That is, whenever every finite prefix of a timed word $\underline{\pi}$ in which the time steps sum to infinity can be generated by a finite run of A, then the infinite sequence $\underline{\pi}$ is accepted by a run of A. Bounded regions have no analogue to Proposition 3.23, and this greatly complicates the proof of limit closure. Limit closure of $Lang(A)$ is now proven by a detailed case analysis of the activity function. The following technical lemma is used to establish the boundedness of all zones appearing in the proof of limit closure.

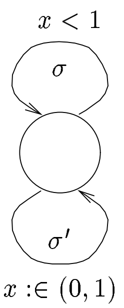

Figure 7: The need for time divergence

Lemma 3.30 *Let A be a rectangular automaton with bounded nondeterminism. For every bounded multirectangular zone $Z \subset Q_A$, and every $\varpi \in \mathbb{R}_{\geq 0} \cup E_A$, the zone $Post_A^{\varpi}(Z)$ of ϖ-successors of Z is bounded and multirectangular.*

The next lemma gives the heart of the proof of limit closure. It says that for 1-dimensional A with constant bounded activity, $Lang(A)$ is limit closed with respect to timed edge words without updates of the continuous state.

Lemma 3.31 *Let $\mathcal{I}$ be a finite set of nonempty intervals, and let R, act be bounded intervals. Let $(t_k)_{k\in\mathbb{N}}$ be a sequence of positive real numbers with $\sum_{k=0}^{\infty} t_k = \infty$, and let $(I_k)_{k\in\mathbb{N}}$ be a sequence of intervals such that $I_0 = R$, and for every $k \geq 1$, I_k is the intersection of one or more members of $\mathcal{I}$. Suppose for each $k \in \mathbb{N}$, there is a finite sequence $x_0, x_1, \ldots, x_k$ of real numbers such that for all $0 \leq j \leq k$, $x_j \in I_j$, and for each $0 \leq j < k$, $\frac{x_{j+1}-x_j}{t_j} \in act$. Then there is an infinite sequence $(x_k)_{k\in\mathbb{N}}$ such that for each $k \in \mathbb{N}$, $x_k \in I_k$ and $\frac{x_{k+1}-x_k}{t_k} \in act$.*

Proof. Think of a continuous variable x, initialized nondeterministically to some value in R, and with $\dot{x} \in act$. Call a finite sequence $x_0, x_1, \ldots, x_k$ *k-admissible* if for all $0 \leq j \leq k$, $x_j \in I_j$, and for each $0 \leq j < k$, $\frac{x_{j+1}-x_j}{t_j} \in act$. Call an infinite sequence $(x_k)_{k\in\mathbb{N}}$ *admissible* if for each $k \in \mathbb{N}$, $x_k \in I_k$ and $\frac{x_{k+1}-x_k}{t_k} \in act$.

Case 1: $0 \notin \overline{act}$. Suppose $act \subset (\epsilon, \infty)$, where $\epsilon > 0$. The case of $act \subset (-\infty, \epsilon)$ is handled symmetrically. Let h be larger than all of the finite endpoints of the intervals in $\mathcal{I}$. The point here is that the speed of x is bounded below by ϵ, and so that once $\frac{h-\inf R}{\epsilon}$ time has passed, no matter what the initial value of x, the value of x will be greater than all of the finite bounds defining intervals from $\mathcal{I}$. Let m be large enough so that $\sum_{k=0}^{m-1} t_i > \frac{h-\inf R}{\epsilon}$. We claim that for every $x_0 \in R$ for which there exists an m-admissible finite sequence $x_0, x_1, \ldots, x_m$, there is in fact an admissible infinite sequence $(x_k)_{k\in\mathbb{N}}$ extending $x_0, x_1, \ldots, x_m$. By assumption, for every $k \in \mathbb{N}$, a k-admissible sequence exists. For any such sequence $y_0, y_1, \ldots, y_k$, it must be that $y_j > h$ for each $j > m$. It follows that since $\mathcal{I}$ is finite, and every I_i is an intersection of elements of $\mathcal{I}$, that for every $k \geq m$, $I_k \supset (h, \infty)$. Since act contains some $\epsilon > 0$, any m-admissible finite sequence $x_0, x_1, \ldots, x_m$ can be extended to the admissible infinite sequence

$$x_0, x_1, \ldots, x_m, x_m + \epsilon t_m, x_m + \epsilon(t_m + t_{m+1}), \ldots$$

Case 2: $0 = \inf act$. The case of $0 = \sup act$ is handled symmetrically. Among the I_i are only finitely many distinct intervals, because there are only finitely many intersections of the finitely many elements of $\mathcal{I}$. Let $\mathcal{W}$ be the set $\{I \subset \mathbb{R} \mid I = I_i \text{ for infinitely many } i\}$. Then $\bigcap \mathcal{W} \neq \emptyset$, because $act \cap (-\infty, 0) = \emptyset$, so x can never descend from an I_i to an I_j all of whose elements are less than those of I_i. Let m_1 be large enough so that for every $k \in \mathbb{N}$, $I_{m_1+k} \in \mathcal{W}$, and moreover that for every $W \in \mathcal{W}$, there is a $k < m_1$ with $I_k = W$. I.e., m_1 is large enough so that all elements of $\mathcal{W}$ have been met in the past, and only elements of $\mathcal{W}$ will be met in the future. Let $m_2 > m_1$ be large enough so that all elements of $\mathcal{W}$ are represented among $I_{m_1+1}, \dots I_{m_2-1}$. Let $x_0, x_1, \dots, x_{m_2}$ be an m_2-admissible sequence. Then $x_{m_1} \in \bigcap \mathcal{W}$, because x cannot decrease, each element of $\mathcal{W}$ contains at least one of the x_i with $i < m_1$, and each element of $\mathcal{W}$ contains at least one of the x_i with $m_1 < i$. If $0 \in act$, then the infinite sequence $x_0, x_1, \dots, x_{m_1}^{\omega}$ is admissible. If $0 \notin act$, then $x_{m_1} < \sup \bigcap \mathcal{W}$. Let $\delta = (\sup \bigcap \mathcal{W}) - x_{m_1}$. For each $i > m_1$, let ϵ_i be so that $0 < \epsilon_i < \frac{\delta}{t_i 2^i}$. Then the infinite sequence

$$x_0, x_1, \dots, x_{m_1}, x_{m_1} + \epsilon_{m_1+1} t_{m_1+1}, x_{m_1} + \epsilon_{m_1+1} t_{m_1+1} + \epsilon_{m_1+2} t_{m_1+2}, \dots$$

is admissible.

Case 3: 0 is in the interior of act **and** $\bigcap \mathcal{W} \neq \emptyset$**.** Since 0 is in the interior of act, every trajectory can be slowed down to give another trajectory. Let m_1 be as in the previous paragraph. Since $0 \in act$, whenever an $(m_1 + \ell)$-admissible finite sequence terminates in $\bigcap \mathcal{W}$, it can be extended to an admissible infinite sequence by repeating the last state ad infinitum. Such an $(m_1+\ell)$-admissible sequence terminating in $\bigcap \mathcal{W}$ exists, because there exist $W_1, W_2 \in \mathcal{W}$ with $\inf W_1 = \inf \bigcap \mathcal{W}$ (with same strictness) and $\sup W_2 = \sup \bigcap \mathcal{W}$ (with same strictness). Any $(m + \ell)$-admissible finite sequence with ℓ large enough so that both W_1, W_2 each appear twice in $I_{m+1}, \dots, I_{m+\ell-1}$ must have $m < i < k$ with $x_i \in W_1$ and $x_k \in W_2$. By slowing down the trajectory, $\bigcap \mathcal{W}$ can be reached: if $x_i \geq \sup \bigcap \mathcal{W}$ and $x_k \leq \inf \bigcap \mathcal{W}$, then for some j with $i \leq j < k$, $x_j \geq \sup \bigcap \mathcal{W}$ and $x_{j+1} < \sup \bigcap \mathcal{W}$. By letting y be any number such that $x_{j+1} < y$ and $y \in \bigcap \mathcal{W}$, the infinite sequence

$$x_0, x_1, \dots, x_m, \dots, x_j, y^{\omega}$$

is admissible.

Case 4: 0 is in the interior of act **and** $\bigcap \mathcal{W} = \emptyset$**.** Let $W_1, W_2 \in \mathcal{W}$ be such that every element of W_1 is greater than every element of W_2. Let m_1 be as in the previous two paragraphs. Let $m_1 < p_1 < q_1 < p_2$ be so that $I_{p_1} = I_{p_2} = W_1$ and $I_{q_1} = W_2$. Let $x_0, x_1, \dots, x_{p_2}$ be p_2-admissible. We will first show that for every $k \in \mathbb{N}$, there is a k-admissible finite sequence starting from x_0. This is obvious for $k \leq p_2$, so suppose $k > p_2$. Let $y_0, y_1, \dots, y_k$ be k-admissible. We now have three cases.

Subcase 4a: $x_{p_1} = y_{p_1}$**.** In this case $x_0, x_1, \dots, x_{p_1}, y_{p_1+1}, y_{p_1+2}, \dots, y_k$ is a k-admissible sequence.

Subcase 4b: $x_{p_1} < y_{p_1}$**.** If $x_{q_1} < y_{q_1}$, then by slowing down, the x_i sequence can meet up with the y_i sequence somewhere along the descent from W_1 to W_2. If $x_{q_1} > y_{q_1}$, then for some $p_1 < j \leq q_1$, $x_j < y_j$ and $x_{j+1} \geq y_{j+1}$. Since $\frac{y_{j+1} - y_j}{t_j} \in act$ and $\frac{x_{j+1} - x_j}{t_j} \in act$, and

$$y_{j+1} - y_j < y_{j+1} - x_j < x_{j+1} - x_j,$$

it must be that $\frac{y_{j+1} - x_j}{t_j} \in act$. Hence the finite sequence

$$x_0, x_1, \dots, x_j, y_{j+1}, y_{j+2}, \dots, y_k$$

is k-admissible.

Subcase 4c: $x_{p_1} > y_{p_1}$. If $x_{q_1} < y_{q_1}$, then by slowing down, the x_i sequence can meet up with the y_i sequence somewhere along the descent from W_1 to W_2. So suppose $x_{q_1} > y_{q_1}$. Now if $x_{p_2} > y_{p_2}$, then by slowing down, the x_i sequence can meet up with the y_i sequence somewhere along the ascent from W_2 to W_1. If $x_{p_2} < y_{p_2}$, the the y_i sequence must cross the x_i sequence as above, and the same $x_0, x_1, \ldots, x_j, y_{j+1}, y_{j+2}, \ldots, y_k$ construction provides a k-admissible finite sequence beginning with x_0. The subcase of $x_{p_1} > y_{p_1}$ is complete.

It remains to construct an admissible infinite sequence. Let $x_0 \in R$ be such that for every $k \in \mathbb{N}$, there is a k-admissible finite sequence starting with x_0. Let R_1 be the set of t_0-successors of x_0, i.e., $R_1 = \{y \in \mathbb{R} \mid \frac{y-x_0}{t_0} \in act\}$. Then R_1 is bounded by Lemma 3.30. So applying what we have already proven to R_1, the time sequence $\lambda k.t_{k+1}$, and the interval sequence $\lambda_k.I_{k+1}$, there is an $x_1 \in R_1$ such that for every k, there is a k-admissible (with respect to $\lambda k.t_{k+1}$ and $\lambda_k.I_{k+1}$) sequence beginning from x_1. Continuing inductively, we form an admissible sequence beginning at x_0. ■

Now the proof of the main theorem consists of reducing to one dimension, eliminating updates, and applying Lemma 3.31.

Theorem 3.32 *Let A be an initialized rectangular hybrid automaton with bounded nondeterminism and let $Z \subset Q_A$ be a bounded rectangular zone. Suppose $\underline{\varpi} \in (\mathbb{R}_{\geq 0} \cup E_A)^\omega$ is a timed edge word such that for every $k \in \mathbb{N}$, $Post_A^{\varpi_0\varpi_1\cdots\varpi_k}(Z) \neq \emptyset$. Then there is a state $q \in Z$ such that $\underline{\varpi} \in ELang_A(\{q\})$.*

Proof. It suffices to prove the proposition for 1-dimensional A, for each component of a run of a multi-dimensional A is independent of the other components—that is, an n-dimensional automaton A has the edge run $\underline{\varpi}$ iff each of the n 1-dimensional automata defined by restricting A to one continuous components has the corresponding component sequence of $\underline{\varpi}$ as an edge run. So suppose A is 1-dimensional. If there exist infinitely many k with $1 \in upd(\varpi_k)$, then by stringing together the pieces in between the updates, $\underline{\varpi} \in ELang_A(\{q\})$ for every state $q \in Z$ such that $Post^{\varpi_0\varpi_1\cdots\varpi_k}(q) \neq \emptyset$ for some k with $1 \in upd(\varpi_k)$. So assume that $1 \in upd(\varpi_k)$ for only finitely many k. It now suffices to assume that $1 \in upd(\varpi_k)$ for no k. Because if $k_{\max} = \max\{k \in \mathbb{N} \mid 1 \in upd(\varpi_k)\}$, then by proving the theorem with $Post^{\varpi_0\varpi_1\cdots\varpi_{k_{\max}}}(Z)$ in place of Z, and $\lambda p.\varpi_{1+k_{\max}+p}$ in place of $\underline{\varpi}$, the result for Z and $\underline{\varpi}$ follows by picking any $q \in Z$ with $Post^{\varpi_0\varpi_1\cdots\varpi_{k_{\max}}}(\{q\}) \neq \emptyset$. The proposition now follows from an application of Lemma 3.31, where the set $\mathcal{I}$ of intervals is the set of all nonempty values of inv_A, pre_A, and $post_A$. ■

As Corollary 3.28 follows from Theorem 3.27, so does the next corollary follow from Theorem 3.32.

Corollary 3.33 *For every initialized rectangular hybrid automaton A with bounded nondeterminism, $Lang(A) = Lang(M_A)$.*

Corollary 3.34 *The ω-language emptiness problem for initialized rectangular automata with bounded nondeterminism is PSPACE-complete.*

3.4 Simulation Relations

In the above translations, we used several mappings between the state spaces of the original automaton and the transformed automaton. We were interested only that the translations preserved reachability and ω-languages. Here we study these mappings in greater detail and show that they are (bi)simulations on the underlying labeled transition systems. In particular, the map α_M from the initialized multirate automaton M to the initialized stopwatch automaton S_M (see Section 3.1) is a timed bisimulation, and the map β_A from the initialized rectangular automaton A to the initialized multirate automaton M_A (see Section 3.2) flattens out into a timed forward simulation in one direction, and a timed backward simulation in the other.

Let A and B be two rectangular automata with the same observation alphabet. A relation $\chi \subset Reach_A \times Reach_B$ is a *timed forward simulation of B by A* [LV92] if

1. For every state $r \in Reach_B$, there exists a state $q \in Reach_A$ with $(q, r) \in \chi$.
2. For every initial state $r \in Init_B$, there is an initial state $q \in Init_A$ such that $(q, r) \in \chi$.
3. For all states $r, r' \in Reach_B$, every state $q \in Q_A$ with $(q, r) \in \chi$, and every $\pi \in \mathbb{R}_{\geq 0} \cup \Sigma$, if $r \xrightarrow{\pi} r'$ in B, then there exists a state $q' \in Reach_A$ such that $(q', r') \in \chi$ and $q \xrightarrow{\pi} q'$ in A.

The relation χ is a *timed backward simulation of B by A* if

1. For every state $r \in Reach_B$, there exists a state $q \in Reach_A$ with $(q, r) \in \chi$.
2. for every initial state $r \in Init_B$, and every $q \in Reach_A$, if $(q, r) \in \chi$, then $q \in Init_A$.
3. For all states $r, r' \in Reach_B$, every state $q' \in Q_A$ with $(q', r') \in \chi$, and every $\pi \in \mathbb{R}_{\geq 0} \cup \Sigma$, if $r \xrightarrow{\pi} r'$ in B, then there exists a state $q \in Reach_A$ such that $(q, r) \in \chi$ and $q \xrightarrow{\pi} q'$ in A.

A relation χ such that both χ and χ^{-1} are timed forward simulations is a called a *timed bisimulation*. It follows immediately from Lemma 3.2 that α_M is a bisimulation between the initialized multirate automaton M and the initialized stopwatch automaton S_M. There is also a bisimulation γ_S between the initialized stopwatch automaton S and the timed automaton T_S from Section 3.1. It is defined as follows. Let $(v, \mathbf{x}) \in Q_S$ be a state of S. Then $((v, \mathbf{x}), (w, f, \mathbf{y})) \in \gamma_S$ iff for each i, (1) $w = v$, (2) if $acts_S(v)_i = \{1\}$ then $y_i = x_i$ and $f(i) = \perp$, and (3) if $acts_S(v)_i = \{0\}$ then $f(i) = x_i$.

Let A be an initialized rectangular automaton. Define the relation $\hat{\beta}_A \subset Q_{M_A} \times Q_A$ by $(q, q') \in \hat{\beta}_A$ iff $q' \in \beta_A(q)$. Then $\hat{\beta}_A$ is a forward simulation of A by M_A, and $\hat{\beta}_A^{-1}$ is a backward simulation of M_A by A, if we restrict attention to $U_{M_A} \cap Reach_{M_A}$, the reachable part of the upper half space. The proof is immediate from Lemmas 3.16 and 3.19.

Proposition 3.35 *Let A be an initialized rectangular automaton. The relation $\hat{\beta}_A$ is a timed forward simulation of A by M_A, and $\hat{\beta}_A^{-1}$ is a timed backward simulation of M_A, restricted to its upper half-space, by A.*

The complete chain of relationships between A, M_A, S_{M_A}, and $T_{S_{M_A}}$ is shown in Figure 8. It follows that A timed backward simulates $T_{S_{M_A}}$, and $T_{S_{M_A}}$ timed forward simulates A. The opposite statements are false. In fact, A does not even forward simulate M_A in a time-abstract way, nor does M_A backward simulate A in a time-abstract way. *Time-abstract* simulations are defined by treating all time steps equally [ACH94]. Define $\stackrel{\text{time}}{\to} = \bigcup_{t \in \mathbb{R}_{\geq 0}} \stackrel{t}{\to}$. By replacing the alphabet $\mathbb{R}_{\geq 0} \cup \Sigma$ with $\{\stackrel{\text{time}}{\to}\} \cup \Sigma$ in the above definitions of timed simulations, we arrive at their time-abstract counterparts. Clearly, every timed simulation is also a time-abstract simulation (but not vice versa).

Proposition 3.36 *There exists a 1-dimensional (compact) initialized rectangular automaton C such that there is no time-abstract forward simulation of M_C by C, and no time-abstract backward simulation of C by M_C.*

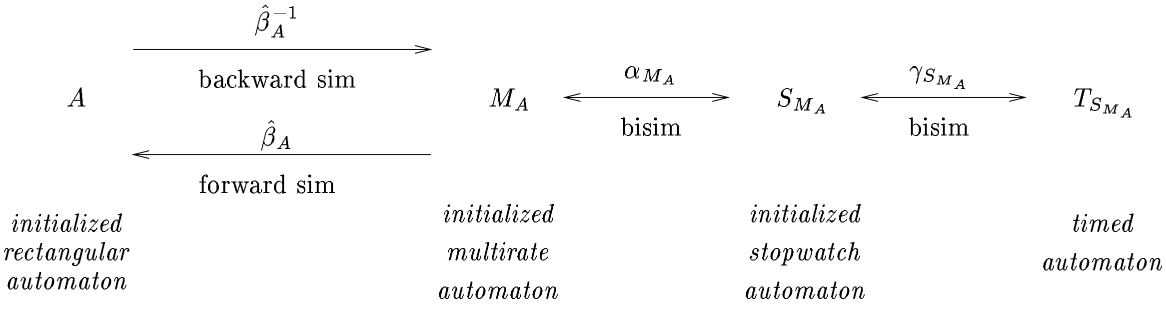

Figure 8: Chain of simulations from A to $T_{S_{M_A}}$

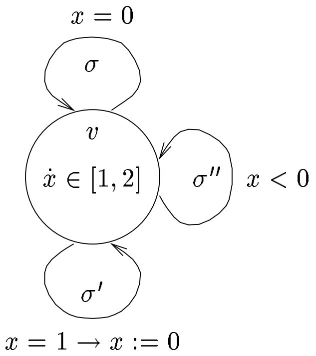

Figure 9: C does not forward simulate M_C, and M_C does not backward simulate C

Proof. Consider the automaton C in Figure 9. We suppress the $\vec{\lambda}$, $\vec{\nu}$, and *tp* components of M_C. If α is a forward simulation of M_C by C, then there must be some $(v, x) \in Q_C$ such that $((v, x), (v, (0, 1))) \in \alpha$. But this is impossible, for no such (v, x) can traverse both the σ and σ' edges. Suppose κ is a backward simulation κ of C by M_C. Then κ relates some state $(v, (\gamma, \delta))$ of M_C to the state $(v, 0)$. Since the edge labeled σ' assigns x to 0, and κ is a backward simulation, it follows that $\gamma = \delta = 0$. Now suppose $(v, x) \xrightarrow{t} (v, 0)$ in A, where $t > 0$. Since κ is a time-abstract backward simulation, there is a $t' \geq 0$ and a reachable state $(v, (-t', -2t')) \in Q_{M_C}$ that κ relates to (v, x). Since $-t' > -2t'$ for $t > 0$, and since $(v, (-t', -2t'))$ is reachable, it follows that $t' = 0$. So $((v, 0, 0), (v, x) \in \kappa$. This is impossible, because (v, x) is the target of a σ'' transition, whereas $(v, 0, 0)$ is not. ■

3.5 Automatic Verification

HyTech is an automatic analysis tool for hybrid systems [AHH93, HHWT95]. The core of HyTech is a semi-decision procedure that attacks the reachability problem for hybrid automata by iterating the *Post* operation on zones. That is, to check if a zone Z is reachable in a rectangular automaton A, HyTech computes the sequence $Init_A$, $Post_A(Init_A)$, $Post_A(Post_A(Init_A))$,..., until either Z is met or a fixpoint is reached within a finite number of iterations of $Post_A$. The HyTech procedure is known to terminate on every timed automaton with bounded invariants [HNSY94], where the rectangular automaton A has *bounded invariants* if for every $v \in V_A$, $inv(v)$ is bounded. Since $Post_A$ commutes with β_A, we obtain the following corollary, which asserts that the HyTech procedure

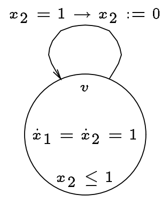

Figure 10: $Post^k(\{(v, 0, 0)\})$ does not converge

terminates on every initialized rectangular automaton with bounded invariants.

Corollary 3.37 *Let A be an initialized rectangular automaton with bounded invariants. For every rectangular zone $Z \subset Q_A$, there is a natural number $i \in \mathbb{N}$ such that $Pre^*_A(Z) = Pre^i_A(Z)$ and $Post^*_A(Z) = Post^i_A(Z)$.*

Proof. The corresponding statements are true for initialized multirate automata with bounded invariants [HNSY94]. Hence the result follows from Lemmas 3.16 and 3.19. ■

The HyTech procedure, however, does not terminate on all initialized rectangular automata. Consider Figure 10, in which both x_1 and x_2 are clocks. If Z is the singleton initial zone $\{(v, 0, 0)\}$, then for each $i \geq 1$, $Post^{2i-1}(Z) = \{(v, a, b) \mid 0 \leq a \leq i \text{ and } b = a - \lfloor a \rfloor\}$. So the computation does not reach a fixpoint within finitely many iterations of *Post*. In this section, we remedy this deficiency by preprocessing the input automaton. Suppose we wish to check if the zone Z is reachable in the initialized rectangular automaton A. We preprocess A, obtaining an initialized rectangular automaton A'', such that (1) Z is reachable in A iff Z is reachable in A'', and (2) the HyTech procedure terminates on A''. To facilitate the proof of (2), we first introduce another automaton A', which is exponentially larger than A.

An exponential preprocessing step

Let A be a n-dimensional rectangular automaton, and let $Z_1, Z_2 \subset Q_A$ be rectangular zones of A. We define an n-dimensional rectangular automaton A' with bounded invariant, and zones Z'_1, Z'_2 of A', such that $Z_2 \cap Post^*_A(Z_1) \neq \emptyset$ iff $Z'_2 \cap Post^*_{A'}(Z'_1) \neq \emptyset$. Let h be one more than the largest rational constant appearing in the definitions of A, Z_1, and Z_2. Let g be one less than the smallest such constant. The idea is to truncate all invariants, preguards, and postguards by intersection with $[g, h]^n$. When a variable reaches the upper or lower boundary, it stops moving. The automaton A' has vertex set $V_{A'} = V_A \times \{0, 1, 2\}^n$. Put $low = 0$, $ok = 1$, and $high = 2$, and let ok^n be the n-vector $(ok, ok, \ldots, ok)$. Vertices $(v, \vec{\kappa})$ with $\kappa_i = low$ represent states of A in which the ith continuous component x_i is no greater than g, vertices $(v, \vec{\kappa})$ with $\kappa_i = ok$ represent states of A in which $g \leq x_i \leq h$, and vertices $(v, \vec{\kappa})$ with $\kappa_i = high$ represent states of A in which $x_i \geq h$. The initial function of A' is defined by $init_{A'}(v) = init_A(v) \cap [g, h]^n$. The invariant function of A' is defined by

$$inv_{A'}(v, \vec{\kappa})_i = \begin{cases} \{h\}, & \text{if } \kappa_i = high, \\ inv_A(v)_i \cap [g, h], & \text{if } \kappa_i = ok, \\ \{g\}, & \text{if } \kappa_i = low. \end{cases}$$

The activity function of A' is defined by

$$act_{A'}(v, \vec{\kappa})_i = \begin{cases} act_A(v)_i, & \text{if } \kappa_i = ok, \\ \{0\}, & \text{if } \kappa_i \neq ok. \end{cases}$$

For each edge $e = (v, w)$ of A, the automaton A' has an edge $e' = ((v, ok^n), (w, ok^n))$ with $pre_{A'}(e') = pre_A(e) \cap [g, h]^n$, $upd_{A'}(e') = upd_A(e)$, and $post_{A'}(e') = post_A(e) \cap [g, h]^n$. Define $trunc : \mathbb{R}^n \to [g, h]^n$ by

$$trunc(\mathbf{x})_i = \begin{cases} g, & \text{if } x_i < g, \\ x_i, & \text{if } g \leq x_i \leq h, \\ h, & \text{if } x_i > h. \end{cases}$$

A rectangular region $R \subset \mathbb{R}^n$ is *gh-limited* if for each $1 \leq i \leq n$,

- either $\inf R_i = -\infty$ or $g + 1 \leq \inf R_i \leq h - 1$, and
- either $\sup R_i = \infty$ or $g + 1 \leq \sup R_i \leq h - 1$.

By definition of g and h, the following zones are gh-limited: Z_1, Z_2, and all values of inv_A, pre_A, and $post_A$. It follows that guards in A' have the same effect as guards in A.

Lemma 3.38 *Let $\mathbf{x} \in \mathbb{R}^n$, and let $R \subset \mathbb{R}^n$ be a gh-limited rectangular region. Then $\mathbf{x} \in R$ iff $trunc(\mathbf{x}) \in R \cap [g, h]^n$.*

The automaton A' has τ-edges to toggle the κ_i. For each vertex $v \in V_A$, each $i \in \{1, \ldots, n\}$, and each $\vec{\kappa}$ with $\kappa_i = ok$, there is an τ-edge $e_{\vec{\kappa}, low}$ from $(v, \vec{\kappa})$ to $(v, \vec{\kappa}[\kappa_i := low])$, and also an edge in the reverse direction from $(v, \vec{\kappa}[\kappa_i := low])$ to $(v, \vec{\kappa})$, each labeled with the guarded command $x_i = g \to x_i := g$. The trivial assignment keeps A' technically initialized. Similarly, there are τ-edges from $(v, \vec{\kappa})$ to $(v, \vec{\kappa}[\kappa_i := high])$ and from $(v, \vec{\kappa}[\kappa_i := high])$ to $(v, \vec{\kappa})$, each labeled with the guarded command $x_i = h \to x_i := h$. This completes the definition of the initialized rectangular automaton with bounded invariant A'.

Define $\zeta : Q_A \to Q_{A'}$ by $\zeta(v, \mathbf{x}) = ((v, ok^n), trunc(\mathbf{x}))$, and extend ζ to zones in the usual way. Define $Z'_j = \zeta(Z_j)$ for $j = 1, 2$.

Theorem 3.39 *Let A be an initialized rectangular automaton, and let $Z_1, Z_2 \subset Q_A$ be rectangular zones of A. Then $Z_2 \cap Post^*_A(Z_1) \neq \emptyset$ iff $Z'_2 \cap Post^*_{A'}(Z'_1) \neq \emptyset$.*

Proof. To see that if $Z_2 \cap Post^*_A(Z_1) \neq \emptyset$ then $Z'_2 \cap Post^*_{A'}(Z'_1) \neq \emptyset$, it suffices to note that ζ^{-1} is a timed forward simulation of A by A'. To simplify the notation, we prove this for 1-dimensional A. The extension to n dimensions is immediate. Suppose $\zeta(v^j, x^j) = (v^j, ok, y^j)$ for $j = 1, 2$.

First, suppose $(v, x^1) \xrightarrow{t} (v, x^2)$ where $t > 0$. Then $\frac{x^2 - x^1}{t} \in act_A(v)$. If $g \leq x^1, x^2 \leq h$, then $y^1 = x^1$ and $y^2 = x^2$, and

$$\frac{y^2 - y^1}{t} = \frac{x^2 - x^1}{t} \in act_A(v) = act_{A'}(v, ok).$$

Hence $\zeta(v, x^1) \xrightarrow{t} \zeta(v, x^2)$. Now suppose $x^1 \leq g \leq h \leq x^2$. Then there exist $t_1, t_2, t_3 \in \mathbb{R}_{\geq 0}$ such that $t = t_1 + t_2 + t_3$ and

$$(v, x^1) \xrightarrow{t_1} (v, g) \xrightarrow{t_2} (v, h) \xrightarrow{t_3} (v, x^2).$$

In A', $y^1 = g$ and $y^2 = h$, and since $act_{A'}(v, ok) = act_A(v)$, we have

$$(v, ok, y^1) \xrightarrow{\tau} (v, low, g) \xRightarrow{t_1} (v, low, g) \xrightarrow{\tau} (v, ok, g) \xRightarrow{t_2} (v, ok, h) \xrightarrow{\tau} (v, high, h) \xRightarrow{t_3} (v, high, h) \xrightarrow{\tau} (v, ok, y^2).$$

Again $\zeta(v, x^1) \xrightarrow{t} \zeta(v^2, x^2)$. Other positions of x^1 and x^2 relative to g and h are handled similarly.

For edge transitions, the key fact is that all preguards and postguards are gh-limited. Suppose $(v^1, x^1) \xrightarrow{e} (v^2, x^2)$ where $e \in E_A$. Then $x^1 \in pre_A(e)$, and $x^2 \in post_A(e)$, and so by Lemma 3.38,

$y^1 = trunc(x^1) \in pre_{A'}(e')$ and $y^2 = trunc(x^2) \in post_{A'}(e')$. In addition, $upd_A(e) = upd_{A'}(e)$, and so $\zeta(v^1, x^1) \xrightarrow{e'} \zeta(v^2, x^2)$.

We now show that if $Z'_2 \cap Post^*_{A'}(Z'_1) \neq \emptyset$ then $Z_2 \cap Post^*_A(Z_1) \neq \emptyset$. Again, we prove the result for 1-dimensional A—the generalization to n dimensions being immediate. Suppose

$$(v^0, ok, y^0) \xrightarrow{\pi_1} (v^1, ok, y^1) \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_m} (v^m, ok, y^m)$$

in A', where $(v^0, y^0) \in Z'_1$ and $(v^m, y^m) \in Z'_2$. We will find $x^0, \ldots, x^m$ such that

$$(v^0, x^0) \xrightarrow{\pi_1} (v^1, x^1) \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_m} (v^m, x^m)$$

in A, and for each $0 \leq j \leq m$, $trunc(x^j) = y^j$. Since all postguards are rectangular, it suffices to assume that each $\pi_j \in E_{A'}$ has $upd(E_{A'}) = \emptyset$, otherwise we string together solutions obtained in between variable assignments. Consequently, $act_A(v^j) = act_A(v^k)$ for each $0 \leq j, k \leq m$. Now by Lemma 3.38, it suffices to assume that $\pi_j \in \mathbb{R}_{\geq 0}$ for each j, that is, each π_j is a time step. Let act be the common value of the $act_A(v^j)$. If $0 \in act$, then

$$(v^0, y^0) \xrightarrow{\pi_1} (v^1, y^1) \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_m} (v^m, y^m)$$

already in A, so putting $x^j = y^j$ for each j, we are finished, because $y^0 \in Z_1$ and $y^1 \in Z_2$ by Lemma 3.38. Now suppose $act \subset (0, \infty)$. Here the most interesting case is given by $y^0 = g$ and $y^m = h$. In this case, there exist $0 \leq k \leq k' < m$ such that $y^j = g$ for $j < k$, $g < y^j < h$ for $k \leq j \leq k'$, and $y^j = h$ for $j > k'$. We put $x^j = y^j$ for $k \leq j \leq k'$. To set the x^j for $j > k'$, we need only determine a suitable slope. Let p be such that for some $h' \geq h$, $p = \frac{h' - y^{k'}}{\pi_{k'+1}} \in act$. Such a p exists because $(v^{k'}, ok, y^{k'}) \xrightarrow{\pi_{k'+1}} (v^{k'+1}, ok, h)$ in A'. Put $x^j = y^{k'} + p(j - k')$ for each $j \geq k'$. Then $(v^j, x^j) \xrightarrow{\pi_{j+1}} (v^{j+1}, x^{j+1})$ for $j = k', k'+1, \ldots, m-1$. It remains to set x^j for $j \leq k$, which is done in the same way. Since $(v^k, ok, g) \xrightarrow{\pi_{k+1}} (v^{k+1}, ok, y^{k+1})$, there exists a $p \in act$ such that for some $g' \leq g$, $p = \frac{y^{k+1} - g'}{\pi_{k+1}} \in act$. Then for each $0 \leq j \leq k$, define $x^j = y^{k+1} - p(k + 1 - j)$. Then $(v^j, x^j) \xrightarrow{\pi_{j+1}} (v^{j+1}, x^{j+1})$ for $j = 0, 1, \ldots, k$, and we are finished. Other cases are handled in a similar fashion. ■

A linear preprocessing step

Whereas the automaton A' uses the discrete part of the state space to store information about variables that get too large, the automaton A'' uses the continuous part. Instead of stopping a variable when it reaches h, the automaton A'' supplies a nondeterministic jump to any value above h. Formally, the automaton A'' is identical to A, except for some additional edges. For each vertex $v \in V_A$, and each $1 \leq i \leq n$, the automaton A'' has two τ-edges labeled respectively with the guarded commands

$$x_i \leq g \rightarrow x_i :\in (-\infty, g) \text{ and } x_i \geq h \rightarrow x_i :\in (h, \infty).$$

Here $x_i :\in I$ is a nondeterministic assignment into the interval I. Our first theorem shows that reachability in A is equivalent to reachability in A''. Since A'' is simply A with some extra edges, the "only if" portion of the proof is immediate. The "if" is extremely similar to the second part of the proof of Theorem 3.39 above.

Theorem 3.40 *Let A be an initialized rectangular automaton, and let $Z_1, Z_2 \subset Q_A$ be rectangular zones of A. Then $Z_2 \cap Post^*_A(Z_1) \neq \emptyset$ iff $Z_2 \cap Post^*_{A''}(Z_1) \neq \emptyset$.*

Next we show that HyTech terminates on A''.

Theorem 3.41 *Let A be an initialized rectangular automaton. For every $\varpi \in (\mathbb{R}_{\geq 0} \cup E_A)$, and every gh-limited rectangular zone Z,*

$$Post^{\varpi}_{A'}(trunc(Z)) = trunc(Post^{\varpi}_{A''}(Z)).$$

Proof. The statement for $\varpi \in E_A$ is proven by Lemma 3.38. For $\varpi \in \mathbb{R}_{\geq 0}$, it suffices to prove the result for one-dimensional A. Suppose $(v, z) \in Z$, and $(v, z^1) \xrightarrow{\varpi} (v, z^2)$ in A''. If $g < z^1, z^2 < h$, then immediately $(v, z^1) \xrightarrow{\varpi} (v, z^2)$ in A'. The most interesting case is $z^1 < g < h < z^2$. In this case there exists a $t \leq \varpi$ such that $\frac{h-g}{t} \in act(v)$. Hence

$$trunc(v, z^1) = (v, ok, g) \xrightarrow{\tau} (v, low, g) \overset{\pi-t}{\Rightarrow} (v, low, g) \xrightarrow{\tau} (v, ok, g) \overset{t}{\Rightarrow} (v, ok, h) = trunc(v, z^2)$$

in A'. Similar arguments apply to different relative positions of z^1, z^2, g, and h. Therefore $Post^{\varpi}_{A'}(trunc(Z)) \supset trunc(Post^{\varpi}_{A''}(Z))$. The reverse inclusion is easy. ■

Now from Theorem 3.41, Corollary 3.37, and Proposition 2.2 follows this corollary.

Corollary 3.42 *Let A be an initialized rectangular automaton, and let $Z_1, Z_2 \subset Q_A$ be rectangular zones of A. Then there there is a natural number $i \in \mathbb{N}$ such that $Pre^*_{A''}(Z_1) = Pre^i_{A''}(Z_1)$ and $Post^*_{A''}(Z_1) = Post^i_{A''}(Z_1)$.*

We conclude that with the addition of a preprocessing step (creating A'' from A by adding $2nV$ edges), HyTech may be used to solve the reachability problem for initialized rectangular automata.

4 Undecidability

In the previous section, we showed that initialized rectangular automata form a decidable class of hybrid automata. In this section, we show that they form a maximal such class. We proceed in two steps. First, we show that without initialization, even a single two-slope variable leads to an undecidable reachability problem. Second, we show that the rectangularity of the model must remain inviolate. Any coupling between coordinates, such as comparisons between variables, already brings undecidability with a single non-clock variable. (Timed automata, which have only clock variables, remain decidable in the presence of variable comparisons [AD94].) A main consequence is the undecidability of compact automata with clocks and one stopwatch. These automata are important for the verification of duration properties.

The rectangular automaton A is *simple* if it meets the following restrictions:

1. Exactly one variable of A is *not* a clock.
2. For every vertex v, $inv(v)$ and $act(v)$ are compact.
3. For every edge $e \in E$, and for all $1 \leq i \leq n$, if $i \in upd(e)$ then $post(e)_i = [0, 0]$, and if $i \notin upd(e)$ then $post(e)_i = pre(e)_i$.
4. For every edge $e \in E$, $pre(e)$ is compact (and hence $post(e)$ is compact by 3).

The automaton A is *m-simple* if it meets restrictions 2–4, and exactly m variables of A are not clocks. We use simple automata for our undecidability results. Restriction 3 allows only deterministic variable updates, and so the nondeterminism of jumps in the continuous state, allowed in our model of rectangular automata, does not contribute to our undecidability results. Many limited decidability results are based on the digitization of real-numbered delays [HMP92, BES93, BER94, PV94]. Since the digitization technique requires closed guards and invariants, restrictions 2, 3, and 4 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, BR95, Hen95]. Restriction 1 implies that these results do not generalize either. We might also replace condition 2 with the trivial invariant $\lambda v \in V. \mathbb{R}^n$, when our proofs would require only minor modifications.

All of our undecidability proofs are reductions from the halting problem for two-counter machines to the reachability problem for simple rectangular automata. A two-counter machine consists of a finite control and two unbounded counters. Three types of instructions are used: branching based upon whether a specific counter has value 0, incrementing a counter, and decrementing a counter (which leaves unchanged a counter value of 0). In our reductions, each counter is modeled by a clock. Counter value r (usually) corresponds to clock value $k_1(\frac{k_2}{k_1})^r$, where k_1 and k_2 are the slopes of a two-slope variable in a simple automaton, k_1 being the larger. When $\frac{k_1}{k_2} = 2$, decrementing (resp. incrementing) a counter corresponds to doubling (resp. halving) the value of the corresponding clock. Notice that since $k_1 > k_2$, it is the density of the continuous domain, rather than its infinite extent, that is used to code the potentially unbounded counter values.

4.1 Uninitialized Automata

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

Theorem 4.1 *For every two slopes $k_1, k_2 \in \mathbb{Q}$ with $k_1 \neq k_2$, the reachability problem is undecidable for simple rectangular automata with one two-slope variable of slopes k_1 and k_2.*

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. Let W be a positive rational number. A simple rectangular automaton A is *W-wrapping* if

- for every variable a of A that is a clock, and for every vertex v, $inv(v)(a) = [0, W]$, and
- if z is the non-clock variable of A, and z takes only nonnegative slopes (i.e., $act(v)(z) \subset [0, \infty)$ for each vertex v), then for each vertex v, $inv(v)(z) = [0, W \cdot \max_{w \in V} \max act(w)(z)]$, and
- if z is the non-clock variable of A, and z takes only nonpositive slopes, then for each vertex v, $inv(v)(z) = [W \cdot \min_{w \in V} \min act(w)(z), 0]$, and
- if z is the non-clock variable of A, and z takes both positive and negative slopes, then for each vertex v, $inv(v)(z) = [W \cdot \min_{w \in V} \min act(w)(z), W \cdot \max_{w \in V} \max act(w)(z)]$.

A *W-wrapping edge* for a clock a is an edge $e = (v, v)$ from vertex v to itself such that $pre(e)(a) = [W, W]$, $upd(e) = \{a\}$, and $post(e)(a) = [0, 0]$. That is, a wrapping edge for a is labeled with the guarded command $a = W \to a := 0$. A *W-wrapping edge* for a non-clock variable z and a vertex v with $act(v) = [k, k]$ is an edge from v to itself labeled with the guarded command $z = kW \to z := 0$. The invariant of a wrapping automaton forces wrapping edges to be taken when

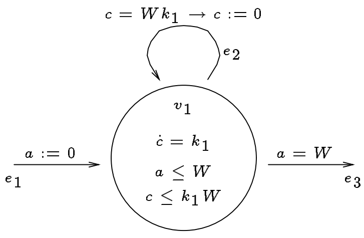

Figure 11: Wrapping lemma: the skewed clock c retains its entry value upon exit

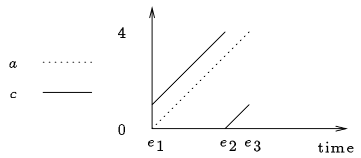

Figure 12: Proof of the Wrapping Lemma for slope 1

they are enabled. We use wrapping to simulate discrete events by continuous rounds taking W (or some multiple thereof) units of time. The wrapping edges ensure that variables take the same values at the beginning and end of a round, unless they are explicitly reassigned by a non-wrapping edge. This is the content of our first lemma. *We stress that in figures, we leave these wrapping conditions implicit*, in particular, we omit invariants from every figure after those regarding the basic lemmas, and we omit wrapping edges beginning with Figure 17. The wrapping technique originated in [Cer92].

Wrapping lemma. *Let W be a positive rational number. Let $k_1 \in \mathbb{Q}$, and consider the simple W-wrapping automaton fragment of Figure 11, Suppose that $c = \gamma$ when edge e_1 is traversed into v_1, where $0 < \gamma < k_1 W$ if $k_1 > 0$, and $k_1 W < \gamma < 0$ if $k_1 < 0$. Then the next time e_3 is traversed out of v_1, again $c = \gamma$.*

Proof. Figure 12 contains a time portrait illustrating the proof for $W = 4$ and $k_1 = 1$. The markings e_1, e_2, and e_3 along the time axis show at which points these edges are traversed. We give the proof for $k_1 > 0$. In order for e_3 to be taken in the future, the following series of steps must occur: 1) e_1 is traversed; 2) exactly $\frac{1}{k_1}(Wk_1 - \gamma)$ units of time elapse, after which c has value Wk_1, and a has value $\frac{1}{k_1}(Wk_1 - \gamma)$; 3) the wrapping edge e_2 is traversed, after which c has value 0, and a has value $\frac{1}{k_1}(Wk_1 - \gamma)$; 4) exactly $W - \frac{1}{k_1}(Wk_1 - \gamma) = \frac{\gamma}{k_1}$ units of time elapse, after which a has value W and c has value γ. ■

By only allowing clocks c and d to wrap simultaneously, we can check if the two have the same value.

Equality lemma. *Let W be a positive rational number. Consider the simple W-wrapping automaton fragment of Figure 13, in which all variables are clocks. Suppose that $c = \gamma$ and $d = \delta$ when edge e_1 is traversed into v_1, where $0 < \gamma, \delta < W$. 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, by assigning skewed clock d to 0 at the same time as wrapping skewed clock c to 0, we perform the assignment $d := \frac{k_2}{k_1} c$, where $\dot{c} = k_1$ and $\dot{d} = k_2$.

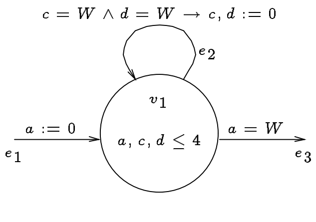

Figure 13: Equality lemma: testing $c = d$

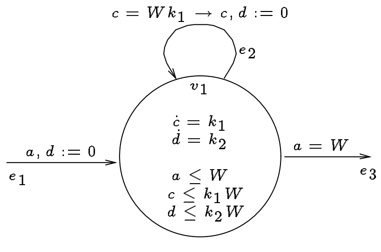

Figure 14: Assignment lemma: performing the assignment $d := \frac{k_2}{k_1} c$

Assignment lemma. *Let W be a positive rational number. Let $k_1, k_2 \in \mathbb{Q}$, $k_1 \neq 0$, and consider the simple W-wrapping automaton fragment of Figure 14. Suppose that $c = \gamma$ when edge e_1 is traversed into v_1, where $0 < \gamma < k_1 W$ if $k_1 > 0$, and $k_1 W < \gamma < 0$ if $k_1 < 0$. Then the next time e_3 is traversed, $c = \gamma$ and $d = \frac{k_2}{k_1}\gamma$.*

Proof of Theorem 4.1. We reduce the halting problem for two-counter machines to the reachability problem for simple wrapping rectangular automata with a two-slope variable taking slopes k_1 and k_2. Let M be a two-counter machine with counters C and D. Let a, b, c, and d be clocks, and let z be a two-slope variable of slopes k_1 and k_2.

Case 1: $k_1 > k_2 > 0$ or $k_1 < k_2 < 0$. Our automaton is W-wrapping, where W may be chosen to be any number larger than k_1. We encode the values of the counters C and D in the values of the clocks c and d, respectively. We encode counter value r by clock value $|k_1|(\frac{k_2}{k_1})^r$. The relationships $c = |k_1|(\frac{k_2}{k_1})^C$ and $d = |k_1|(\frac{k_2}{k_1})^D$ hold when $a = 0$ or $a = W$ (except when more than one time interval of duration W is needed to simulate one computation step). The test $C = 0$ is implemented by two edges e_1 and e_2, where $pre(e_1)(c) = [k_1, k_1]$ (corresponding to $C = 0$) and $pre(e_2)(c) = [0, k_2]$ (corresponding to $C \neq 0$). Decrementing a counter corresponds to first checking if it is zero as above, and if not, then multiplying the corresponding clock value by $\frac{k_1}{k_2}$. This is implemented by two assignment lemma constructions in series as in Figure 15. In the first, $\dot{z} = k_1$; it performs $z := k_1 c$. In the second, $\dot{z} = k_2$; it performs $c := \frac{1}{k_2} z$. The bottom portion of Figure 15 contains a time portrait showing the operation of the decrementation fragment with $W = 4$, $k_1 = 2$, and $k_2 = 1$. Incrementing a counter corresponds to multiplying the corresponding clock value by $\frac{k_2}{k_1}$. It is done by reversing these assignments, as in Figure 16. First $z := k_2 c$ is performed, and then $c := \frac{1}{k_1} z$. The bottom portion of Figure 16 contains a time portrait showing the operation of the incrementation fragment with $W = 4$, $k_1 = 2$, and $k_2 = 1$.

Case 2: $k_2 = 0$. In the remaining figures, we omit the wrapping edges required for the clock d. The construction is insensitive to the sign of k_1. The encoding of the two-counter machine is given by counter value r corresponding to clock value 2^{1-r}. We use wrapping constant 4. Decrementing

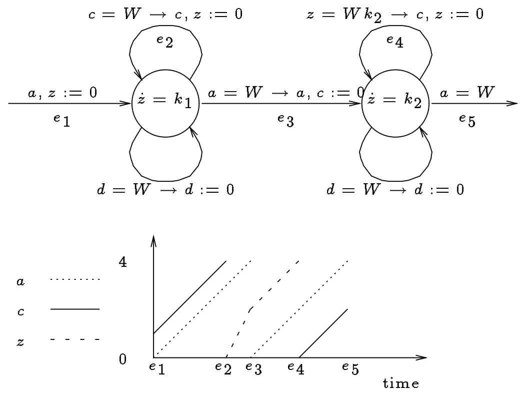

Figure 15: Counter decrement: multiplying c by $\frac{k_1}{k_2}$ using the two-slope variable z

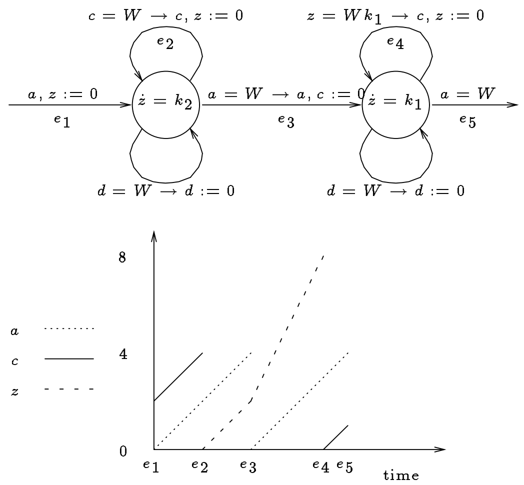

Figure 16: Counter increment: multiplying c by $\frac{k_2}{k_1}$ using the two-slope variable z

a counter corresponds to doubling the corresponding clock. The doubling procedure is given in Figure 17. The idea is to perform $z := k_1 c$ using the assignment lemma, then to put $\dot{z} = 0$ until c reaches W again, and then to put $\dot{z} = k_1$ so that when a reaches W, $z = 2k_1\gamma$, where γ is the original value of c. Then perform $c := \frac{1}{k_1}z$ with the assignment lemma. The lower portion of the figure gives a time portrait illustrating the operation of the fragment for $k_1 = 2$. Halving c requires two auxiliary clocks x and y. First, a value is guessed in x. Then $y := 2x$ is performed using the above doubling procedure. Then $c = y$ is checked by the equality lemma, and if this succeeds, then $c := x$ is performed using the assignment lemma.

Case 3: $k_2 < 0 < k_1$**.** First suppose $|k_2| < |k_1|$. We use clock value $k_1(\frac{|k_2|}{k_1})^r$ to encode counter value r. The wrapping constant W can be any number larger than k_1. But now we use two synchronization clocks a and b. Clock c is synchronized with a, and clock d is synchronized with b. The relationship $c = k_1(\frac{|k_2|}{k_1})^C$ holds when $a = 0$ or $a = W$, and the relationship $d = k_1(\frac{|k_2|}{k_1})^D$ holds when $b = 0$ or $b = W$. To multiply c by $\frac{k_1}{|k_2|}$, we first perform $z := k_1 c$ and reset c to 0. Then we put $\dot{z} = k_2$, and when z reaches 0, we reset a to 0. At this point $c = \frac{k_1}{|k_2|}\gamma$, where γ is the original value of c. See Figure 18. The bottom portion of the figure contains a time portrait for $W = 4$,

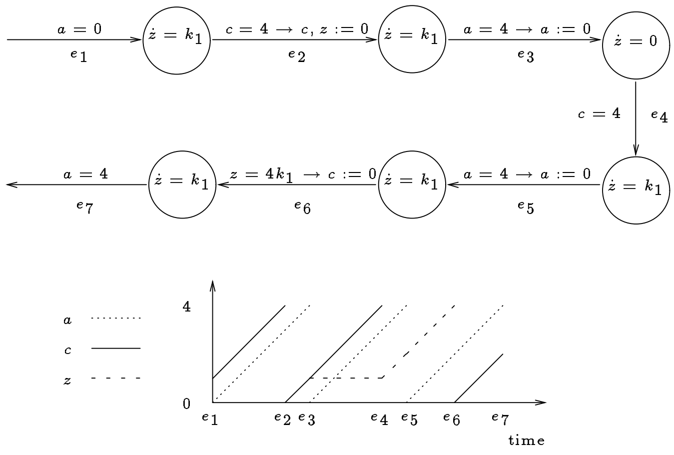

Figure 17: Doubling c using variable z taking slopes $0, k_1$

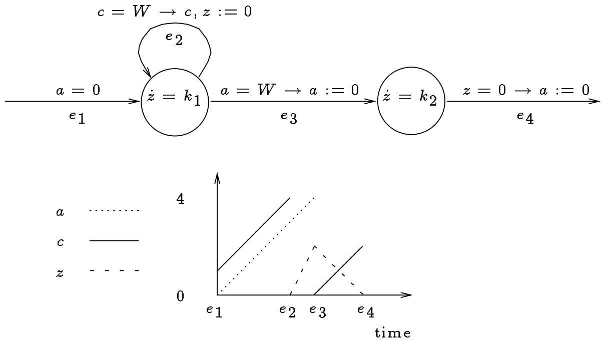

Figure 18: Multiplying c by $\frac{k_1}{k_2}$ when $k_2 < 0 < k_1$

$k_1 = 2$, and $k_2 = -1$. To multiply c by $\frac{k_2}{k_1}$, simply reverse the slopes of z. I.e., perform $z := k_2 c$, reset c to 0, then put $\dot{z} = k_1$ and when z reaches 0, reset a to 0. See Figure 19. The bottom portion of the figure contains a time portrait for $W = 4$, $k_1 = 2$, and $k_2 = -1$.

If $|k_2| > |k_1|$, we use clock value $|k_2|(\frac{k_1}{|k_2|})^r$ for counter value r, which simply switches the roles of multiplying by $\frac{k_1}{|k_2|}$ and multiplying by $\frac{|k_2|}{k_1}$. Finally, suppose $k_2 = -k_1$. In this case we use clock value 2^{1-r} for counter value r, and the wrapping constant is 4. Again we use separate synchronization clocks for c and d. To double c, perform $z := k_1 c$, and then put $\dot{z} = -k_1$, resetting a when z reaches 0. See Figure 20, which gives the construction, and also a time portrait for $k_1 = 3$. Halving c is done by nondeterministically guessing the midpoint of the interval of time between $c = 4$ and $a = 4$. The guess is checked by starting z at value 0, giving z at slope k_1 for the first half, and slope $-k_1$ for the second half. If z returns to 0 at the same instant that a reaches 4, the guess was correct. See Figure 21, which gives the construction and a time portrait for $k_1 = 5$, $W = 4$. ■

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.

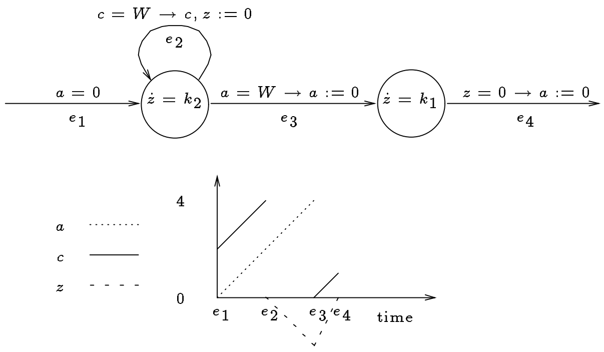

Figure 19: Multiplying c by $\frac{k_2}{k_1}$ when $k_2 < 0 < k_1$

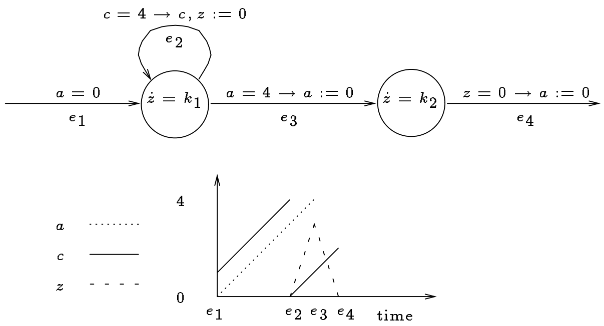

Figure 20: Doubling c when $k_2 = -k_1$

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 $assign(e)\colon \{1, \ldots, n\} \to \{1, \ldots, n\}$. The transition-step relation $\xrightarrow{\sigma}$ is then redefined as follows: $(v, \mathbf{x}) \xrightarrow{\sigma} (w, \mathbf{y})$ 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.*

Proof. First assume $k > 0$. With assignment updates, it is simple to multiply the value of the clock c by k when a skewed clock z of slope k is available. Simply use the assignment lemma to perform $z := kc$, and then use an assignment update to perform $c := z$. To divide c by k, do the reverse: use an assignment update to perform $z := c$, and then use the assignment lemma to perform $c := \frac{1}{k}z$. We give the construction in Figure 22, along with a time portrait for $k = 3$, $W = 4$.

Now assume $k < 0$ and $k \neq -1$. We use one synchronization clock a for clock c, and another synchronization clock b for clock d, as in the proof of Theorem 4.1 for $k < 0$. To multiply c by $|k|$, perform $z := kc$ by the assignment lemma, and then perform $a := z; c := 0$ with an assignment update. If γ was the original value of c, then after this sequence $a = k\gamma$ and $c = 0$. After $k\gamma$ time

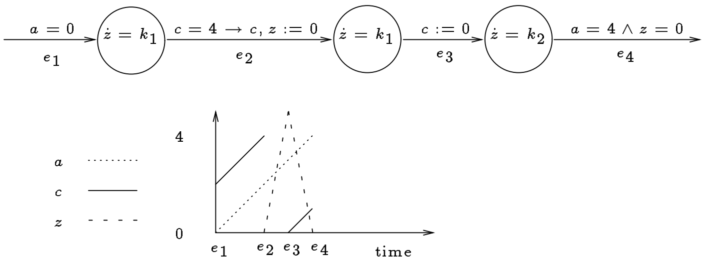

Figure 21: Halving c when $k_2 = -k_1$

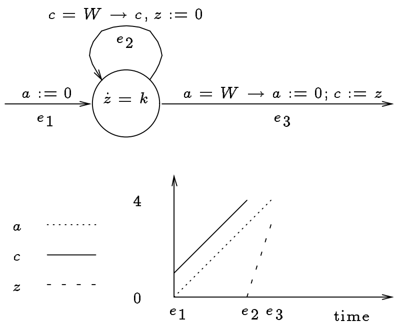

Figure 22: Multiplying by $k > 0$ with assignment updates and a skewed clock of slope k

units pass, $a = 0$ and $c = k\gamma$. See Figure 23, which includes the construction and a time portrait for $k = -2$, $W = 4$. To divide by $|k|$, perform $z := c; c := 0$ with an assignment update, and then $\frac{\gamma}{|k|}$ time units later when z reaches 0 (and c reaches $\frac{\gamma}{|k|}$), perform $a := 0$. The constructions for $k = -1$ are similar.

When $k = 0$, we have a memory cell, which we refer to as m. We use clock value 2^{1-r} for counter value r. The doubling procedure is given in Figure 24. Simply assign $m := c$ when $a = 0$, then wait for c to reach 4 and then assign $c := m$. When a reaches 4, c has twice its original value. Halving is done by guessing and checking, as in Case 2 of Theorem 4.1. ■

Triangular preguards, postguards, and 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 $\xrightarrow{\sigma}$ is then redefined as follows: $(v, \mathbf{x}) \xrightarrow{\sigma} (w, \mathbf{y})$ 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 states of A 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).*

Proof. Triangular preguards, postguards, or invariants allow comparisons between the variables of the form $x = y$. This allows an assignment update $y := x$ to be simulated by the unguarded reset

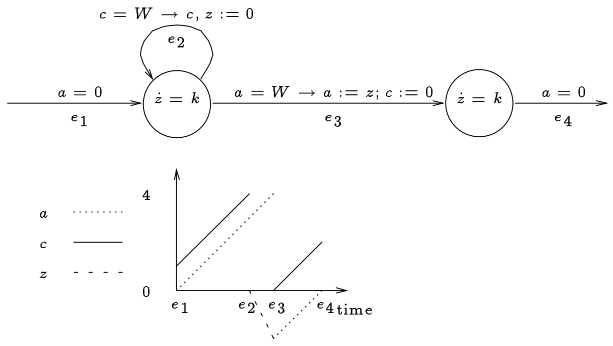

Figure 23: Multiplying by $|k|$ with assignment updates and a skewed clock of slope $k < 0$

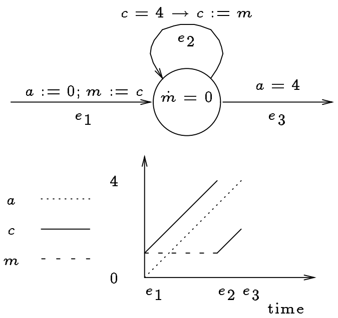

Figure 24: Doubling with assignment updates and a memory cell

$y := 0$ followed later in time by the test $y = x$. It follows that the constructions of Corollary 4.2 can be implemented with triangular preguards, postguards, or invariants replacing assignment updates. We give an example multiplication construction (performing $c := kc$) for triangular invariants and $k = 2$ in Figure 25. The "$c = z$" inside of the rightmost vertex indicates the triangular invariant. ■

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.*

Proof. For this proof we use three variables x, y, z with global triangular activity $1 \leq \dot{x} \leq \dot{y} \leq \dot{z} \leq 2$. The doubling construction is given in Figure 26. The variable y will actually take only slopes 1 and 2; the former is accomplished by resetting z to 0 when a wraps to 0, and then later testing $a = 4 \wedge z = 4$; similarly the latter is accomplished by resetting x to 0 when a wraps to 0, and then later testing $a = 4 \wedge x = 8$. In this way, the two-slope variable constructions of Theorem 4.1 can be duplicated. ■

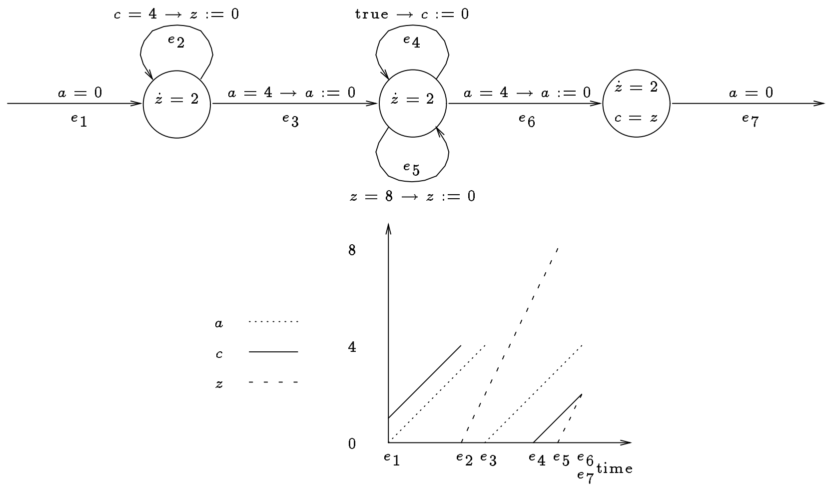

Figure 25: Doubling with triangular invariants and a skewed clock

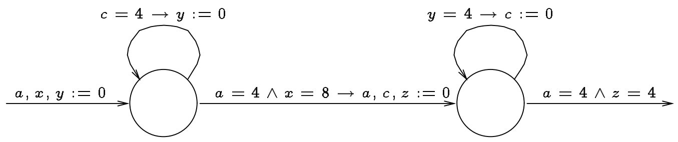

Figure 26: Doubling with the triangular activity $1 \leq \dot{x} \leq \dot{y} \leq \dot{z} \leq 2$

A decidable class of triangular activities

Last, we show that symmetric triangular activities are harmless if all variables that are unrelated by the activity are completely asynchronous. 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 $\leq$ is thus an equivalence relation, and consequently induces a partition of $\{1, \ldots, n\}$. This 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.

Let A be a rectangular automaton with a clock-partition activity. The automaton A cannot have bounded nondeterminism, because of its unbounded activity function. We say that A has *bounded assignments* if $Init_A$ is bounded, and for every $e \in E_A$ and every $i \in upd(e)$, $post(e)_i$ is bounded.

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

Proof. We show that if the clock-partition activity of an automaton A is the trivial equivalence relating each pair of indices, then A has an effective finite timed bisimulation. It follows that any finite product of such systems again has a finite timed bisimulation, the product of the component bisimulations [Hen95].

Note that a rectangular automaton A with the trivial clock partition activity is essentially a timed automaton in which the time scale varies, for each variable moves at the same rate as all of the others. It follows that the time-abstract bisimulation of timed automata [AD94] (called *region* *equivalence*) is a finite *timed* bisimulation for A. The reachability and ω-language emptiness problems can be solved in space $\log B$, where B is the number of bisimulation equivalence classes. This gives the desired PSPACE inclusions. PSPACE-hardness follows from the PSPACE-hardness of timed automata. ■

5 Conclusion

There are three uniform extensions of finite-state machines with real-valued variables. *Timed automata* [AD94] equip finite-state machines with perfect clocks, and the reachability problem for timed automata is decidable. *Linear hybrid automata* [ACHH93] equip finite-state machines with continuous variables whose behavior satisfies linear constraints, and the reachability problem for linear hybrid automata is undecidable. Yet because the *Pre* and *Post* operations of linear hybrid automata maintain the linearity of zones, the reachability problem is semidecidable, and thus the verification of many linear hybrid systems is possible. This observation has been exploited in the model checker HyTech [AHH93, HHWT95]. *Initialized rectangular automata* equip finite-state machines with drifting clocks, that is, continuous variables whose behavior satisfies rectangular constraints. Initialized rectangular automata lie strictly between timed automata and linear hybrid automata, at the boundary of decidability. One one hand, initialized rectangular automata generalize timed automata without incurring a complexity penalty. Their reachability problem is PSPACE-complete, and under the natural restriction of bounded nondeterminism, so is their ω-language emptiness problem. (We do not know the complexity of the ω-language emptiness problem without the restriction of bounded nondeterminism.) On the other hand, initialized rectangular automata form a maximal decidable class of hybrid systems, because even the simplest uninitialized or non-rectangular systems have undecidable reachability problems.

In summary, there are two factors for decidability: (1) rectangularity, that is, the behavior of all variables is decoupled; (2) initialization, i.e., a variable is reinitialized whenever its activity changes.

Initialized rectangular automata are also interesting from a practical perspective. First, the model checker HyTech terminates on every initialized rectangular automaton with bounded invariants, and on every initialized rectangular automaton after a linear preprocessing step. Second, many distributed communication protocols assume that local clocks have bounded drift. Such protocols are naturally modeled as initialized rectangular hybrid automata. HyTech has recently been applied successfully to verify one such protocol used in Philips audio components [HW95]. Third, initialized rectangular automata can be used to conservatively approximate hybrid systems with general dynamical laws [OSY94, PV95, HH95a].

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

References

[ACH93] R. Alur, C. Courcoubetis, and T.A. Henzinger. Computing accumulated delays in real-time systems. In C. Courcoubetis, editor, *CAV 93: Computer-aided Verification*, Lecture Notes in Computer Science 697, pages 181–193. Springer-Verlag, 1993.

[ACH94] R. Alur, C. Courcoubetis, and T.A. Henzinger. The observational power of clocks. In B. Jonsson and J. Parrow, editors, *CONCUR 94: Concurrency Theory*, Lecture Notes in Computer Science 836, pages 162–177. Springer-Verlag, 1994.

[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*, 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.

[AHH93] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. In *Proceedings of the 14th Annual Real-time Systems Symposium*, pages 2–11. IEEE Computer Society Press, 1993.

[AHV93] R. Alur, T.A. Henzinger, and M.Y. Vardi. Parametric real-time reasoning. In *Proceedings of the 25th Annual Symposium on Theory of Computing*, pages 592–601. ACM Press, 1993.

[Alu91] R. Alur. *Techniques for Automatic Verification of Real-time Systems.* PhD thesis, Stanford University, 1991.

[BER94] A. Bouajjani, R. Echahed, and R. Robbana. Verifying invariance properties of timed systems with duration variables. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *FTRTFT 94: Formal Techniques in Real-time and Fault-tolerant Systems*, Lecture Notes in Computer Science 863, pages 193–210. Springer-Verlag, 1994.

[BES93] A. Bouajjani, R. Echahed, and J. Sifakis. On model checking for real-time properties with durations. In *Proceedings of the Eighth Annual Symposium on Logic in Computer Science*, pages 147–159. IEEE Computer Society Press, 1993.

[BR95] A. Bouajjani and R. Robbana. Verifying ω-regular properties for subclasses of linear hybrid systems. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 437–450. Springer-Verlag, 1995.

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

[DOY94] C. Daws, A. Olivero, and S. Yovine. Verifying ET-LOTOS programs with KRONOS. In *Proceedings of Seventh International Conference on Formal Description Techniques*, 1994.

[Hen95] T.A. Henzinger. Hybrid automata with finite bisimulations. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata, Languages, and Programming*, Lecture Notes in Computer Science 944, pages 324–335. Springer-Verlag, 1995.

[HH95a] T.A. Henzinger and P.-H. Ho. Algorithmic analysis of nonlinear hybrid systems. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 225–238. Springer-Verlag, 1995.

[HH95b] T.A. Henzinger and P.-H. Ho. HyTech: The Cornell Hybrid Technology Tool. To appear in the Proceedings of the 1994 Workshop on Hybrid Systems and Autonomous Control, 1995.

[HHWT95] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: The next generation. To appear at RTSS, 1995.

[HKWT95] T.A. Henzinger, P.W. Kopke, and H. Wong-Toi. The expressive power of clocks. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata, Languages, and Programming*, Lecture Notes in Computer Science 944, pages 417–428. Springer-Verlag, 1995.

[HMP92] T.A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In W. Kuich, editor, *ICALP 92: Automata, Languages, and Programming*, Lecture Notes in Computer Science 623, pages 545–558. Springer-Verlag, 1992.

[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.

[HRP94] N. Halbwachs, P. Raymond, and Y.-E. Proy. Verification of linear hybrid systems by means of convex approximation. In B. LeCharlier, editor, *SAS 94: Static Analysis Symposium*, Lecture Notes in Computer Science 864, pages 223–237. Springer-Verlag, 1994.

[HW95] P.-H. Ho and H. Wong-Toi. Automated analysis of an audio control protocol. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 381–394. Springer-Verlag, 1995.

[KPSY93] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, Lecture Notes in Computer Science 736, pages 179–208. Springer-Verlag, 1993.

[LV92] N.A. Lynch and F. Vaandrager. Forward and backward simulations for timing-based systems. In J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, Lecture Notes in Computer Science 600, pages 397–446. Springer-Verlag, 1992.

[MP93] O. Maler and A. Pnueli. Reachability analysis of planar multi-linear systems. In C. Courcoubetis, editor, *CAV 93: Computer-aided Verification*, Lecture Notes in Computer Science 697, pages 194–209. Springer-Verlag, 1993.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In E.W. Mayr and C. Puech, editors, *STACS 95*, Lecture Notes in Computer Science 900, pages 229–242. Springer-Verlag, 1995.

[MV94] J. McManis and P. Varaiya. Suspension automata: a decidable class of hybrid automata. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 105–117. Springer-Verlag, 1994.

[NOSY93] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, Lecture Notes in Computer Science 736, pages 149–178. Springer-Verlag, 1993.

[OSY94] A. Olivero, J. Sifakis, and S. Yovine. Using abstractions for the verification of linear hybrid systems. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 81–94. Springer-Verlag, 1994.

[PV94] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 95–104. Springer-Verlag, 1994.

[PV95] A. Puri and P. Varaiya. Verification of hybrid systems using abstractions. To appear in the Proceedings of the 1994 Workshop on Hybrid Systems and Autonomous Control, 1995.

[VW86] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the First Annual Symposium on Logic in Computer Science*, pages 322–331. IEEE Computer Society Press, 1986.