# What's Decidable about Hybrid Automata ${ }^{1}$ 

Thomas A. Henzinger*, ${ }^{2}$, Peter W. Kopke ${ }^{\dagger}$, Anuj Puri*, and Pravin Varaiya*<br>* Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720;<br>${ }^{\dagger}$ Computer Science Department, Cornell University, Ithaca, New York 14853

Received January 1998


#### 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 a boundary between decidability and undecidability for the reachability problem of 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 independent 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 $\omega$-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. © 1998 Academic Press


## 1. INTRODUCTION

A hybrid automaton [ACHH93, NOSY93] combines the discrete dynamics of a finite automaton with the continuous dynamics of a dynamical system. Hybrid automata thus provide a mathematical model for digital computer systems that interact with an analog environment in real time. Case studies indicate that the model of hybrid automata is useful for the analysis of embedded software and hardware, including distributed processes with drifting clocks, real-time schedulers, and protocols for the control of manufacturing plants, vehicles,

[^0]and robots (see, for example, [HRP94, ACH ${ }^{+} 95$, HHWT95, HW95, NS95, AHH96, Cor96, HWT96, SMF97]). Two problems that are central to the analysis of hybrid automata are the reachability problem and the more general $\omega$-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 of the $\omega$-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 attempts a systematic characterization 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 $4 \mathrm{~cm} \mathrm{~s}^{-1}$, or rises at any rate between 1 and $3 \mathrm{~cm} \mathrm{~s}^{-1}$. We model these two situations by the dynamical laws $\dot{x} \in[-4,-2]$ and $\dot{x} \in[1,3]$-so-called rectangular flow constraints [PV94]-which enforce piecewise-linear envelopes on the water-level trajectories. Rectangular-flow 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, PBV96, HHWT98], 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 of rectan-gular-flow automata with two restrictions: (1) the values of two variables with different flow constraints are never
compared; (2) whenever the flow constraint of a variable changes, the value of the variable is reinitialized. Second, under the additional assumption of bounded nondeterminism (which requires that the successor of a bounded region be bounded), we obtain a PSPACE algorithm for checking $\omega$-language emptiness of rectangular-flow 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 flow constraints are admitted.

The first two results are proven by translating rectangularflow automata of dimension $n$ into timed automata of dimension $2 n+1$, where the dimension is the number of real-valued variables. The translation preserves finitary languages, and in the case of bounded nondeterminism, also $\omega$-languages. In addition, the translation implies that, when applied to rectangular-flow automata that meet restrictions (1) and (2), existing semidecision procedures for the reachability problem of hybrid automata terminate. Such procedures have been implemented in the НуТесн verification tool [AHH96, HHWT97].

The third result is proven by reduction from the halting problem for two-counter machines. In an attempt to characterize the undecidability frontier, we sharpen the reduction as much as possible. First, we prove that any relaxation of restriction (1) leads to the undecidability of the reachability problem for timed automata augmented with a single constant-slope variable whose slope is different from 1. Second, we prove that any relaxation of restriction (2) leads to the undecidability of the reachability problem for timed automata augmented with a single uninitialized two-slope variable, such as a stopwatch, which is a variable whose slope is always either 0 or 1 .

Previous work. Over the past few years, there have been many decidability and undecidability results about hybrid systems; we list only those that led to the present work. The first decidability result for hybrid automata was obtained for timed automata, whose reachability and $\omega$-language emptiness problems are PSPACE-complete [AD94]. Under restrictions (1) and (2), that result was later generalized to automata with variables that run at any constant positive slopes $\left[\mathrm{ACH}^{+} 95\right]$, and to the reachability problem for automata with nonstrict rectangular flow constraints [PV94]. In [BES93, KPSY93, BER94, MV94, BR95, ACH97], it was shown that, under various strong side conditions, reachability is decidable for timed automata with one stopwatch, but the general problem of one-stopwatch automata was left open.

As far as undecidability results are concerned, in [Cer92] it was shown that reachability is undecidable for timed automata with three stopwatches, as well as for timed automata with one memory cell (a variable of constant slope 0 ) and assignments between variables. It was also known that reachability is undecidable for timed automata
with six memory cells and no assignments [AHV93], for timed automata with two three-slope variables and restriction (1) [KPSY93], for timed automata with two nonclock con-stant-slope variables $\left[\mathrm{ACH}^{+} 95\right]$, and for timed automata with additive clock constraints [AD94].

## 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 $\mathbb{R}^{n}\left[\mathrm{ACH}^{+} 95\right]$. A run of a hybrid automaton is a sequence of edge steps and time steps. During an edge step (also called jump) the discrete and continuous states are updated according to a guarded command. During a time step (also called flow) 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 $\mathbb{R}_{\geqslant 0}$ to denote the set $\{x \in \mathbb{R} \mid x \geqslant 0\}$ of the 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 $n>0$, a subset of $\mathbb{R}^{n}$ is called a region. A closed and bounded region is compact. A region $R \subset \mathbb{R}^{n}$ is rectangular if it is a cartesian product of (possibly unbounded) intervals, all of whose finite endpoints are rational. We write $R_{i}$ for the projection of $R$ on the $i$ th coordinate, so that $R=\prod_{i=1}^{n} R_{i}$. The set of all rectangular regions in $\mathbb{R}^{n}$ is denoted $\mathscr{R}^{n}$.

## Definition of Rectangular Automata

An $n$-dimensional rectangular automaton $A$ consists of a finite directed multigraph $(V, E)$, a finite observation alphabet $\Sigma$, three vertex labeling functions init: $V \rightarrow \mathscr{R}^{n}$, inv: $V \rightarrow \mathscr{R}^{n}$, and flow: $V \rightarrow \mathscr{R}^{n}$, and four edge labeling functions pre: $E \rightarrow \mathscr{R}^{n}$, post: $E \rightarrow \mathscr{R}^{n}$, jump: $E \rightarrow 2^{\{1, \ldots, n\}}$, and obs: $E \rightarrow \Sigma$. An $n$-dimensional rectangular automaton with $\varepsilon$ moves differs in that the function obs maps $E$ into $\Sigma^{\varepsilon}$, where $\Sigma^{\varepsilon}=\Sigma \cup\{\varepsilon\}$ augments the observation alphabet with the null observation $\varepsilon \notin \Sigma$. When we discuss more than one automaton, we use the subscript $A$ to identify the components of $A$. For example, the vertex set of $A$ may be denoted $V_{A}$.

The initial function init specifies a set of initial automaton states. When the discrete state begins at vertex $v$, the continuous state must begin in the initial region $\operatorname{init}(v)$. The preguard function pre, the postguard function post, and the jump function jump 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 vertex $v$ and the continuous state lies in the preguard region pre( $e$ ). For each $i$ in the jump set $\operatorname{jump}(e)$, the $i$ th coordinate of the continuous state is nondeterministically assigned a new value in the postguard interval post $(e)_{i}$. For each $i \notin j u m p(e)$, the $i$ th coordinate of the continuous state is not changed and must lie in $\operatorname{post}(e)_{i}$. The observation function obs identifies every edge traversal with an observation from $\Sigma$ or $\Sigma^{\varepsilon}$. The invariant function inv and the flow function flow 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 $\left(C^{\infty}\right)$ trajectory within the invariant region $\operatorname{inv}(v)$, whose first time derivative remains within the flow region flow $(v)$. A rectangular automaton with $\varepsilon$ moves may traverse $\varepsilon$ edges during time steps.

Note that if we replace rectangular regions with arbitrary linear regions in the definition of rectangular automata, we obtain the linear hybrid automata of [AHH96]. 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 $e=(v, w)$ of $A$, and every coordinate $i \in\{1, \ldots, n\}$ with $\operatorname{flow}(v)_{i} \neq \operatorname{flow}(w)_{i}$, we have $i \in \operatorname{jump}(e)$. It follows that whenever the $i$ th continuous coordinate of an initialized automaton changes its dynamics, as given by the flow function, then its value is nondeterministically reinitialized according to the postguard function.

The rectangular automaton $A$ has bounded nondeterminism if (1) for every vertex $v \in V$, the regions $\operatorname{init}(v)$ and $\operatorname{flow}(v)$ are bounded, and (2) for every edge $e \in E$ and every coordinate $i \in\{1, \ldots, n\}$ with $i \in j u m p(e)$, the interval post $(e)_{i}$ is bounded. Note that bounded nondeterminism does not imply finite branching. It ensures that the edge and time successors of a bounded region are bounded.

## The Labeled Transition System of a Rectangular Automaton

The rectangular automaton $A$, possibly with $\varepsilon$ moves, defines a labeled transition system with an infinite state space $Q$, the infinite set $\Sigma \cup \mathbb{R}_{\geqslant 0}$ of labels, and the binary transition relations $\xrightarrow{\tau}$ on $Q$, one for each label $\tau \in \Sigma \cup \mathbb{R} \geqslant 0$. Each transition with label $\sigma \in \Sigma$ corresponds to an edge step whose observation is $\sigma$. Each transition with label $t \in \mathbb{R}_{\geqslant 0}$ corresponds to a time step of duration $t$. The states and the transitions of $A$ are defined formally as follows.

States. 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 $\mathbf{x} \in \operatorname{inv}(v)$. The state space $Q \subset V \times \mathbb{R}^{n}$ of $A$ is the set of all states of $A$. A subset of $Q$ is called a zone of $A$. Each zone $Z \subset Q$ can be uniquely decomposed into a collection $\bigcup_{v \in V}\{v\} \times[Z]^{v}$ of regions $[Z]^{v} \subset \mathbb{R}^{n}$, one for each vertex $v \in V$. The zone $Z$ is rectangular (resp. bounded; compact) if each region $[Z]^{v}$ is rectangular
(resp. bounded; compact). The state ( $v, \mathbf{x}$ ) is an initial state of $A$ if $\mathbf{x} \in \operatorname{init}(v)$. The initial zone of $A$, denoted Init, is the set of all initial states of $A$. Notice that both the state space $Q$ and the initial zone Init are rectangular.

Jump transitions. For each edge $e=(v, w)$ of $A$, we define the binary relation $\xrightarrow{e} \subset Q^{2}$ by $(v, \mathbf{x}) \xrightarrow{e}(w, \mathbf{y})$ iff $\mathbf{x} \in \operatorname{pre}(e)$, and $\mathbf{y} \in \operatorname{post}(e)$, and for every coordinate $i \in\{1, \ldots, n\}$ with $i \notin \operatorname{jump}(e)$, we have $x_{i}=y_{i}$. Hence $\mathbf{x}$ and $\mathbf{y}$ differ only at coordinates in the jump set jump $(e)$. For each observation $\sigma \in \Sigma^{\varepsilon}$, we define the edge-step relation $\xrightarrow{\sigma} \subset Q^{2}$ by $q \xrightarrow{\sigma} r$ iff $q \xrightarrow{e} r$ for some edge $e \in E$ with $o b s(e)=\sigma$.

Flow transitions. For each nonnegative real $t \in \mathbb{R}_{\geqslant 0}$, we define the binary relation $\xrightarrow{[t]} \subset Q^{2}$ by $(v, \mathbf{x}) \xrightarrow{[t]}(w, \mathbf{y})$ iff (1) $v=w$ and (2) either $t=0$ and $\mathbf{x}=\mathbf{y}$, or $t>0$ and $(\mathbf{y}-\mathbf{x}) / t \in \operatorname{flow}(v)$. Notice that due to the convexity of rectangular regions, $(v, \mathbf{x}) \xrightarrow{[t]}(v, \mathbf{y})$ iff there is a smooth function $f:[0, t] \rightarrow \operatorname{inv}(v)$, with first derivative $f^{\prime}$, such that $f(0)=\mathbf{x}$, and $f(t)=\mathbf{y}$, and for all reals $s \in(0, t)$, we have $f^{\prime}(s) \in f l o w(v)$. Hence the continuous state may evolve from $\mathbf{x}$ to $\mathbf{y}$ via any smooth trajectory satisfying the constraints imposed by $\operatorname{inv}(v)$ and flow $(v)$. If $A$ does not have $\varepsilon$ moves, then we define the time-step relation $\xrightarrow{t}$ to be $\xrightarrow{[t]}$. If $A$ has $\varepsilon$ moves, then the time-step relation $\xrightarrow{t} \subset Q^{2}$ is defined by $q \xrightarrow{t} r$ iff there exists an integer $m \geqslant 1$, nonnegative reals $t_{1}, \ldots, t_{m}$, and states $q_{1}, \ldots, q_{2 m-2}$ such that $q \xrightarrow{\left[t_{1}\right]} q_{1} \xrightarrow{\varepsilon} q_{2}$ $\xrightarrow{\left[t_{2}\right]} q_{3} \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q_{2 m-2} \xrightarrow{\left[t_{m}\right]} r$ and $t=\sum_{i=1}^{m} t_{i}$.

We write $\Pi=\Sigma^{\varepsilon} \cup \mathbb{R}_{\geqslant 0} \cup E \cup\left[\mathbb{R}_{\geqslant 0}\right]$ for the set of labels that arise in connection with the automaton $A$, where $\left[\mathbb{R}_{\geqslant 0}\right]=\left\{[t] \mid t \in \mathbb{R}_{\geqslant 0}\right\}$. Let $Z$ be a zone of $A$, and let $\pi$ be a label from $\Pi$. We define $\operatorname{Post}^{\pi}(Z)=\{q \in Q \mid \exists r \in Z . r \xrightarrow{\pi} q\}$ to be the zone of states that are reachable in one $\pi$ step from $Z$, and we define $\operatorname{Post}(Z)=\bigcup_{\pi \in \Sigma \cup \mathbb{R}_{\geqslant 0}} \operatorname{Post}^{\pi}(Z)$ to be the zone of states that are reachable in one edge or time step from $Z$. Similarly, we define $\operatorname{Pr}^{\pi}(Z)=\{q \in Q \mid \exists r \in Z . q \xrightarrow{\pi} r\}$ to be the zone of states from which $Z$ is reachable in one $\pi$ step, and we define $\operatorname{Pre}(Z)=\bigcup_{\pi \in \Sigma \cup \mathbb{R}_{\geq 0}} \operatorname{Pr} e^{\pi}(Z)$ to be the zone of states from which $Z$ is reachable in one edge or time step. Notice that $\operatorname{Post}(Z) \supset Z$ and $\operatorname{Pre}(Z) \supset Z$ because of time steps of the form $\xrightarrow{0}$.

## The Reverse Automaton

For an $n$-dimensional rectangular automaton $A$, the reverse automaton $-A$ is an $n$-dimensional rectangular automaton that defines the same state space as $A$, but with the transition relations reversed. The vertex set, observation alphabet, initial and invariant functions of $-A$ are the same as for $A$. For each vertex $v$, the flow region of $-A$ is defined by flow $_{-A}(v)=$ $\left\{\mathbf{x} \in \mathbb{R}^{n} \mid-\mathbf{x} \in\right.$ flow $\left._{A}(v)\right\}$. For each edge $e=(v, w)$ of $A$, the reverse automaton $-A$ has the edge $-e=(w, v)$ with $\operatorname{pre}_{-A}(-e)=\operatorname{post}_{A}(e), \quad \operatorname{jump}_{-A}(-e)=j u m p_{A}(e), \quad$ and
$\operatorname{post}_{-A}(-e)=$ pre $_{A}(e)$. From these definitions, Proposition 2.1 follows immediately.

Proposition 2.1. For all states $q$ and $r$ of a rectangular automaton $A$, and every label $\pi \in \Pi$, we have $q \xrightarrow{\pi}{ }_{A} r$ iff $r \xrightarrow{\pi}{ }_{-A} q$.

It follows that for every zone $Z$ of $A$, and every label $\pi \in \Pi, \operatorname{Post}_{A}^{\pi}(Z)=\operatorname{Pre}_{-A}^{\pi}(Z)$ and $\operatorname{Pre}_{A}^{\pi}(Z)=\operatorname{Post}_{-A}^{\pi}(Z)$.

## Multirectangular Zones

A zone $Z$ is multirectangular if $Z$ is a finite union of rectangular zones. Multirectangularity is preserved by edge and time steps.

Proposition 2.2. For every multirectangular zone $Z$ of a rectangular automaton $A$, and every label $\pi \in \Sigma^{\varepsilon} \cup\left[\mathbb{R}_{\geqslant 0}\right]$, the zones $\operatorname{Post}^{\pi}(Z)$ and $\operatorname{Pre}^{\pi}(Z)$ are multirectangular.

Proof. We give the proof for Post; the result for Pre then follows from Proposition 2.1. Since each relation $\xrightarrow{\sigma}$ with $\sigma \in \Sigma^{\varepsilon}$ is a finite union of relations $\xrightarrow{e}$ with $e \in E$, it suffices to prove the proposition for $\pi \in E \cup\left[\mathbb{R}_{\geqslant 0}\right]$. 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 every elementary zone $Z=\{v\} \times R$, the successor zone $\operatorname{Post}^{\pi}(Z)$ is elementary. If $\pi=(v, w)$ is an edge of $A$, then $\operatorname{Post}^{\pi}(Z)=$ $\{w\} \times S$, where

$$
S_{i}=\left\{\begin{array}{l}
R_{i} \cap \operatorname{pre}(\pi)_{i} \cap \operatorname{post}(\pi)_{i} \cap \operatorname{inv}(w)_{i}, \\
\quad \text { if } \quad i \notin \operatorname{jump}(\pi)_{i}, \\
\operatorname{post}(\pi)_{i} \cap \operatorname{inv}(w)_{i}, \\
\text { if } \quad i \in \operatorname{jump}(\pi)_{i} \quad \text { and } \quad R_{i} \cap \operatorname{pre}(\pi)_{i} \neq \varnothing, \\
\varnothing, \quad \text { if } R_{i} \cap \operatorname{pre}(\pi)_{i}=\varnothing
\end{array}\right.
$$

If $\pi=[0]$, then $\operatorname{Post}^{\pi}(Z)=Z$. If $\pi \in\left[\mathbb{R}_{\geqslant 0}\right]$ and $\pi>0$, then $\operatorname{Post}^{\pi}(Z)=\{v\} \times S$, where

$$
\begin{aligned}
S_{i}= & \operatorname{inv}(v)_{i} \cap\left\{{ }^{1} \inf \left(R_{i}\right)+\pi \cdot \inf \left(\text { flow }(v)_{i}\right), \infty\right) \\
& \cap\left(-\infty, \sup \left(R_{i}\right)+\pi \cdot \sup \left(\text { flow }(v)_{i}\right)\right\}^{2}
\end{aligned}
$$

Here, $\left\{{ }^{1}\right.$ stands for [ if $R_{i}$ and $\operatorname{flow}(v)_{i}$ are left-closed, $\left\{{ }^{1} \text { stands for (if } R_{i} \text { or flow }(v)_{i} \text { are left-open, }\right\}^{2}$ stands for ] if $R_{i}$ and flow $(v)_{i}$ are right-closed, and $\}^{2}$ stands for ) if $R_{i}$ or flow $(v)_{i}$ are right-open.

Given a zone $Z$ of the rectangular automaton $A$ and a finite sequence $\pi_{0} \pi_{1} \cdots \pi_{m} \in \Pi^{*}$ of labels, we define $\operatorname{Post}^{\pi_{0} \pi_{1}}(Z)$ $=\operatorname{Post}^{\pi_{1}}\left(\operatorname{Post}^{\pi_{0}}(Z)\right)$, and we define Post $^{\pi_{0} \pi_{1} \cdots \pi_{m}}(Z)$ inductively in the usual way. Also, we define $\operatorname{Post}^{*}(Z)=$ $\bigcup_{i \in \mathbb{N}} \operatorname{Post}^{i}(Z)$ to be the zone of states that are reachable from $Z$ in a finite number of edge and time steps (here Post ${ }^{i}$ denotes $i$ applications of the function Post). Analogous
definitions are made for Pre. A state $q$ is a reachable state of $A$ if $q \in$ Post $^{*}($ Init $)$. The reachable zone of $A$, denoted $\operatorname{Reach}(A)$, is the set Post*(Init) of all reachable states of $A$. The reachable zone $\operatorname{Reach}(A)$ is an infinite union of rectangular zones and may not be multirectangular.

## The $\omega$-Language of a Rectangular Automaton

Let $A$ be a rectangular automaton, possibly with $\varepsilon$ moves. A timed word for $A$ is a finite or infinite sequence $\bar{\tau}=$ $\tau_{0} \tau_{1} \tau_{2} \cdots$ of letters from $\Sigma \cup \mathbb{R}_{\geqslant 0}$; that is, each $\tau_{i}$ is either an observation of $A$, or a nonnegative real that denotes a duration of time between observations. The timed word $\bar{\tau}$ is divergent if $\bar{\tau}$ is infinite and $\sum\left\{\tau_{i} \mid i \in \mathbb{N}\right.$ and $\left.\tau_{i} \in \mathbb{R} \geqslant 0\right\}=\sigma_{\tau_{0}}$. A run $\rho$ of $A$ is a finite or infinite sequence of the form $q_{0} \xrightarrow{\tau_{0}}$ $q_{1} \xrightarrow{\tau_{1}} q_{2} \xrightarrow{\tau_{2}} \cdots$, where $q_{0} \in$ Init, and for all $i \geqslant 0$, we have $q_{i} \in Q$ and $\tau_{i} \in \Sigma \cup \mathbb{R}_{\geqslant 0}$. The run $\rho$ accepts the timed word $\bar{\tau}=\tau_{0} \tau_{1} \tau_{2} \cdots$, and $\rho$ is called divergent if $\bar{\tau}$ is divergent. The $\omega$-language of $A$, denoted $\operatorname{Lang}(A)$, is the set of all divergent timed words that are accepted by runs of $A$.

## Example

It is often convenient to refer to each coordinate of the continuous state as a variable. We use letters from the beginning of the alphabet, such as $a, b, c, d$, for variables. If the variable $a$ corresponds to the $i$ th coordinate of the continuous state, we write $\operatorname{flow}(v)(a)$ for $f l o w(v)_{i}$, etc. In pictorial descriptions of rectangular automata, we annotate each vertex with its flow region, and sometimes with its invariant region. For example, if flow $(v)(a)=[3,5]$, $\operatorname{flow}(v)(b)=[4,4], \operatorname{inv}(v)(a)=(1,7]$, and $\operatorname{inv}(v)(b)=$ $(-\infty, 0]$, we write " $\dot{a} \in[3,5]$," " $\dot{b}=4$," " $1<a \leqslant 7$," and " $b \leqslant 0$ " inside vertex $v$. Often, however, we give the invariant function in the text and omit it from the figure. Edges are annotated with observations and guarded commands. A guarded command $\phi$ defines two regions pre $(\phi)$ and $\operatorname{post}(\phi)$, and a jump set $\operatorname{jump}(\phi)$, in a natural manner. For example, if $\phi$ is the (nondeterministic) guarded command

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

then $\operatorname{pre}(\phi)(a)=(-\infty, 5], \operatorname{pre}(\phi)(b)=[4,4], \operatorname{pre}(\phi)(c)=$ $(-\infty, \infty), j u m p(\phi)=\{b, c\}, \operatorname{post}(\phi)(a)=(-\infty, \infty), \operatorname{post}(\phi)(b)$ $=[7,7]$, and $\operatorname{post}(\phi)(c)=[0,5]$. As usual, when writing guarded commands, the guard true is omitted, and so is the empty list of assignments.
Consider, for instance, the 2 D rectangular automaton $\hat{A}$ of Fig. 1. The observation alphabet of $\hat{A}$ is $\left\{\sigma_{1}, \sigma_{2}, \sigma_{3}, \sigma_{4}\right\}$, and the invariant function of $\hat{A}$ is the constant function $\lambda v \cdot[-20,20]^{2}$ (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 flow regions change. Figure 2 shows a sample trajectory of $\hat{A}$ from the initial zone Init $_{\hat{A}}=$ $\left\{\left(v_{1},(0,1)\right)\right\}$. Each arc is labeled with a vertex giving the discrete state, while the continuous state follows the arc.


FIG. 1. The initialized rectangular automaton $\hat{A}$.
The discontinuity between the arcs labeled $v_{2}$ and $v_{3}$ correspond to the jump of variable $d$ from -5 to -4 upon traversal of the edge from $v_{2}$ to $v_{3}$. The divergent timed word $\left(4 \sigma_{1} 1 \sigma_{2} 1 \sigma_{3} 1 \sigma_{4}\right)^{\omega}$ is an example of a timed word that is accepted by $\hat{A}$. This timed word is accepted by a run with the state sequence

$$
\begin{aligned}
& \left(\left(v_{1},(0,1)\right)\left(v_{1},(5,-10)\right)\left(v_{2},(4,-10)\right)\left(v_{2},(0,-12.5)\right)\right. \\
& \left.\quad\left(v_{3},(0,-4)\right)\left(v_{3},(-3,-2)\right)\left(v_{4},(-1,-2)\right)\left(v_{4},(0,0)\right)\right)^{\omega} .
\end{aligned}
$$

## CNF Edge Families

We sometimes annotate edges of rectangular automata with positive boolean combinations of guarded commands. Consider the two guarded commands $\phi_{1}$ and $\phi_{2}$. First, the edge annotation $\phi_{1} \wedge \phi_{2}$ stands for a guarded command $\phi_{3}$ with $\operatorname{pre}\left(\phi_{3}\right)=\operatorname{pre}\left(\phi_{1}\right) \cap \operatorname{pre}\left(\phi_{2}\right), \operatorname{post}\left(\phi_{3}\right)=\operatorname{post}\left(\phi_{1}\right) \cap \operatorname{post}\left(\phi_{2}\right)$, and $\operatorname{jump}\left(\phi_{3}\right)=j u m p\left(\phi_{1}\right) \cup j u m p\left(\phi_{2}\right)$. Second, an edge with the annotation $\phi_{1} \vee \phi_{2}$ stands for two edges that share source vertex, target vertex, and observation, one annotated with $\phi_{1}$ and the other with $\phi_{2}$. These conventions generalize to DNF expressions of guarded commands. An edge annotated with a CNF expression of guarded commands is interpreted by first converting the expression into DNF. A CNF edge family $((v, w), \sigma, \psi)$, then, consists of a pair $(v, w)$ of vertices, an observation $\sigma$, and a CNF expression $\psi$ of guarded


FIG. 2. A sample trajectory of $\hat{A}$.
commands. Consider, for example, the CNF edge family with the vertex pair ( $v, w$ ), the observation $\sigma$, and the CNF expression

$$
\begin{aligned}
((a<2 & \rightarrow a:=2) \vee(2 \leqslant a \leqslant 5)) \\
& \wedge((b>7 \rightarrow b:=7) \vee(4 \leqslant b \leqslant 7)) .
\end{aligned}
$$

This edge family corresponds to four edges from $v$ to $w$, each annotated with the observation $\sigma$ and one of the following guarded commands:

1. $a<2 \wedge b>7 \rightarrow a:=2 ; b:=7$,
2. $a<2 \wedge 4 \leqslant b \leqslant 7 \rightarrow a:=2$,
3. $2 \leqslant a \leqslant 5 \wedge b>7 \rightarrow b:=7$,
4. $2 \leqslant a \leqslant 5 \wedge 4 \leqslant b \leqslant 7$.

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

## Two Problems Concerning Rectangular Automata

We study the following two problems about rectangular automata.

Reachability. Given a rectangular automaton $A$, and a rectangular zone $Z_{f}$ of $A$, is $\operatorname{Reach}(A) \cap Z_{f}$ nonempty? That is, does $Z_{f}$ contain a reachable state of $A$ ? If so, we say that the zone $Z_{f}$ is reachable for $A$. 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 rectangular final zones, then the reachability problem is equivalent to the finitary language emptiness problem.
$\omega$-Language Emptiness. Given a rectangular automaton $A$, is $\operatorname{Lang}(A)$ 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 $\omega$-language emptiness problem. We then show that the reachability problem (and therefore $\omega$-language emptiness) is undecidable for very restricted classes of uninitialized rectangular automata and, also, for initialized automata
with slightly generalized invariant, flow, preguard, postguard, or jump 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 singular automata (Section 3.2), and from initialized singular automata to timed automata (Section 3.1). For the subclass of automata with bounded nondeterminism, the translation also preserves $\omega$-languages (Section 3.3), and therefore reduces the $\omega$-language emptiness problem for these automata to the corresponding problem for timed automata. 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 a practical implication of our translations, showing that symbolic execution [ $\mathrm{ACH}^{+} 95$ ] terminates on initialized rectangular automata after a linear-time preprocessing step.

### 3.1. From Initialized Singular Automata to Timed Automata

We begin by defining several special cases of rectangular automata for which, using known results about timed automata, the reachability and $\omega$-language emptiness problems can be solved easily.

## Finite-slope Variables

Consider a rectangular automaton $A$. The variable $c$ is a one-slope variable of $A$ if there exists a rational number $k$ such that for each vertex $v$ of $A$, we have $f l o w(v)(c)=[k, k]$. A one-slope variable with slope $k=0$ is called a memory cell. A one-slope variable with slope $k=1$ is called a clock. A one-slope variable with any other slope is called a skewed clock. Notice that, if every variable of $A$ is a one-slope variable, then $A$ is initialized. The variable $c$ is a two-slope variable of $A$ if there exist two rational numbers $k_{1}$ and $k_{2}$, with $k_{1} \neq k_{2}$, such that for each vertex $v$ of $A$, either flow $(v)(c)=\left[k_{1}, k_{1}\right]$ or $\operatorname{flow}(v)(c)=\left[k_{2}, k_{2}\right]$. A stopwatch is a two-slope variable with $k_{1}=1$ and $k_{2}=0$. The variable $c$ is a finite-slope variable of $A$ if for each vertex $v$ of $A$, the interval flow $(v)(c)$ is a singleton.

An $n$-dimensional rectangular automaton $A$ has deterministic jumps if (1) for every vertex $v$ of $A$, the region init $(v)$ is either empty or a singleton, and (2) for every edge $e$ of $A$ and every coordinate $i \in\{1, \ldots, n\}$ with $i \in j u m p(e)$, the interval post(e) is a singleton. The first requirement ensures that the number of initial states is finite. The second requirement says that along each edge step, each variable either remains unchanged or is deterministically assigned a new
value. Notice that, if $A$ has deterministic jumps and every variable of $A$ is a finite-slope variable, then $A$ has bounded nondeterminism.

## Timed Automata

A timed automaton $D$ is a rectangular automaton with deterministic jumps such that every variable of $D$ is a clock.

Theorem 3.1 [AD94]. The reachability and $\omega$-language emptiness problems for timed a automata (with or without $\varepsilon$ moves) are complete for PSPACE.

More precisely, the $\omega$-language emptiness problem for an $n$-dimensional timed automaton $D$ with $\varepsilon$ moves can be solved in space $O\left(\log \left(|V| \cdot n!\cdot K^{n}\right)\right)=O(\log |V|+n$. $(\log n+\log K)$ ), where the integer constant $K$ is determined by the rational numbers that are used in the definition of $D$, as finite endpoints of initial, invariant, preguard, and postguard intervals. If the definition of $D$ uses only nonnegative integer constants, then $K$ is the largest of these constants. If the definition of $D$ uses only nonnegative rational constants, let $\mathscr{K}$ be the set of these constants, and let $l$ be their least common denominator. Then $K=\max \{k \cdot l \mid k \in \mathscr{K}\}$. If the definition of $D$ uses negative rational constants, then subtract the constant with the least value from all constants, and compute $K$ from the resulting set of nonnegative numbers as in the nonnegative case. We assume that, in the definition of $D$, all constants are written in binary notation. It follows that the value of $K$ is at most singly exponential in the size of the definition of $D$, and hence, the above space requirement is polynomial. The reachability problem for a timed automaton $D$ and a rectangular zone $Z_{f}$ can be solved in the same amount of space, only that the constant $K$ must take into account also the finite endpoints of all intervals in the definition of $Z_{f}$.

We consider generalizations of timed automata. Therefore, all of our PSPACE hardness results follow from the corresponding hardness results for timed automata.

## Stopwatch Automata

A stopwatch automaton $C$ is a rectangular automaton with deterministic jumps such that every variable of $C$ is a stopwatch. Unlike timed automata, not every stopwatch automaton is initialized. We will see that for nonintialized stopwatch automata, the reachability problem is undecidable (Section 4.1). Initialized stopwatch automata, however, can be polynomially encoded by timed automata.

Let $C$ be an $n$-dimensional initialized stopwatch automaton with $\varepsilon$ moves, let $\mathscr{K}_{C}$ be the set of rational constants used in the definition of $C$, and let $\mathscr{K}_{\perp}=\mathscr{K}_{C} \cup\{\perp\}$. We define an $n$-dimensional timed automaton $D_{C}$ with the set $V_{D_{C}}=V_{C} \times \mathscr{K}_{\perp}^{\{1, \ldots, n\}}$ of vertices. Thus, each vertex $(v, f)$ of $D_{C}$ consists of a vertex $v$ from $C$ and a function $f$ from $\{1, \ldots, n\}$ to $\mathscr{K}_{\perp}$. Each state $q=((v, f), \mathbf{x})$ of $D_{C}$ is intended
to represent the state $\alpha(q)=(v, \mathbf{y})$ of $C$, where $y_{i}=x_{i}$ if $f(i)=\perp$, and $y_{i}=f(i)$ if $f(i) \neq \perp$. Intuitively, if the $i$ th stopwatch of $C$ is running (slope 1 ), then its value is tracked by the value of the $i$ th clock of $D_{C}$; if the $i$ th stopwatch is halted (slope 0 ), at value $k \in \mathscr{K}_{C}$, then this value is remembered by the vertices of $D_{C}$. Note that $\alpha: Q_{D_{C}} \rightarrow Q_{C}$ is an onto function from the states of $D_{C}$ to the states of $C$. It is extended to zones $Z \subset Q_{D_{C}}$ in the natural way, by $\alpha(Z)=$ $\bigcup_{q \in Z} \alpha(q)$.

It is useful to define also an "inverse" of the function $\alpha$. We define $-\alpha: Q_{C} \rightarrow Q_{D_{C}}$ so that $-\alpha(v, \mathbf{y})=((v, f), \mathbf{y})$, where for every coordinate $i \in\{1, \ldots, n\}$, if flow $_{C}(v)_{i}=[1,1]$ then $f(i)=\perp$, and if flow $_{C}(v)_{i}=[0,0]$ then $f(i)=y_{i}$. The function $-\alpha$ is a one-to-one function from the states of $C$ to the states of $D_{C}$, so that for each state $r$ of $C$, we have $\alpha(-\alpha(r))=r$. The map $-\alpha$ is extended to zones $Z \subset Q_{C}$ in the natural way. Then, for each zone $Z$ of $C$, we have $\alpha(-\alpha(Z))=Z$.

The remaining components of the timed automaton $D_{C}$ are defined as follows. The observation alphabet of $D_{C}$ is the same as for $C$. The initial zone of $D_{C}$ is $-\alpha\left(\right.$ Init $\left._{C}\right)$. For each vertex $(v, f)$ of $D_{C}, i n v_{D_{C}}(v, f)=i n v_{C}(v)$. For each edge $e=(v, w)$ of $C$, the timed automaton $D_{C}$ has all edges of the form $e^{\prime}=((v, f),(w, g))$, where for every coordinate $i \in\{1, \ldots, n\}$, either flow $_{C}(w)_{i}=[1,1]$ and $g(i)=\perp$, or flow $_{C}(v)_{i}=[1,1]$ and flow $_{C}(w)_{i}=[0,0]$ and $g(i)=\operatorname{post}_{C}(e)_{i}$, (recall that $C$ has deterministic jumps), or flow $_{C}(v)_{i}=$ $[0,0]$ and flow $_{C}(w)_{i}=[0,0]$ and $g(i)=f(i)$. Furthermore, $e$ and $e^{\prime}$ have the same preguard and postguard regions, jump sets, and observations. From these definitions, Lemma 3.2 follows immediately.

Lemma 3.2. Let C be an initialized stopwatch automaton with $\varepsilon$ moves. First, Init $_{C}=\alpha\left(\right.$ Init $\left._{D_{C}}\right)$. Second, for all states $q$ and $r$ of $D_{C}$, and every label $\tau \in \Sigma^{z} \cup\left[\mathbb{R}_{\geqslant 0}\right]$, if $q \xrightarrow{\tau}{ }_{D_{C}} r$, then $\alpha(q) \xrightarrow{\tau}{ }_{C} \alpha(r)$. Third, for all states $q$ of $D_{C}$, all states $r^{\prime}$ of $C$, and every label $\tau \in \Sigma^{\varepsilon} \cup\left[\mathbb{R}_{\geqslant 0}\right]$, if $\alpha(q) \xrightarrow{\tau}{ }_{C} r^{\prime}$, then there is a state $r$ of $D_{C}$ such that $q \xrightarrow{\tau}{D_{C}}_{C} r$ and $\alpha(r)=r^{\prime}$; similarly, if $r^{\prime} \xrightarrow{\tau} \alpha(q)$, then there is a state $r$ of $D_{C}$ such that $r \xrightarrow{\tau} D_{C} q$ and $\alpha(r)=r^{\prime}$.

From the second and third claims of Lemma 3.2, it follows that for every zone $Z$ of $D_{C}$, and every label $\tau \in \Sigma \cup \mathbb{R}_{\geqslant 0}$, $\operatorname{Post}_{C}^{\tau}(\alpha(Z))=\alpha\left(\operatorname{Post}_{D_{C}}^{\tau}(Z)\right)$ and $\operatorname{Pre}_{C}^{\tau}(\alpha(Z))=\alpha\left(\operatorname{Pre}_{D_{C}}^{\tau}(Z)\right)$. These relationships are illustrated in the commuting diagram
of Fig. 3. From the first claim of Lemma 3.2 and the fact that Post commutes with $\alpha$, we conclude that $\operatorname{Reach}(C)=$ $\alpha\left(\operatorname{Reach}\left(D_{C}\right)\right)$. From Lemma 3.2 it also follows that $\operatorname{Lang}(C)$ $=\operatorname{Lang}\left(D_{C}\right)$.
Theorem 3.3. The reachability and $\omega$-language emptiness problems for initialized stopwatch automata (with or without $\varepsilon$ moves) are complete for PSPACE.

Proof. It suffices to show containment in PSPACE. Define the constant $K_{C}$ as for timed automata. Notice that $\left|V_{D_{C}}\right| \leqslant\left|V_{C}\right| \cdot\left(K_{C}+1\right)^{n}$, and that the definition of $D_{C}$ uses the same constants as the definition of $C$. It follows that the space requirement $O\left(\log \left|V_{D_{C}}\right|+n \cdot\left(\log n+\log K_{D_{C}}\right)\right)=$ $O\left(\log \left|V_{C}\right|+n \cdot\left(\log n+\log K_{C}\right)\right)$ is polynomial in the size of $C$.

## Singular Automata

A singular automaton $B$ is a rectangular automaton with deterministic jumps such that every variable of $B$ is a finiteslope variable. Initialized singular automata can be rescaled to initialized stopwatch automata.

Let $B$ be an $n$-dimensional initialized singular automaton with $\varepsilon$ moves. We define an $n$-dimensional initialized stopwatch automaton $C_{B}$ with the same vertex set, edge set, and observation alphabet as $B$. Each state $q=(v, \mathbf{x})$ of $C_{B}$ is intended to represent the state $\beta(q)=\left(v, \beta_{v}(\mathbf{x})\right)$ of $B$, for the following definition of the bijections $\beta_{v}: \mathbb{R}^{n} \rightarrow \mathbb{R}^{n}$. For each vertex $v$ of $B$, if flow $_{B}(v)=\prod_{i=1}^{n}\left[k_{i}, k_{i}\right]$, then $\beta_{v}\left(x_{1}, \ldots, x_{n}\right)=$ $\left(l_{1} \cdot x_{1}, \ldots, l_{n} \cdot x_{n}\right)$, where $l_{i}=k_{i}$ if $k_{i} \neq 0$, and $l_{i}=1$ if $k_{i}=0$. Note that $\beta: Q_{C_{B}} \rightarrow Q_{B}$ is a bijection between the states of $C_{B}$ and the states of $B$, and can be viewed as a rescaling of the state space. Similar rescaling techniques for hybrid state spaces can be found in $\left[\mathrm{ACH}^{+} 95\right]$. The maps $\beta_{v}$ are extended to regions in the natural way, and the map $\beta$ is extended to zones in the natural way.

The remaining components of the initialized stopwatch automaton $C_{B}$ are defined as follows. For each vertex $v$ of $C_{B}$, init $_{C_{B}}(v)=\beta_{v}^{-1}\left(\right.$ init $\left._{B}(v)\right)$, inv ${ }_{C_{B}}(v)=\beta_{v}^{-1}\left(\right.$ inv $\left._{B}(v)\right)$, and for every coordinate $i \in\{1, \ldots, n\}$, flow $_{C_{B}}(v)_{i}=[0,0]$ if flow $_{B}(v)_{i}=[0,0]$, and flow $_{C_{B}}(v)_{i}=[1,1]$ if flow $_{B}(v)_{i} \neq$ $[0,0]$. For each edge $e=(v, w)$ of $C_{B}$, pre $_{C_{B}}(e)=\beta_{v}^{-1}\left(\right.$ pre $\left._{B}(e)\right)$, post $_{C_{B}}(e)=\beta_{w}^{-1}\left(\operatorname{post}_{B}(e)\right), \quad \operatorname{jump}_{C_{B}}(e)=\operatorname{jump}_{B}(e), \quad$ and $o b s_{C_{B}}(e)=o b s_{B}(e)$. From these definitions, Lemma 3.4 follows immediately.


FIG. 3. Commuting diagram for zones $Z$ of an initialized stopwatch automaton $C$.

Lemma 3.4. Let $B$ be an initialized singular automaton with $\varepsilon$ moves. First, Init $_{B}=\beta\left(\right.$ Init $\left._{C_{B}}\right)$. Second, for all states $q$ and $r$ of $C_{B}$, and every label $\tau \in \Sigma^{\varepsilon} \cup\left[\mathbb{R}_{\geqslant 0}\right]$, we have $q \xrightarrow{\tau}_{C_{B}} r$ iff $\beta(q) \xrightarrow{\tau}{ }_{B} \beta(r)$.

From the second claim of Lemma 3.4, it follows that for every zone $Z$ of $C_{B}$ and every label $\tau \in \Sigma \cup \mathbb{R}_{\geqslant 0}$, Post $_{B}^{\tau}(\beta(Z))$ $=\beta\left(\operatorname{Post}_{C_{B}}^{\tau}(Z)\right)$ and $\operatorname{Pr}_{B}^{\tau}(\beta(Z))=\beta\left(\operatorname{Pre}_{C_{B}}^{\tau}(Z)\right)$. These relationships are illustrated in the commuting diagram of Fig. 4. From the first claim of Lemma 3.4 and the fact that Post commutes with $\beta$, we conclude that $\operatorname{Reach}(B)=\beta\left(\operatorname{Reach}\left(C_{B}\right)\right)$. From Lemma 3.4 it also follows that $\operatorname{Lang}(B)=\operatorname{Lang}\left(C_{B}\right)$.

Theorem 3.5. The reachability and $\omega$-language emptiness problems for initialized singular automata (with or without $\varepsilon$ moves) are complete for PSPACE.

Proof. It suffices to show containment in PSPACE. Consider the case that the definition of $B$ uses only nonnegative integer constants; the general case is similar to the analysis of timed automata with rational and negative constants. As for timed automata, define $K_{B}$ to be the maximum over all finite endpoints of initial, invariant, preguard, and postguard intervals. Define $L_{B}$ to be the product of $K_{B}$ and the least common multiple of all finite endpoints of flow intervals. Then the value of $L_{B}$ is at most singly exponential in the size of the definition of $B$, and $K_{C_{B}} \leqslant L_{B}$. It follows that the space requirement $O\left(\log \left|V_{C_{B}}\right|+n \cdot\left(\log n+\log K_{C_{B}}\right)\right)=$ $O\left(\log \left|V_{B}\right|+n \cdot\left(\log n+\log L_{B}\right)\right)$ is polynomial in the size of $B$.

### 3.2. From Initialized Rectangular Automata to Initialized Singular Automata

Let $A$ be an $n$-dimensional initialized rectangular automaton. We translate $A$ into a $(2 n+1)$-dimensional initialized singular automaton $B_{A}$ with $\varepsilon$ moves so that $B_{A}$ contains all reachability information about $A$. (While we assume, for simplicity, that the given automaton $A$ has no $\varepsilon$ moves, the translation and its results apply also to initialized rectangular automata with $\varepsilon$ moves.) The translation is similar to the subset construction for determinizing finite automata. 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, but the general case requires additional bookkeeping.

Preliminary assumptions. Without loss of generality, we assume that for each vertex $v$ of $A$, $\operatorname{init}_{A}(v) \subset i n v_{A}(v)$, and for each edge $e=(v, w)$ of $A$, pre $_{A}(e) \subset \operatorname{inv}_{A}(v), \operatorname{post}_{A}(e) \subset \operatorname{inv}_{A}(w)$, and for every coordinate $i \notin j u m p_{A}(e)$, pre $_{A}(e)_{i}=\operatorname{post}_{A}(e)_{i}$. If this is not the case, then first we replace each initial and guard region by its intersection with the appropriate invariant region, and second we replace each guard interval $\operatorname{pre}_{A}(e)_{i}$ and post $_{A}(e)_{i}$ with $i \notin j u m p_{A}(e)$ by their intersection $\operatorname{pre}_{A}(e)_{i} \cap \operatorname{post}_{A}(e)_{i}$. These replacements do not change the labeled transition system of $A$.

## The Compact Case

We first restrict our attention to the case where $\operatorname{inv}_{A}$ is the trivial invariant function $\lambda v . \mathbb{R}^{n}$, and all initial, flow, preguard, and postguard regions of $A$ are compact. In this case, we say that the automaton $A$ is compact.

In the compact case, we translate $A$ into a $2 n$-dimensional initialized singular automaton denoted $B_{A}^{c}$. The idea is to replace each variable $c$ of $A$ by two finite-slope variables $c_{\ell}$ and $c_{u}$ such that when flow $_{A}(v)(c)=\left[k_{\ell}, k_{u}\right]$, then flow $_{B_{A}^{c}}(v)\left(c_{\ell}\right)$ $=\left[k_{\ell}, k_{\ell}\right]$ and flow $_{B_{A}^{c}}(v)\left(c_{u}\right)=\left[k_{u}, k_{u}\right]$. Intuitively, the variable $c_{\ell}$ tracks the least possible value of $c$, and $c_{u}$ tracks the greatest possible value of $c$. Consider Fig. 5. With each time step, the flow interval of $c$ creates an envelope, whose boundaries are tracked by $c_{\ell}$ and $c_{u}$. With each edge step, the values of $c_{\ell}$ and $c_{u}$ are updated so that the interval [ $c_{\ell}, c_{u}$ ] remains precisely the range of possible values of $c$. In Fig. 5, at time $t$ a jump transition is taken along an edge $e$ with pre $_{A}(e)(c)=[p, \infty)$ and $c \notin j u m p_{A}(e)$. Since the value of $c_{\ell}$ is below $p$ at time $t$, the variable $c_{\ell}$ is updated to the new value $p$.

In the following formal definition of the automaton $B_{A}^{c}$, we use $\left\{a_{i} \mid 1 \leqslant i \leqslant n\right\}$ for the variables of $A$, and $\left\{b_{i} \mid 1 \leqslant\right.$ $i \leqslant 2 n\}$ for the variables of $B_{A}^{c}$. The $\ell(i)$ th coordinate $b_{\ell(i)}$ of $B_{A}^{c}$ represents the lower bound on the $i$ th coordinate $a_{i}$ of $A$, and the $u(i)$ th coordinate $b_{u(i)}$ of $B_{A}^{c}$ represents the upper bound on $a_{i}$. For concreteness, put $\ell(i)=2 i-1$ and $u(i)=2 i$.

The $2 n$-dimensional automaton $B_{A}^{c}$ has the same vertex set and observation alphabet as $A$. The initial function init $_{B_{A}^{c}}$ is given by init $_{B_{A}^{c}}(v)_{\ell(i)}=\min \left(\right.$ init $\left._{A}(v)_{i}\right)$ and init $_{B_{A}^{c}}(v)_{u(i)}=$ $\max \left(\right.$ init $\left._{A}(v)_{i}\right)$. The invariant function $\operatorname{inv}_{B_{A}^{c}}$ is the trivial invariant function. The flow function flow $_{B_{A}^{c}}$ is defined as


FIG. 4. Commuting diagram for zones $Z$ of an initialized singular automaton $B$.


FIG. 5. Envelope created by the flow interval flow $(v)(c)=\left[k_{\ell}, k_{u}\right]$.
outlined above: if flow $_{A}(v)_{i}=\left[k_{\ell}, k_{u}\right]$, then flow $_{B_{A}^{c}}(v)_{\ell(i)}=$ $\left[k_{\ell}, k_{\ell}\right]$ and flow $_{B_{A}^{c}}(v)_{u(i)}=\left[k_{u}, k_{u}\right]$. We are left with defining a set of CNF edge families for $B_{A}^{c}$. For each edge $e=(v, w)$ of $A$, the automaton $B_{A}^{c}$ has the CNF edge family $\Psi_{e}=\left(v, w, o b s_{A}(e), \psi_{e}\right)$, which shares the observation of $e$. The CNF expression $\psi_{e}$ is a conjunction $\bigwedge_{i=1}^{n} \psi_{e}^{i}$ of $n$ CNF expressions $\psi_{e}^{i}$. Suppose that $\operatorname{pre}_{A}(e)_{i}=\left[p_{\ell}, p_{u}\right]$ and post $_{A}(e)_{i}=\left[p_{\ell}^{\prime}, p_{u}^{\prime}\right]$. If $i \in j u m p_{A}(e)$, then the CNF expression $\psi_{e}^{i}$ is the guarded command

$$
b_{\ell(i)} \leqslant p_{u} \wedge b_{u(i)} \geqslant p_{\ell} \quad \rightarrow \quad b_{\ell(i)}:=p_{\ell}^{\prime} ; b_{u(i)}:=p_{u}^{\prime} .
$$

The values of $b_{\ell(i)}$ and $b_{u(i)}$ satisfy the guard iff $\left[b_{\ell(i)}, b_{u(i)}\right]$ intersects $\operatorname{pre}_{A}(e)_{i}$. Since $i \in$ jump $_{A}(e)$, the range of values of $a_{i}$ after traversing $e$ is exactly post $_{A}(e)_{i}$, and hence $b_{\ell(i)}$ is set to the minimum of this interval, and $b_{u(i)}$ is set to the maximum. If $i \notin j u m p_{A}(e)$, then by assumption $\left[p_{\ell}, p_{u}\right]=\left[p_{\ell}^{\prime}, p_{u}^{\prime}\right]$, and the CNF expression $\psi_{e}^{i}$ is

$$
\begin{aligned}
\left(\left(b_{\ell(i)}\right.\right. & \left.\left.<p_{\ell} \rightarrow b_{\ell(i)}:=p_{\ell}\right) \vee\left(p_{\ell} \leqslant b_{\ell(i)} \leqslant p_{u}\right)\right) \\
& \wedge\left(\left(b_{u(i)}>p_{u} \rightarrow b_{u(i)}:=p_{u}\right) \vee\left(p_{\ell} \leqslant b_{u(i)} \leqslant p_{u}\right)\right) .
\end{aligned}
$$

The idea is that when the edge $e$ is traversed in $A$, new information becomes available about the value of $a_{i}$, namely, that it lies within the interval $\left[p_{\ell}, p_{u}\right]$. Therefore, if $b_{\ell(i)}<p_{\ell}$, it must be updated to $p_{\ell}$, and if $b_{u(i)}>p_{u}$, it must be updated to $p_{u}$, in order to keep $\left[b_{\ell(i)}, b_{u(i)}\right]$ the range of possible values of $a_{i}$.

This completes the definition of the automaton $B_{A}^{c}$. The automaton $B_{A}^{c}$ is an initialized singular automaton. Figure 6 shows the initialized singular automaton $B_{A}^{c}$ that corresponds to the initialized rectangular automaton $\hat{A}$ from Fig. 1. Figure 7 shows the initialized stopwatch automaton $C_{B_{A}^{c}}$ that corresponds to $\hat{A}$. Notice that $C_{B_{A}^{c}}$ is a timed automaton.

The relationship between $A$ and $B_{A}^{c}$ is formalized through the function $\gamma^{c}: Q_{B_{A}^{c}} \rightarrow 2^{Q_{A}}$, which maps each state of $B_{A}^{c}$ to a zone of $A$ : define $\gamma^{c}(v, \mathbf{x})=\{v\} \times \prod_{i=1}^{n}\left[x_{\ell(i)}, x_{u(i)}\right]$. The map $\gamma^{c}$ is extended to zones $Z \subset Q_{B_{A}^{c}}$ by $\gamma^{c}(Z)=\bigcup_{q \in Z} \gamma^{c}(q)$. Not all states of $B_{A}^{c}$ are of interest. The state $(v, \mathbf{x}) \in Q_{B_{A}^{c}}$ is
an upper-half state of $B_{A}^{c}$ if for every coordinate $i \in\{1, \ldots, n\}$, we have $x_{\ell(i)} \leqslant x_{u(i)}$. Notice that $q$ is an upper-half state of $B_{A}^{c}$ iff $\gamma^{c}(q) \neq \varnothing$. The upper-half space of $B_{A}^{c}$, denoted $U_{B_{A}^{c}}$, is the set of all upper-half states of $B_{A}^{c}$. Lemma 3.6 shows that, for reachability purposes, each upper-half state $q$ of $B_{A}^{c}$ represents the set $\gamma^{c}(q) \subset Q_{A}$ of states of $A$.

It is useful to define also an "inverse" of the function $\gamma^{c}$. For a compact rectangular region $R \subset \mathbb{R}^{n}$, define the vector $-\gamma^{c}(R) \in \mathbb{R}^{2 n}$ by $-\gamma^{c}(R)_{\ell(i)}=\min \left(R_{i}\right)$ and $-\gamma^{c}(R)_{u(i)}=$ $\max \left(R_{i}\right)$; that is, $-\gamma^{c}(R)$ is the vector of all boundary points of the intervals $R_{i}$. For a compact rectangular zone $Z$ of $A$, define $-\gamma^{c}(Z)$ to be the zone $\bigcup_{v \in V_{A}}\left\{\left(v,-\gamma^{c}\left([Z]^{v}\right)\right)\right\}$ of $B_{A}^{c}$. Notice that $-\gamma(Z) \subset U_{B_{A}^{c}}$, and that Init $_{B_{A}^{c}}=-\gamma^{c}\left(\right.$ Init $\left._{A}\right)$.

Lemma 3.6. Let $A$ be a compact initialized rectangular automaton. First, for every compact rectangular zone $Z$ of $A$, we have $\gamma^{c}\left(-\gamma^{c}(Z)\right)=Z$; in particular, Init $_{A}=\gamma^{c}\left(\right.$ Init $\left._{B_{A}^{c}}\right)$. Second, for every upper-half state $q$ of $B_{A}^{c}$, and every label $\tau \in \Sigma \cup \mathbb{R}_{\geqslant 0}$, Post $_{A}^{\tau}\left(\gamma^{c}(q)\right)=\gamma^{c}\left(\operatorname{Post}_{B_{A}^{c}}^{\tau}(q)\right)$.

We delay a detailed proof of these claims to the general case. From the second claim of Lemma 3.6, it follows that for every upper-half zone $Z \subset U_{B_{A}^{c}}$ and every label $\tau \in \Sigma \cup \mathbb{R}_{\geqslant 0}$, $\operatorname{Post}_{A}^{\tau}\left(\gamma^{c}(Z)\right)=\gamma^{c}\left(\operatorname{Post}_{B_{A}^{c}}^{\tau}(Z)\right)$ and, by Proposition 2.1, $\operatorname{Pre}_{A}^{\tau}\left(\gamma^{c}(Z)\right)=\gamma^{c}\left(\operatorname{Pre}_{-B_{-A}^{c}}^{\tau}(Z)\right)$. These relationships are illustrated in the commuting diagram of Fig. 8. From the first claim of Lemma 3.6 and the fact that Post commutes with $\gamma^{c}$, we conclude that $\operatorname{Reach}(A)=\gamma^{c}\left(\operatorname{Reach}\left(B_{A}^{c}\right)\right)$.

Theorem 3.7. The reachability problem for compact initialized rectangular automata is complete for PSPACE.

Indeed, since Init $_{A}=\gamma^{c}\left(\right.$ Init $\left._{B_{A}^{c}}\right)$ and Post commutes with $\gamma^{c}$, we also have the following. For every run of $A$ we can, starting from the beginning of the run, inductively construct a run of $B_{A}^{c}$ that accepts the same timed word. Therefore $\operatorname{Lang}(A) \subset$ $\operatorname{Lang}\left(B_{A}^{c}\right)$. Furthermore, for every finite run of $B_{A}^{c}$ we can, starting from the end of the run, inductively construct a run of $A$ that accepts the same timed word. Therefore $A$ and $B_{A}^{c}$ accept the same finite timed words.

## The General Case

The extension from the compact to the general case is mostly a matter of bookkeeping. In particular, for each lower-bound variable $b_{\ell(i)}$ and upper-bound variable $b_{u(i)}$ one bit is used to distinguish a weak from a strict bound, and a second bit is used to distinguish a finite from an infinite bound. The reader who is uninterested in the details can skip ahead to Theorem 3.18 without loss of continuity (in this case, the reader should know that $B_{A}$ is the generalization of $B_{A}^{c}$, and $\gamma$ is the generalization $\gamma^{c}$ ).

We encounter the following difficulties when the rectangular automaton $A$ has noncompact components and a nontrivial invariant function. (1) The lower-bound and


FIG. 6. The initialized singular automaton $B_{A}^{c}$.
upper-bound variables $b_{\ell(i)}$ and $b_{u(i)}$ may violate the invariant region of a vertex, and the function $\gamma$ must be changed to account for this. (2) The lower and upper bounds represented by $b_{\ell(i)}$ and $b_{u(i)}$ may be weak (inclusive) or strict (exclusive). To solve this problem, we introduce a bit called the weak/strict bit for each variable of $B_{A}$. (3) Lower and upper bounds may be finite or infinite. For this, we introduce a bit called the finite/infinite bit for each variable of $B_{A}$. (4) Unbounded flow regions cause discontinuous jumps in lower and upper bounds. For example, suppose that the variable $a_{i}$ is assigned the value 3 by traversal of the edge $e=(v, w)$, where flow $(w)_{i}=[1, \infty)$. Then in $B_{A}$, both $b_{\ell(i)}$ and $b_{u(i)}$ are assigned the value 3 by traversal of $e$. But after any positive amount of time passes, the upper bound on $a_{i}$ should be $\infty$. For this, we introduce an $\varepsilon$ edge, which is taken before any positive time step. The $\varepsilon$ edge sets the finite/infinite bit for
$b_{u(i)}$ to infinite. Since the result of the $\varepsilon$ edge presupposes that some positive amount of time has passed, all edge steps inherited from $A$ are disabled in $B_{A}$ until time passes. Implementing this restriction requires a new clock called the synchronization clock and a bit called the time-passage bit. (5) Strict flow regions may cause a weak bound to change to strict after any positive amount of time passes. For example, suppose in the above case that $\operatorname{flow}(w)_{i}=[1,5)$. Then after the edge $e$ is traversed, the upper bound on $a_{i}$ is a weak bound of 3 . But after $t>0$ time units pains, the upper bound is a strict bound of $3+5 t$. Once again, we use the $\varepsilon$ edge to solve this problem. When the $\varepsilon$ edge is traversed, the weak/ strict bit for $b_{u(i)}$ is set to strict.

We now proceed formally to define the $(2 n+1)$-dimensional initialized singular automaton $B_{A}$ with $\varepsilon$ moves. The observation alphabet of $B_{A}$ is the same as for $A$. We define


FIG. 7. The initialized stopwatch automaton (and timed automaton) $C_{B_{A}^{c}}$.
first the continuous state (variables) and discrete state (vertices) of $B_{A}$ and then the flow and invariant functions. Next we define the map $\gamma$, which relates states of $B_{A}$ to zones of $A$. Then come the edges of $B_{A}$, which are classified into $\varepsilon$ edges and edges inherited from $A$. Last, we define the initial function. We provide lemmas about $B_{A}$ as soon as enough definitions have been made to give the proofs.

Variables and vertices of $B_{A}$. For each variable $a_{i}$ of $A$, the automaton $B_{A}$ has two variables, $b_{\ell(i)}$ and $b_{u(i)}$. We add a zeroth coordinate $b_{0}$ to $B_{A}$ which represents the synchronization clock. Hence the dimension of $B_{A}$ is $2 n+1$. The finite/infinite and weak/strict bitvectors and the timepassage bit are encoded in the vertex set of $B_{A}$. Therefore, $V_{B_{A}}=V_{A} \times\left(\{0,1\}^{2 n}\right)^{2} \times\{0,1\}$. A state $((v, \boldsymbol{\mu}, \boldsymbol{v}, t p), \mathbf{x})$ of $B_{A}$ then consists of a vertex $v$ of $A$, a vector $\boldsymbol{\mu}$ of $2 n$ finite/ infinite bits ( $\operatorname{fin}=0, \inf =1$ ), a vector $\boldsymbol{v}$ of $2 n$ weak/strict bits ( $w k=0, s t r=1$ ), a time-passage bit $t p$, and a vector $\mathbf{x} \in \mathbb{R}^{2 n+1}$ of values for the $2 n+1$ variables of $B_{A}$.

Notation. For the remainder of Section 3.2, $i$ ranges over $\{1, \ldots, n\}$, the symbol $v$ ranges over $V_{A}$, the vectors $\boldsymbol{\mu}$ and $\boldsymbol{v}$ range over $\{0,1\}^{2 n}$, and $t p$ ranges over $\{0,1\}$. So when we implicitly or explicitly quantify these variables, as in "for all $v, \boldsymbol{\mu}, \boldsymbol{v}, t p, i$, " the quantification is over the domains just specified. Recall that for a zone $Z$ of $A$, we have the canonical decomposition $Z=\bigcup_{v}[Z]^{v}$. We lift the functions [ $\cdot]^{v}$ to the zones $Z^{\prime}$ of $B_{A}$ by defining $\left[Z^{\prime}\right]^{v}=\left\{\mathbf{x} \in \mathbb{R}^{2 n+1} \mid \exists \boldsymbol{\mu}, \mathbf{v}, t p .((v, \boldsymbol{\mu}, \mathbf{v}, t p), \mathbf{x}) \in Z^{\prime}\right\}$. We say that the vector $\mathbf{x} \in \mathbb{R}^{2 n+1}$ satisfies a predicate $\varphi$ over the variables $b_{0}, b_{1}, \ldots, b_{2 n}$ iff $\varphi$ evaluates to true when each variable $b_{i}$ is replaced by the $i$ th component of $\mathbf{x}$. Finally, consider an interval $I \subset \mathbb{R}$ of the real line. We define the lower strictness of $I$ by strict $\downarrow(I)=w k$ if $\inf (I) \in I$, and strict $\downarrow(I)=s t r$ if $\inf (I) \notin I$. Similarly, we define the upper strictness of $I$ by strict $\uparrow(I)=w k$ if $\sup (I) \in I$ and strict $\uparrow(I)$ $=\operatorname{str}$ if $\sup (I) \notin I$.

Flow regions of $B_{A}$. The flow function flow $_{B_{A}}$ is defined by

$$
\begin{aligned}
& \text { flow }_{B_{A}}(v, \boldsymbol{\mu}, \boldsymbol{v}, t p)_{\ell(i)}=\left\{\begin{array}{l}
\left\{\inf \left(\text { flow }_{A}(v)_{i}\right)\right\}, \\
\quad \text { if } \inf \left(\text { flow }_{A}(v)_{i}\right) \neq-\infty, \\
\{0\}, \quad \text { otherwise } ;
\end{array}\right. \\
& \text { flow }_{B_{A}}(v, \boldsymbol{\mu}, \boldsymbol{v}, t p)_{u(i)}=\left\{\begin{array}{l}
\left\{\sup \left(\text { flow }_{A}(v)_{i}\right)\right\}, \\
\text { if } \sup \left(\text { flow }_{A}(v)_{i}\right) \neq \infty, \\
\{0\}, \\
\text { otherwise. }
\end{array}\right.
\end{aligned}
$$

The slope of $b_{\ell(i)}\left(\right.$ resp. $\left.b_{u(i)}\right)$ in $B_{A}$ is the infimum (resp. supremum) of the allowable slopes for $a_{i}$ in $A$, unless that infimum (resp. supremum) is infinite. The zeroth coordinate $b_{0}$ is a clock: define flow $_{B_{A}}(v, \boldsymbol{\mu}, \boldsymbol{v}, t p)_{0}=\{1\}$.

Invariant regions of $B_{A}$. The lower-bound components of the invariant function $\operatorname{inv}_{B_{A}}$ are defined by

$$
\begin{aligned}
& \operatorname{inv}_{B_{A}}(v, \boldsymbol{\mu}, \boldsymbol{v}, t p)_{\ell(i)} \\
& \quad= \begin{cases}(-\infty, \infty), & \text { if } \mu_{\ell(i)}=\inf , \\
\left(-\infty, \sup \left(i n v_{A}(v)_{i}\right)\right], & \text { if } \mu_{\ell(i)}=\text { fin and } v_{\ell(i)}= \\
& \text { strict } \uparrow\left(i n v_{A}(v)_{i}\right)=w k, \\
\left(-\infty, \sup \left(i n v_{A}(v)_{i}\right)\right), & \text { otherwise. }\end{cases}
\end{aligned}
$$

If the finite/infinite bit $\mu_{\ell(i)}$ is infinite, then the value of the lower-bound variable $b_{\ell(i)}$ is irrelevant, so we do not constrain it. If the finite/infinite bit is finite, then the interval of allowable values for $b_{\ell(i)}$ is right-open unless both the weak/strict bit $v_{\ell(i)}$ is weak and the invariant interval $i n v_{A}(v)_{i}$ is right-closed. The motivation for this is that if $I$ and $J$ are intervals, and $\inf (I)=\sup (J)$, then $I \cap J \neq \varnothing$ iff strict $\downarrow(I)=\operatorname{strict} \uparrow(J)=w k$. Here $I$ is meant to represent the range of possible values for $a_{i}$ in $A$, as determined by the state of $B_{A}$, and $J$ is meant to represent the invariant interval $i n v_{A}(v)_{i}$. The corresponding definition for the upper-bound components of $i n v_{B_{A}}$ is given by

$$
\begin{aligned}
& \operatorname{inv}_{B_{A}}(v, \boldsymbol{\mu}, \boldsymbol{v}, t p)_{u(i)} \\
& \left.\quad=\left\{\begin{array}{ll}
(-\infty, \infty), & \text { if } \mu_{u(i)}=\inf , \\
{\left[\inf \left(\operatorname{inv}_{A}(v)_{i}\right), \infty\right),} & \text { if } \mu_{u(i)}=\text { fin and } v_{u(i)}= \\
\left(\operatorname { s t r i c t } \downarrow \left(\inf \left(\operatorname{inv}_{A}(v)_{i}\right)=w k,\right.\right.
\end{array}\right), \infty\right), \\
& \text { otherwise. }
\end{aligned} .
$$

The invariant intervals for the synchronization clock $b_{0}$ let time pass iff the time-passage bit $t p$ is 1 : define $\operatorname{inv}_{B_{A}}(v, \boldsymbol{\mu}, \boldsymbol{v}, 0)_{0}$ $=[0,0]$ and $\operatorname{inv}_{B_{A}}(v, \boldsymbol{\mu}, \boldsymbol{v}, 1)_{0}=[0, \infty)$.

Relating the state spaces of $B_{A}$ and $A$. The function $\gamma^{\prime}: Q_{B_{A}} \rightarrow 2^{V_{A} \times \mathbb{R}^{n}}$ specifies how each state of $B_{A}$ indicates a range of possible values for each variable of $A$. Consider a state $q=((v, \boldsymbol{\mu}, \boldsymbol{v}, t p), \mathbf{x})$ of $B_{A}$. We define $\gamma^{\prime}(q)=\{v\} \times \gamma_{2}^{\prime}(q)$, where the function $\gamma_{2}^{\prime}: Q_{B_{A}} \rightarrow 2^{\mathbb{R}^{n}}$ maps $q$ to a rectangular region in $\mathbb{R}^{n}$. For every coordinate $i \in\{1, \ldots, n\}$, the interval $\gamma_{2}^{\prime}(q)_{i}$ is completely specified by its infimum, supremum, and which, if any, of the two it contains:

- If $\mu_{\ell(i)}=\inf$, then $\inf \left(\gamma_{2}^{\prime}(q)_{i}\right)=-\infty$; if $\mu_{\ell(i)}=f i n$, then $\inf \left(\gamma_{2}^{\prime}(q)_{i}\right)=x_{\ell(i)}$.
- If $\mu_{u(i)}=\inf$, then $\sup \left(\gamma_{2}^{\prime}(q)_{i}\right)=\infty$; if $\mu_{u(i)}=f i n$, then $\sup \left(\gamma_{2}^{\prime}(q)_{i}\right)=x_{u(i)}$.
- If $\mu_{\ell(i)}=f i n$, then $\inf \left(\gamma_{2}^{\prime}(q)_{i}\right) \in \gamma_{2}^{\prime}(q)_{i}$ iff $v_{\ell(i)}=w k$.
- If $\mu_{u(i)}=f i n$, then $\sup \left(\gamma_{2}^{\prime}(q)_{i}\right) \in \gamma_{2}^{\prime}(q)_{i}$ iff $v_{u(i)}=w k$.

Since we have not taken into account the invariant function of $A$, we may have $\gamma^{\prime}(q) \not \subset Q_{A}$ for some states $q$ of $B_{A}$. We remedy this deficiency by defining $\gamma: Q_{B_{A}} \rightarrow 2^{Q_{A}}$ by $\gamma(q)=\gamma^{\prime}(q) \cap Q_{A}$ if $t p=0$ or $x_{0}>0$, and $\gamma(q)=\varnothing$ if $t p=1$ and $x_{0}=0$. The latter adjustment is necessary because the
states of $B_{A}$ in which the time-passage bit is 1 and the synchronization clock is 0 are visited only transiently between an $\varepsilon$ edge and a positive time step, and they do not correspond to any states of $A$. The function $\gamma_{2}: Q_{B_{A}} \rightarrow 2^{\mathbb{R}^{n}}$ strips off the vertex; it is defined by the equation $\gamma(q)=$ $\{v\} \times \gamma_{2}(q)$. The functions $\gamma, \gamma^{\prime}, \gamma_{2}$, and $\gamma_{2}^{\prime}$ are extended to zones $Z \subset Q_{B_{A}^{c}}$ by $\gamma(Z)=\bigcup_{q \in Z} \gamma(q)$, etc. The state $q$ is an upper-half state of $B_{A}$ if $\gamma(q) \neq \varnothing$. The upper-half space $U_{B_{A}}$ is the set of all upper-half states of $B_{A}$. The truncation of $\gamma^{\prime}$ to $\gamma$ is justified by the following fact, which follows from the assumption that the preguard region of each edge of $A$ is contained in the invariant region of the source vertex.

Fact 3.8. For every edge $e$ of an initialized rectangular automaton $A$, and every upper-half state $q$ of $B_{A}$, we have $\gamma_{2}^{\prime}(q) \cap \operatorname{pre}_{A}(e) \neq \varnothing$ iff $\gamma_{2}(q) \cap \operatorname{pre}_{A}(e) \neq \varnothing$.

Edges of $B_{A}$. As in the compact case, the automaton $B_{A}$ 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, $B_{A}$ has a set of $\varepsilon$ edges. The $\varepsilon$ edges provide changes to the finite/infinite and weak/strict bitvectors that are caused by the passage of any positive amount of time, however small. Such changes can be carried out only through edge steps.

Epsilon edges. We first define the $\varepsilon$ edges of $B_{A}$. For all $v, \boldsymbol{\mu}, \boldsymbol{v}$, there is an edge $e$ from $(v, \boldsymbol{\mu}, \boldsymbol{v}, 0)$ to $\left(v, \boldsymbol{\mu}^{\prime}, \boldsymbol{v}^{\prime}, 1\right)$ with observation $\varepsilon$. The target vertex components $\boldsymbol{\mu}^{\prime}$ and $\boldsymbol{v}^{\prime}$ will be defined presently. The preguard and postguard regions of $e$ are both $[0,0] \times \mathbb{R}^{2 n}$, so this edge can be traversed only when both the time-passage bit and the synchronization clock have the value 0 . The update set of $e$ is empty. The finite/infinite bitvector must be changed to account for finite bounds that become infinite due to an unbounded flow interval:

$$
\begin{aligned}
& \mu_{\ell(i)}^{\prime}= \begin{cases}\inf , & \text { if } \inf \left(\text { flow }_{A}(v)_{i}\right)=-\infty, \\
\mu_{\ell(i)}, & \text { otherwise }\end{cases} \\
& \mu_{u(i)}^{\prime}= \begin{cases}\inf , & \text { if } \sup \left(\text { flow }_{A}(v)_{i}\right)=\infty, \\
\mu_{u(i)}, & \text { otherwise }\end{cases}
\end{aligned}
$$

The weak/strict bitvector must be changed to account for weak bounds that become strict due to a strict flow interval:

$$
\begin{aligned}
& v_{\ell(i)}^{\prime}= \begin{cases}\text { str, }, & \text { if } \text { strict } \downarrow\left(\text { flow }_{A}(v)_{i}\right)=s t r, \\
v_{\ell(i)}, & \text { otherwise } ;\end{cases} \\
& v_{u(i)}^{\prime}= \begin{cases}\text { str, } & \text { if } \operatorname{strict} \uparrow\left(\text { flow }_{A}(v)_{i}\right)=\text { str, } \\
v_{u(i)}, & \text { otherwise. } .\end{cases}
\end{aligned}
$$

The $\varepsilon$ edges, just defined, play the following role in $B_{A}$. Suppose that an edge inherited from $A$ is traversed. Then $t p=0$ and $b_{0}=0$, and before any time may pass (since no
time may pass when $t p=0$ ), an $\varepsilon$ edge must be traversed, setting $t p$ to 1 , and performing whatever bookkeeping is required for the finite/infinite and weak/strict bitvectors. The changes made by the $\varepsilon$ edge reflect the situation after some infinitesimal positive amount of time has passed. Therefore no edge inherited from $A$ is allowed after an $\varepsilon$ edge but before a positive time step: as long as $t p=1$ and $b_{0}=0$, no inherited edges are enabled. After the next positive time step, $t p=1$ and $b_{0}>0$, and another inherited edge may be traversed, resetting both $t p$ and $b_{0}$ to 0 . Then the situation repeats.

Inherited edges. We now define the edges of $B_{A}$ that are inherited from $A$. For this purpose, it is convenient to extend the definition of CNF edge families to allow multiple target vertices. For example, in a guarded command, we may write $\mu_{\ell(i)}:=$ inf to change the $\ell(i)$ th component of the finite/infinite bitvector $\boldsymbol{\mu}$ to inf. In this way, a disjunction of guarded commands can refer to several target vertices. An extended CNF edge family for $B_{A}$ 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 that includes assignments to the bitvectors $\boldsymbol{\mu}$ and $\boldsymbol{v}$. The translation of such an extended CNF edge family into a set of edges for $B_{A}$ is a straightforward extension of the translation for standard CNF edge families and will not be detailed.

For each edge $e=(v, w)$ of $A$, all bitvectors $\boldsymbol{\mu}$ and $\boldsymbol{v}$, and each bit $t p$, the automaton $B_{A}$ has the extended CNF edge family $\Psi_{e, \boldsymbol{\mu}, \boldsymbol{v}, t p}=\left((v, \boldsymbol{\mu}, \boldsymbol{v}, t p), o b s_{A}(e), w, 0, \psi_{e, \boldsymbol{\mu}, \mathbf{v}, t p}\right)$. Every edge derived from the family $\Psi_{e, \boldsymbol{\mu}, \mathbf{v}, t p}$ shares the observation label of $e$ (which, by assumption, is different from $\varepsilon$ ), and leads to a target vertex of the form ( $w, \boldsymbol{\mu}^{\prime}, \boldsymbol{v}^{\prime}, 0$ ). The CNF expression $\psi_{e, \boldsymbol{\mu}, \boldsymbol{v}, t p}$ is a conjunction $\phi_{t p} \wedge$ $\bigwedge_{i=1}^{n} \psi_{e, \boldsymbol{\mu}, \boldsymbol{v}}^{i}$ of $n+1 \mathrm{CNF}$ expressions. The guarded command $\phi_{0}$ is $b_{0}=0$, and the guarded command $\phi_{1}$ is $b_{0}>0 \rightarrow b_{0}:=0$. Hence an inherited edge from a state with the time-passage bit $t p=1$ can be taken only if the synchronization clock has a positive value. Upon traversal of the edge, both the timepassage bit and the synchronization clock are reset to 0 .

It remains to define the CNF expressions $\psi_{e, \boldsymbol{\mu}, \boldsymbol{v}}^{i}$. We define their guards so that an edge of $B_{A}$ derived from the edge family $\Psi_{e, \boldsymbol{\mu}, \boldsymbol{v}, t p}$ can be taken from an upper-half state $q$ of $B_{A}$ iff the range of possible values for each variable $a_{i}$ intersects the interval pre $_{A}(e)_{i}$, that is, iff $\gamma_{2}(q) \cap \operatorname{pre}_{A}(e) \neq \varnothing$. Recall that by Fact 3.8, we have $\gamma_{2}(q) \cap \operatorname{pre}_{A}(e) \neq \varnothing$ iff $\gamma_{2}^{\prime}(q) \cap \operatorname{pre}_{A}(e) \neq \varnothing$. If all values are finite and all bounds are weak, then this intersection is nonempty iff both $b_{\ell(i)} \leqslant$ $\max \left(\operatorname{pre}_{A}(e)_{i}\right)$ and $b_{u(i)} \geqslant \min \left(\operatorname{pre}_{A}(e)_{i}\right)$. These were the lower-bound and upper-bound guards given in the construction of $B_{A}^{c}$ for compact $A$. Taking strictness and infinite bounds into account, we obtain the more complicated guards $\ell \operatorname{lguard}\left(i, \operatorname{pre}_{A}(e)_{i}\right)$ and $\operatorname{uguard}\left(i, \operatorname{pre}_{A}(e)_{i}\right)$, defined as follows. For an interval $I \subset \mathbb{R}$, and $i \in\{1, \ldots, n\}$, define
$\ell \operatorname{guard}(i, I)=\left\{\begin{array}{l}\text { true }, \\ b_{\ell(i)} \leqslant \sup (I), \\ b_{\ell(i)}<\sup (I),\end{array}\right.$
$\operatorname{uguard}(i, I)=\left\{\begin{array}{l}\text { true }, \\ b_{u(i)} \geqslant \inf (I), \\ b_{u(i)}>\inf (I),\end{array}\right.$
if $\mu_{\ell(i)}=\inf$,
if $\mu_{\ell(i)}=$ fin and $v_{\ell(i)}=$ strict $\uparrow(I)=w k$, otherwise;

These expressions are guarded commands with no assignments. To understand the definition, consider the conditions under which an interval $J$ intersects the interval $I$.

Fact 3.9. Let $I$ and $J$ be two nonempty intervals of the real line, and let $\varphi_{\ell}$ and $\varphi_{u}$ be defined as in Table I. Then $I \cap J \neq \varnothing$ iff $\varphi_{\ell}$ and $\varphi_{u}$ are both true.

The guard $\ell \operatorname{guard}(i, I)$ on the lower-bound variable $b_{\ell(i)}$ corresponds to the condition $\varphi_{\ell}$, and the guard uguard $(i, I)$ on the upper-bound variable $b_{u(i)}$ corresponds to $\varphi_{u}$. Notice that $\varphi_{\ell}$ is always true if $\inf (J)=-\infty$. Hence, the first line of the definition of $\ell \operatorname{guard}(i, I)$. If strict $\downarrow(J)=\operatorname{strict~} \uparrow(I)=w k$, then $\varphi_{\ell}$ is $\inf (J) \leqslant \sup (I)$. Hence, the second line of the definition of $\ell \operatorname{guard}(i, I)$. Finally, if either strict $\downarrow(J)=s t r$ or strict $\uparrow(I)=\operatorname{str}$, then $\varphi_{\ell}$ is $\inf (J)<\sup (I)$. Hence, the third line of the definition of $\operatorname{\ell guard}(i, I)$. Symmetrical remarks apply to $\operatorname{uguard}(i, I)$ and $\varphi_{u}$. Taking $I=\operatorname{pre}_{A}(e)_{i}$ and $J=\gamma_{2}^{\prime}(q)_{i}$, Lemma 3.10 follows.

Lemma 3.10. Let $e=(v, w)$ be an edge of the $n$-dimensional initialized rectangular automaton $A$, and let $\psi_{e}$ be the predicate $\bigwedge_{i=1}^{n}\left(\ell \operatorname{lguard}\left(i, \operatorname{pre}_{A}(e)_{i}\right) \wedge \operatorname{uguard}\left(i\right.\right.$, pre $\left.\left._{A}(e)_{i}\right)\right)$. For every upper-halfstate $q=((v, \boldsymbol{\mu}, \mathbf{v}, t p), \mathbf{x})$ of $B_{A}$, Post $_{A}^{e}(\gamma(q)) \neq \varnothing$ iff the vector $\mathbf{x} \in \mathbb{R}^{2 n+1}$ satisfies the predicate $\psi_{e}$.

The reader may recall from the compact case that the construction for coordinates $i \in j_{u m p}^{A}(e)$ differs from the

TABLE I
$I \cap J \neq \varnothing$ iff $\varphi_{\ell} \wedge \varphi_{u}$

| strict $\downarrow(J)$ | strict $\uparrow(I)$ | $\varphi_{\ell}$ |
| :---: | :---: | :---: |
| $w k$ | $w k$ | $\inf (J) \leqslant \sup (I)$ |
| $w k$ | str | $\inf (J)<\sup (I)$ |
| str | wk | $\inf (J)<\sup (I)$ |
| str | str |  |
|  | strict $\downarrow(I)$ | $\varphi_{u}$ |
| strict $\uparrow(J)$ | $w k$ | $\sup (J) \geqslant \inf (I)$ |
| wk | str | $\sup (J)>\inf (I)$ |
| wk | wh | $\sup (J)>\inf (I)$ |
| str | str |  |

construction for coordinates $i \notin j u m p_{A}(e)$. We first consider the general construction for the case $i \in j u m p_{A}(e)$. In this case, the lower-bound and upper-bound variables $b_{\ell(i)}$ and $b_{u(i)}$ are assigned to the infimum and supremum, respectively, of the interval $\operatorname{post}_{A}(e)_{i}$, and the finite/infinite and weak/strict bitvectors are updated appropriately. This is done by the lists $\ell \operatorname{assign}\left(i, \operatorname{post}_{A}(e)_{i}\right)$ and $\operatorname{uassign}\left(i\right.$, post $\left._{A}(e)_{i}\right)$ of assignments, defined as follows. For an interval $I \subset \mathbb{R}$, and $i \in\{1, \ldots, n\}$, define

## €assign(i, I)

$$
=\left\{\begin{aligned}
b_{\ell(i)} & :=\inf (I) ; \mu_{\ell(i)}:=\text { fin } ; v_{\ell(i)}:=\operatorname{strict} \downarrow\left(\text { post }_{A}(e)_{i}\right) \\
& \text { if } \inf (I) \neq-\infty, \\
b_{\ell(i)} & :=0 ; \mu_{\ell(i)}:=\inf ; v_{\ell(i)}:=\operatorname{str} \\
& \text { otherwise }
\end{aligned}\right.
$$

uassign(i, I)

$$
=\left\{\begin{aligned}
b_{u(i)} & :=\sup (I) ; \mu_{u(i)}:=\text { fin } ; v_{u(i)}:=\operatorname{strict} \uparrow\left(\operatorname{post}_{A}(e)_{i}\right) \\
& \text { if } \sup (I) \neq \infty, \\
b_{u(i)} & :=0 ; \mu_{u(i)}:=\inf ; v_{u(i)}:=\operatorname{str} \\
& \text { otherwise. }
\end{aligned}\right.
$$

These expressions are guarded commands with the guards true. The assignments to 0 are required for $B_{A}$ to be initialized. After such an assignment, the value of the variable is ignored due to the finite/infinite bit being inf. Now, for $i \in j u m p{ }_{A}(e)$, the extended CNF expression $\psi_{e, \mu, v}^{i}$ is defined by
$\ell \operatorname{lgard}\left(i, \operatorname{pre}_{A}(e)_{i}\right) \wedge \operatorname{uguard}\left(i, \operatorname{pre}_{A}(e)_{i}\right)$
$\wedge \ell \operatorname{assign}\left(i, \operatorname{post}_{A}(e)_{i}\right) \wedge \operatorname{uassign}\left(i, \operatorname{post}_{A}(e)_{i}\right)$.

From these definitions and Lemma 3.10, we obtain Lemma 3.11.

Lemma 3.11. Let $e=(v, w)$ be an edge of the initialized rectangular automaton $A$. For every upper-half state $q=$ $((v, \boldsymbol{\mu}, \mathbf{v}, t p), \mathbf{x})$ of $B_{A}$, and every coordinate $i \in j u m p_{A}(e)$, we have $\left[\text { Post }_{A}^{e}(\gamma(q))\right]_{i}^{w}=\left[\gamma\left(\text { Post }_{B_{A}}^{\psi_{e, \mu}, v, t p}(q)\right)\right]_{i}^{w}$.

The case $i \notin j u m p(e)$ is more complicated, because the lower-bound (resp. upper-bound) variable is reset only if its value is too small (resp. too large). Strictness considerations also contribute some complications. Our definitions follow once again from Fact 3.9. The necessary adjustments to $b_{\ell(i)}$ and $b_{u(i)}$, and to the finite/infinite and weak/strict bitvectors, are given by the extended CNF expressions $\ell$ adjust $\left(i, \operatorname{pre}_{A}(e)_{i}\right)$ and uadjust $\left(i\right.$, pre $\left._{A}(e)_{i}\right)$, defined as follows. For an interval $I \subset \mathbb{R}$, and $i \in\{1, \ldots, n\}$, define

## € adjust(i, I)

$$
\begin{aligned}
& \text { true, } \quad \text { if } \inf (I)=-\infty, \\
& \left(\mu_{\ell(i)}=\inf \rightarrow \mu_{\ell(i)}:=\text { fin; } b_{\ell(i)}:=\inf (I) ;\right. \\
& \left.v_{\ell(i)}:=\operatorname{strict} \downarrow(I)\right) \\
& \vee\left(\mu_{\ell(i)}=\text { fin } \wedge b_{\ell(i)}<\inf (I) \rightarrow b_{\ell(i)}:=\inf (I) ;\right. \\
& \left.v_{\ell(i)}:=\operatorname{strict} \downarrow(I)\right) \\
& \quad \vee\left(\mu_{\ell(i)}=\text { fin } \wedge b_{\ell(i)}=\inf (I) \wedge v_{\ell(i)}=w k \rightarrow\right. \\
& \left.v_{\ell(i)}:=\operatorname{strict} \downarrow(I)\right) \\
& \vee\left(\mu_{\ell(i)}=\text { fin } \wedge b_{\ell(i)}=\inf (I) \wedge v_{\ell(i)}=\operatorname{str}\right) \\
& \vee\left(\mu_{\ell(i)}=\operatorname{fin} \wedge b_{\ell(i)}>\inf (I)\right), \quad \text { otherwise; }
\end{aligned}
$$

uadjust $(i, I)$

$$
\begin{aligned}
& \text { true, } \quad \text { if } \sup (I)=\infty, \\
& \left(\mu_{u(i)}=\inf \rightarrow \mu_{u(i)}:=\operatorname{fin} ; b_{u(i)}:=\sup (I) ;\right. \\
& \left.v_{u(i)}:=\operatorname{strict} \uparrow(I)\right) \\
& \quad \vee\left(\mu_{u(i)}=\operatorname{fin} \wedge b_{u(i)}>\sup (I) \rightarrow b_{u(i)}:=\sup (I) ;\right. \\
& \left.v_{u(i)}:=\operatorname{strict} \uparrow(I)\right) \\
& \vee\left(\mu_{u(i)}=\operatorname{fin} \wedge b_{u(i)}=\sup (I) \wedge v_{u(i)}=w k \rightarrow\right. \\
& \left.v_{u(i)}:=\operatorname{strict} \uparrow(I)\right) \\
& \vee\left(\mu_{u(i)}=\operatorname{fin} \wedge b_{u(i)}=\sup (I) \wedge v_{u(i)}=\operatorname{str}\right) \\
& \vee\left(\mu_{u(i)}=\operatorname{fin} \wedge b_{u(i)}<\sup (I)\right), \quad \text { otherwise. }
\end{aligned}
$$

Then, for $i \notin j u m p_{A}(e)_{i}$, the extended CNF expression $\psi_{e, \mu, \nu}^{i}$ is defined by
$\quad \operatorname{lguard}\left(i, \operatorname{pre}_{A}(e)_{i}\right) \wedge \operatorname{uguard}\left(i, \operatorname{pre}_{A}(e)_{i}\right)$
$\quad \wedge \ell \operatorname{ladjust}\left(i, \operatorname{pre}_{A}(e)_{i}\right) \wedge \operatorname{uadjust}\left(i, \operatorname{pre}_{A}(e)_{i}\right)$.

Let us examine the definition of $\ell \operatorname{adjust}\left(i\right.$, pre $\left._{A}(e)_{i}\right)$. When the edge $e$ is traversed in $A$, then new information about the value of $a_{i}$ is obtained, namely, that it lies within the interval $\operatorname{pre}_{A}(e)_{i}$ (which, by assumption, is equal to post $\left.A_{A}(e)_{i}\right)$. Let $p=\inf \left(\operatorname{pre}_{A}(e)_{i}\right)$. If $p=-\infty$, then there is no new lowerbound information, and so $\ell$ ladjust $\left(i\right.$, pre $\left._{A}(e)_{i}\right)=t r u e$; hence, the first case of the definition. If $p \neq-\infty$, then we distinguish several cases. If $\mu_{\ell(i)}=\inf$, then the present lower bound is infinite, and so $\mu_{\ell(i)}$ must be set to fin, and $b_{\ell(i)}$ must be reassigned to $p$, and the weak/strict bit $v_{\ell(i)}$ must be assigned to the lower strictness of pre $A_{A}(e)_{i}$ (disjunct one of case two). Now suppose that $\mu_{\ell(i)}=$ fin. If $b_{\ell(i)}<p$, then again $b_{\ell(i)}$ and $v_{f(i)}$ must be reset to $p$ and its strictness (disjunct two). If $b_{\ell(i)}=p$ and $v_{\ell(i)}=w k$, then information is gained if the lower strictness of $\operatorname{pre}_{A}(e)_{i}$ is $s t r$. So in this case (disjunct three) we perform the assignment $v_{\ell(i)}:=$ strict $\downarrow\left(\right.$ pre $\left.{ }_{A}(e)_{i}\right)$. But if $b_{\ell(i)}=p$ and $v_{\ell(i)}=s t r$, then no information is gained, and so no assignment is performed (disjunct four). Finally, if $b_{\ell(i)}>p$, then there is no new lower-bound information, and so there is no assignment (disjunct five). The definition of $\operatorname{uadjust}\left(i\right.$, pre $\left._{A}(e)_{i}\right)$ is symmetric. Using Lemma 3.10, we conclude Lemma 3.12.

Lemma 3.12. Let $e=(v, w)$ be an edge of the initialized rectangular automaton $A$. For every upper-half state $q=$
$((v, \boldsymbol{\mu}, \boldsymbol{v}, t p), \mathbf{x})$ of $B_{A}$, and every coordinate $i \notin j u m p_{A}(e)$, we have $\left[\operatorname{Post}_{A}^{e}(\gamma(q))\right]_{i}^{w}=\left[\gamma\left(\operatorname{Post}_{B_{A}, \mu, v, t p}^{\Psi_{e}}(q)\right)\right]_{i}^{w}$.

Putting together Lemmas 3.11 and 3.12, we obtain the following.

Lemma 3.13. Let $A$ be an initialized rectangular automaton. For every upper-half state $q=((v, \boldsymbol{\mu}, \boldsymbol{v}, t p), \mathbf{x})$ of $B_{A}$, and every edge $e$ of $A$, $\operatorname{Post}_{A}^{e}(\gamma(q))=\gamma\left(\operatorname{Post}_{B_{A}, \mu, v, t p}^{\psi+}(q)\right)$. Moreover, the zone $\operatorname{Post}_{B_{A}}^{\psi_{e}, \mu, v, t p}(q)$ of $B_{A}$ is either empty or a singleton.

Proof. To see that $\left|\operatorname{Post}_{B_{A}}^{\psi_{e, \mu}, v, t p}(q)\right| \leqslant 1$, notice that all assignments made by the guarded commands comprising $\Psi_{e, \boldsymbol{\mu}, \mathrm{v}, t p}$ are deterministic, and the disjuncts of ใadjust $\left(i, \operatorname{pre}_{A}(e)_{i}\right)$ are mutually exclusive, as are the disjuncts of $\operatorname{uadjust}\left(i, \operatorname{pre}_{A}(e)_{i}\right)$. So from each state of $B_{A}$, at most one of the disjuncts of each of these CNF expressions can be executed.

Initial regions of $B_{A}$. For a rectangular region $R \subset \mathbb{R}^{n}$, we define the bitvectors $\boldsymbol{\mu}^{R}, \boldsymbol{v}^{R} \in\{0,1\}^{2 n}$ and the vector $\mathbf{x}^{R} \in \mathbb{R}^{2 n+1}$ as follows. If $\inf \left(R_{i}\right)=-\infty$, then $\mu_{\ell(i)}^{R}=\inf$, and $v_{\ell(i)}^{R}=s t r$, and $x_{\ell(i)}^{R}=0$. If $\inf \left(R_{i}\right) \neq-\infty$, then $\mu_{\ell(i)}^{R}=$ fin, and $v_{\ell(i)}^{R}=\operatorname{strict} \downarrow\left(R_{i}\right)$, and $x_{\ell(i)}^{R}=\inf \left(R_{i}\right)$. Similarly, if $\sup \left(R_{i}\right)=\infty$, then $\mu_{u(i)}^{R}=\inf$, and $v_{u(i)}^{R}=\operatorname{str}$, and $x_{u(i)}^{R}=0$. If $\sup \left(R_{i}\right) \neq \infty$, then $\mu_{u(i)}^{R}=$ fin, and $v_{u(i)}^{R}=\operatorname{strict} \uparrow\left(R_{i}\right)$, and $x_{u(i)}^{R}=\sup \left(R_{i}\right)$. Finally, let $x_{0}^{R}=0$. For a rectangular zone $Z$ of $A$, define $-\gamma(Z)$ to be the zone $\bigcup_{v \in V_{A}}\left\{\left(\left(v, \boldsymbol{\mu}^{[Z]^{v}}, \boldsymbol{v}^{[Z]^{v}}, 0\right)\right.\right.$, $\left.\left.\mathbf{x}^{[Z]^{v}}\right)\right\}$ of $B_{A}$. Notice that $-\gamma(Z) \subset U_{B_{A}}$ and that $-\gamma(Z)$ contains at most one state for each vertex of $B_{A}$.

We put Init $_{B_{A}}=-\gamma\left(\right.$ Init $\left._{A}\right)$. It is a straightforward matter to give the actual initial function of $B_{A}$ from this. Lemma 3.14 follows immediately from the definitions.

Lemma 3.14. Let a be an initialized rectangular automaton. For every rectangular zone $Z$ of $A$, we have $\gamma(-\gamma(Z))=Z$. In particular, Init $_{A}=\gamma\left(\right.$ Init $\left._{B_{A}}\right)$.

This completes the definition of the automaton $B_{A}$. Notice that $B_{A}$ is initialized and singular. The automaton $B_{A}$ has exponentially more vertices and edges than $A$. However, as in the translation from initialized stopwatch automata to timed automata, this exponential blowup does not adversely affect the space complexity of reachability analysis. Before we establish this, we first prove the analog of Lemma 3.13 for time steps.

Analysis of time steps. Lemma 3.13 relates the edge steps of $A$ to edge steps of $B_{A}$. We must develop a similar correspondence for time steps. For this purpose, the following two facts about the reachable states of $B_{A}$ are useful. Fact 3.15 says that every reachable state of $B_{A}$ that is the target of a time step has its finite/infinite and weak/strict bits set correctly. This is because reachability ensures that every
sequence of consecutive time steps must have been preceded by an $\varepsilon$ step.

Fact 3.15. Let $((v, \boldsymbol{\mu}, \boldsymbol{v}, t p), \mathbf{x})$ be a reachable state of $B_{A}$ with $t p=1$ and $x_{0}>0$, and let $i \in\{1, \ldots, n\}$. First, if $\inf \left(\right.$ flow $\left._{A}(v)_{i}\right)=-\infty$, then $\mu_{\ell(i)}=i n f$, and if $\sup \left(\right.$ flow $\left._{A}(v)_{i}\right)$ $=\infty$, then $\mu_{u(i)}=\inf$. Second, if strict $\downarrow\left(\right.$ flow $\left._{A}(v)_{i}\right)=s t r$, then $v_{\ell(i)}=s t r$, and if strict $\uparrow\left(\right.$ flow $\left._{A}(v)_{i}\right)=s t r$, then $v_{u(i)}=s t r$.

We say that a state $q$ of $B_{A}$ satisfies the invariant of $A$ if $\gamma(q) \subset Q_{A}$. Fact 3.16 says that every reachable state of $B_{A}$ that does not satisfy the invariant of $A$ cannot move, by $\varepsilon$ and time steps alone, to a state that does satisfy the invariant. This is because all initial states of $B_{A}$ and all target states of inherited edge steps satisfy the invariant of $A$, and because the direction of a flow of $B_{A}$ can change only after an inherited edge step. The latter follows from $B_{A}$ being initialized. The former follows from the assumption that the initial region of each vertex of $A$ is contained in the invariant region, and postguard region of each edge is contained in the invariant region of the target vertex.

Fact 3.16. Let $((v, \boldsymbol{\mu}, \boldsymbol{v}, t p), \mathbf{x})$ be a reachable state of $B_{A}$, and let $i \in\{1, \ldots, n\}$. If $x_{\ell(i)} \notin i n v_{A}(v)_{i}$, then $\inf \left(\right.$ flow $\left._{A}(v)_{i}\right)<0$, and if $x_{u(i)} \notin i n v_{A}(v)_{i}$, then $\sup \left(\right.$ flow $\left._{A}(v)_{i}\right)>0$.

Lemma 3.17 relates the time steps of $A$ to time steps of $B_{A}$.

Lemma 3.17. Let $A$ be an initialized rectangular automaton. For every reachable upper-half state $q$ of $B_{A}$, and every duration $t \in \mathbb{R}_{\geqslant 0}$, Post $_{A}^{t}(\gamma(q))=\gamma\left(\right.$ Post $\left._{B_{A}}^{t}(q)\right)$. Moreover, the zone Post $_{B_{A}}^{t}(q) \cap U_{B_{A}}$ of $B_{A}$ is either empty or a singleton.

Proof. Consider a reachable state $q=((v, \boldsymbol{\mu}, \boldsymbol{v}, t p), \mathbf{x})$ of $B_{A}$, not necessarily upper-half. From the construction of $B_{A}$ we observe the following. If $t p=0$, then there is a unique state $r$ such that $q \xrightarrow{\varepsilon} B_{A} r$; furthermore, $\gamma(r)=\varnothing$, and the time-passage bit of $r$ is 1 . If $t p=1$, then there is no state $r$ such that $q \xrightarrow{\varepsilon} B_{B_{A}} r$. If $t p=0$, then there is no state $r$ such that $q \xrightarrow{[t]}{ }_{B_{A}} r$, for any $t>0$. If $t p=1$, since $B_{A}$ is a singular automaton, for each $t>0$, there is a at most one state $r$ such that $q \xrightarrow{[t]}_{B_{A}} r$; furthermore, the time-passage bit of $r$ is still 1 . This shows the second claim of the lemma, that $\left|\operatorname{Post}_{B_{A}}^{t}(q) \cap U_{B_{A}}\right| \leqslant 1$. Moreover, since $A$ has no $\varepsilon$ moves, the first claim of the lemma is reduced to the following proof obligation: for every reachable upper-half state $q$ of $B_{A}$, and every positive duration $t>0$, show that $\operatorname{Post}_{A}^{[t]}(\gamma(q))=$ $\gamma\left(\operatorname{Post}_{B_{A}}^{\pi}(q)\right)$, where $\pi=\varepsilon \cdot[t]$ if the time-passage bit of $q$ is 0 , and $\pi=[t]$ if the time passage bit of $q$ is 1 .

So suppose that $q=((v, \boldsymbol{\mu}, \boldsymbol{v}, t p), \mathbf{x})$ is a reachable upperhalf state of $B_{A}$, and consider a duration $t>0$. Then $\gamma(q)$ has the form $\{v\} \times R$, for a rectangular region $R \subset \mathbb{R}^{n}$. Recall from the proof of Proposition 2.2 that $\operatorname{Post}_{A}^{[t]}(\gamma(q))$ has the form $\{v\} \times S$, where
$\inf \left(S_{i}\right)=\max \left\{\inf \left(\operatorname{inv}_{A}(v)_{i}\right), \inf \left(R_{i}\right)+t \cdot \inf _{\left.\left(\text {flow }_{A}(v)_{i}\right)\right\},}\right.$
$\sup \left(S_{i}\right)=\min \left\{\sup \left(i n v_{A}(v)_{i}\right), \sup \left(R_{i}\right)+t \cdot \sup \left(\right.\right.$ flow $\left.\left._{A}(v)_{i}\right)\right\}$.

The strictness of the infimum of $S_{i}$ is given as follows. Put $\operatorname{Inv}=\operatorname{inv}_{A}(v)_{i}, \quad$ Flow $=$ flow $_{A}(v)_{i}$, and $\operatorname{Try}=\inf \left(R_{i}\right)+t$. $\inf ($ Flow $)$. If $\inf ($ Inv $)>$ Try, then strict $\downarrow\left(S_{i}\right)=\operatorname{strict} \downarrow($ Inv $)$. If $\inf (\operatorname{Inv})=\operatorname{Try}$, then $\operatorname{strict} \downarrow\left(S_{i}\right)=w k$ iff $\operatorname{strict} \downarrow($ Inv $)=$ strict $\downarrow\left(R_{i}\right)=\operatorname{strict} \downarrow($ Flow $)=w k$. If $\inf ($ Inv $)<$ Try, then strict $\downarrow\left(S_{i}\right)=w k$ iff strict $\downarrow\left(R_{i}\right)=$ strict $\downarrow($ Flow $)=w k$. The strictness of the supremum of $S_{i}$ is given symmetrically. We show that $\gamma\left(\operatorname{Post}_{B_{A}}^{\pi}(q)\right)=\{v\} \times S$, for $\pi=\varepsilon \cdot[t]$ if $t p=0$, and $\pi=[t]$ if $t p=1$.

Case Post $_{B_{A}}^{\pi}(q) \neq \varnothing$. In this case, Post $_{B_{A}}^{\pi}(q)$ has the form $\{r\}$ for a reachable upper-half state $r=\left(\left(v, \boldsymbol{\mu}^{\prime}, \boldsymbol{v}^{\prime}, 1\right), \mathbf{y}\right)$ of $B_{A}$ with $y_{0}>0$. Hence, Fact 3.15 applies to $r$. The set $\gamma(r)$ has the form $\{v\} \times T$, for a rectangular region $T \subset \mathbb{R}^{n}$. We show that $T=S$. For this purpose, we show that for each coordinate $i$, the right endpoints of the intervals $T_{i}$ and $S_{i}$ coincide in value and in strictness. The argument for the left endpoints is symmetric. In the following, put $I s u p=\sup (\operatorname{Inv}), I \uparrow=$ strict $\uparrow($ Inv $), F s u p=\sup ($ Flow $)$, and $F \uparrow=\operatorname{strict~} \uparrow($ Flow $)$.

Subcase $\mu_{u(i)}^{\prime}=$ fin. By Fact 3.15, the flow interval flow $(v)_{i}$ is bounded from above. So the upper-bound variable $b_{u(i)}$ moves at the supremum of the allowable slopes for $a_{i}$. Hence, $y_{u(i)}=x_{u(i)}+t \cdot$ Fsup and $\sup \left(T_{i}\right)=$ $\min \left\{y_{u(i)}\right.$, Isup $\}=\min \left\{x_{u(i)}+t \cdot\right.$ Fsup, Isup $\}$. From the subcase assumption and $q \xrightarrow{\pi} B_{A} r$, we know that $\mu_{u(i)}=$ fin, and therefore $\sup \left(R_{i}\right)=\min \left\{x_{u(i)}, I s u p\right\}$. By Fact 3.16, if $x_{u(i)}>I$ sup, then Fsup $>0$. Thus Eq. (2) implies $\sup \left(S_{i}\right)=$ $\min \left\{\operatorname{Isup}, x_{u(i)}+t \cdot F s u p\right\}$, which is the same as $\sup \left(T_{i}\right)$. The question of strictness remains. If $y_{u(i)}>$ Isup, then $\operatorname{strict} \uparrow\left(T_{i}\right)$ $=I \uparrow=\operatorname{strict} \uparrow\left(S_{i}\right)$. If $y_{u(i)} \leqslant I s u p$ and $v_{u(i)}^{\prime}=\operatorname{str}$, then from $q \xrightarrow{\pi} B_{B_{A}} r$ we know that either $F \uparrow=s t r$ or $v_{u(i)}=s t r$, and in both cases $\operatorname{strict} \uparrow\left(T_{i}\right)=\operatorname{str}=\operatorname{strict} \uparrow\left(S_{i}\right)$. If $y_{u(i)} \leqslant I s u p$ and $v_{u(i)}^{\prime}=w k$, then from $q \xrightarrow{\pi} \vec{B}_{A} r$ we know that $v_{u(i)}=w k$, and Fact 3.15 implies $F \uparrow=w k$. Therefore, strict $\uparrow\left(T_{i}\right)=w k$ iff either $y_{u(i)}<I s u p$ or $I \uparrow=w k$, and the same is true for strict $\uparrow\left(S_{i}\right)$.

Subcase $\mu_{u(i)}^{\prime}=\inf . \quad$ Then $\sup \left(T_{i}\right)=\operatorname{Isup}$ and $\operatorname{strict} \uparrow\left(T_{i}\right)$ $=I \uparrow$. From the subcase assumption and $q \xrightarrow[B_{B_{A}}]{ } r$, we know that either $\operatorname{Fsup}=\infty$, or $\mu_{u(i)}=\inf$ and therefore $\sup \left(R_{i}\right)=$ Isup and $\operatorname{strict} \uparrow\left(R_{i}\right)=I \uparrow$. Either way, Eq. (2) implies $\sup \left(S_{i}\right)=I$ sup and $\operatorname{strict} \uparrow\left(S_{i}\right)=I \uparrow$.

Case $\operatorname{Post}_{B_{A}}^{\pi}(q)=\varnothing$. This means that for some coordinate $i$, the lower-bound variable $b_{\ell(i)}$ rises above the upper boundary of Inv within time $t$, or the upper-bound
variable $b_{u(i)}$ drops below the lower boundary of Inv. In the first case, either $x_{\ell(i)}+t \cdot \inf ($ Flow $)>\sup ($ Inv $)$, or these two expressions are equal and one of $v_{\ell(i)}$ or strict $\downarrow$ (Flow) or strict $\uparrow(\operatorname{Inv})$ is $s t r$. The second case is symmetric and will not be detailed. If $x_{t(i)}+t \cdot \inf ($ Flow $)>\sup ($ Inv $)$, then Eqs. (1) and (2) together imply $\inf \left(S_{i}\right)>\sup \left(S_{i}\right)$. If $x_{\ell(i)}+t$. $\inf ($ Flow $)=\sup ($ Inv $)$ and one of $v_{\ell(i)}$ or strict $\downarrow$ (Flow) or strict $\uparrow(\operatorname{Inv})$ is str, then $\inf \left(S_{i}\right) \geqslant \sup \left(S_{i}\right)$ and either $\operatorname{strict} \uparrow\left(S_{i}\right)=\operatorname{str}$ or strict $\downarrow\left(S_{i}\right)=\operatorname{str}$. In all cases, $S_{i}=\varnothing$ and therefore $\{v\} \times S=\varnothing$. But from the assumption $\operatorname{Post}_{B_{A}}^{\pi}(q)$ $=\varnothing$ it follows that also $\gamma\left(\operatorname{Post}_{B_{A}}^{\pi}(q)\right)=\varnothing$, which concludes the proof.

From Lemmas 3.13 and 3.17 it follows that for every upper-half zone $Z \subset U_{B_{A}}$, and every label $\tau \in \Sigma \cup \mathbb{R}_{\geqslant 0}$, $\operatorname{Post}_{A}^{\tau}(\gamma(Z))=\gamma\left(\operatorname{Post}_{B_{A}}^{\tau}(Z)\right)$ and, by Proposition 2.1, $\operatorname{Pre}_{A}^{\tau}(\gamma(Z))=\gamma\left(\operatorname{Pre}_{-B_{-A}}^{\tau}(Z)\right)$. Hence we again have the commuting diagram of Fig. 8, with all superscripts $c$ and the restriction that $Z$ be compact removed. From Lemma 3.14 and the fact that Post commutes with $\gamma$, we conclude, as in the compact case, that $\operatorname{Reach}(A)=\gamma\left(\operatorname{Reach}\left(B_{A}\right)\right)$, that $\operatorname{Lang}(A) \subset \operatorname{Lang}\left(B_{A}\right)$, and that $A$ and $B_{A}$ accept the same finite timed words.

Theorem 3.18. The reachability problem for initialized rectangular automata is complete for PSPACE.

Proof. For containment in PSPACE, notice that $\left|V_{B_{A}}\right|=\left|V_{A}\right| \cdot 2^{4 n+1}$, and that the definition of $B_{A}$ uses the same constants as the definition of $A$. The space requirement $O\left(\log \left|V_{B_{A}}\right|+(2 n+1) \cdot\left(\log (2 n+1)+\log L_{B_{A}}\right)\right)$ $=O\left(\log \left|V_{A}\right|+n \cdot\left(\log n+\log L_{A}\right)\right)$, where $L_{A}$ is defined as for singular automata (see proof of Theorem 3.5), is polynomial in the size of $A$.

## 3.3. $\omega$-Language Emptiness

While the initialized rectangular automaton $A$ and the initialized singular automaton $B_{A}$ accept the same finite timed words, the automaton $B_{A}$ may accept infinite timed words that are not accepted by $A$. To see this, consider the 1D initialized rectangular automaton $\tilde{A}$ from Fig. 9. The only coordinate of $\tilde{A}$ is a clock, the only vertex of $\tilde{A}$ is $v$, and the invariant region of $v$ is $\mathbb{R}$. Note that $\tilde{A}$ is not a timed automaton, because the initial region $\operatorname{init}_{\tilde{A}}(v)=(-\infty, 0]$ is
unbounded from below. While every finite timed word of the form $\left(1 \sigma_{1}\right)^{m}$ is accepted by $\tilde{A}$, the divergent timed word $\left(1 \sigma_{1}\right)^{\omega}$ is not accepted by $\tilde{A}$. This is because, no matter what the initial value of the clock $c$, in a divergent run it will eventually be positive. However, in $B_{\tilde{A}}$ the finite/infinite bit for the lower bound on $c$ is initially inf, and remains inf during time steps and $\sigma_{1}$ steps. Therefore $\left(1 \sigma_{1}\right)^{\omega}$ is accepted by $B_{\tilde{A}}$. A similar phenomenon is exhibited with unbounded postguard intervals. The definition of bounded nondeterminism (Section 2) precludes both.

## Closure under Divergent Limits

A set $\mathscr{L}$ of infinite timed words is limit-closed if for all infinite timed words $\bar{\tau}$, if every finite prefix of $\bar{\tau}$ is a prefix of some word in $\mathscr{L}$, then $\bar{\tau}$ itself is in $\mathscr{L}$. Since we are interested only in infinite timed words that diverge, we relax the requirement of limit closure as follows [HNSY94]. The set $\mathscr{L}$ is closed under divergent limits if for all divergent timed words $\bar{\tau}$, if every finite prefix of $\bar{\tau}$ is a prefix of some word in $\mathscr{L}$, then $\bar{\tau}$ itself is in $\mathscr{L}$. A set of divergent timed words that is closed under divergent limits is completely determined by its finite prefixes. So if we have two rectangular automata $A_{1}$ and $A_{2}$ such that (1) every finite timed word accepted by $A_{1}$ is accepted also by $A_{2}$, and (2) the $\omega$-language $\operatorname{Lang}\left(A_{2}\right)$ is closed under divergent limits, then $\operatorname{Lang}\left(A_{1}\right) \subset \operatorname{Lang}\left(A_{2}\right)$. Specifically, if $A$ is an initialized rectangular automaton whose $\omega$-language $\operatorname{Lang}(A)$ is closed under divergent limits, then $\operatorname{Lang}(A)=\operatorname{Lang}\left(B_{A}\right)$.

As we have seen, the $\omega$-language of the initialized rectangular automaton $\tilde{A}$ is not closed under divergent limits. The unbounded initial region of $\tilde{A}$ is to blame. We will prove that for every initialized rectangular automaton $A$ with bounded nondeterminism, the $\omega$-language $\operatorname{Lang}(A)$ is closed under divergent limits. Consequently, for initialized rectangular automata $A$ with bounded nondeterminism, the translation from $A$ to $B_{A}$ can be used to solve not only reachability but also $\omega$-language emptiness.

For a rectangular automaton $A$, the $\omega$-language $\operatorname{Lang}^{c}(A)$ with convergent words is the set of all infinite timed words, divergent or not, that are accepted by runs of $A$. One way of showing that $\operatorname{Lang}(A)$ is closed under divergent limits is to prove that $\operatorname{Lang}^{c}(A)$ is limit-closed. While it is not true for all initialized rectangular automata $A$ with bounded nondeterminism that $\operatorname{Lang}^{c}(A)$ is limit-closed, this is true in


FIG. 8. Commuting diagram for compact rectangular zones $Z$ of a compact initialized rectangular automaton $A$.


FIG. 9. The initialized rectangular automaton $\tilde{A}$ with $\operatorname{Lang}(\tilde{A}) \not \supset$ $\operatorname{Lang}\left(B_{\tilde{A}}\right)$.
the special case that $A$ has compact nondeterminism. So we first consider the special case of compact nondeterminism, and then proceed to the more general case of bounded nondeterminism.

Preliminary definitions. Let $A$ be an $n$-dimensional rectangular automaton (without $\varepsilon$ moves). In this section, it is convenient to consider timed words over the alphabet $E \cup \mathbb{R}_{\geqslant 0}$, where the edge set replaces the observation alphabet. Formally, a timed edge word for $A$ is a finite or infinite sequence $\bar{\pi}=\pi_{0} \pi_{1} \pi_{2} \cdots$ of letters from $E \cup \mathbb{R}_{\geqslant 0}$. An edge run of $A$ that accepts $\bar{\pi}$ is a sequence of the form $q_{0} \xrightarrow{\vec{\pi}_{0}} q_{1} \xrightarrow{\pi_{1}} q_{2} \xrightarrow{\pi_{2}} \cdots$ with $q_{0} \in$ Init, and $q_{i} \in Q$ for all $i \geqslant 0$. Divergence for timed edge words and edge runs is defined as for timed words and runs. The edge $\omega$-language of $A$ (with convergent words), denoted $\operatorname{Lang}_{E}(A)$ (resp. $\operatorname{Lang}_{E}^{c}(A)$ ), is the set of all divergent (resp. all infinite) timed edge words that are accepted by runs of $A$. Limit closure and closure under divergent limits for edge $\omega$-languages is defined as for $\omega$-languages. The following observation is due to the fact that the edge set of $A$ is finite.

Proposition 3.19. Let $A$ be a rectangular automaton. First, if the edge $\omega$-language $\operatorname{Lang}_{E}(A)$ is closed under divergent limits, then so is the $\omega$-language Lang $(A)$. Second, if the edge $\omega$-language $\operatorname{Lang}_{E}^{c}(A)$ with convergent words is limit-closed, then so is $\operatorname{Lang}^{c}(A)$.

Proof. We say that a timed edge word $\pi_{0} \pi_{1} \pi_{2} \ldots$ matches the timed word $\tau_{0} \tau_{1} \tau_{2} \cdots$ if for all $i \geqslant 0$, if $\pi_{i} \in E$ then $\tau_{i}=o b s\left(\pi_{i}\right)$, and if $\pi_{i} \in \mathbb{R}_{\geqslant 0}$ then $\tau_{i}=\pi_{i}$. Assume that $\operatorname{Lang}_{E}^{c}(A)$ is limit-closed, and consider all finite prefixes of an infinite timed word $\bar{\tau} \in \operatorname{Lang}^{c}(A)$. The finite timed edge words that match these prefixes and are accepted by $A$ form the nodes of a finitely branching tree: the root of the tree is the empty timed edge word, and the successors of a node $w$ are the (matching and accepted) timed edge words of the form $w \cdot \pi$, for $\pi \in E \cup \mathbb{R}_{\geqslant 0}$. By König's lemma, the tree has an infinite rooted path, and the nodes along this path are the finite prefixes of an infinite timed edge word $\bar{\pi}$. Since $\operatorname{Lang}_{E}^{c}(A)$ is limit-closed, $\bar{\pi} \in \operatorname{Lang}_{E}^{c}(A)$. Since $\bar{\pi}$ matches $\bar{\tau}$, we conclude that $\bar{\tau} \in \operatorname{Lang}^{c}(A)$. This proves the second claim of the lemma. The first claim follows from the fact that if $\bar{\tau}$ is divergent, then $\bar{\pi}$ is also divergent.

Notice that for every singular automaton $A$, the edge $\omega$-language $\operatorname{Lang}_{E}^{c}(A)$ with convergent words (and therefore also $\left.\operatorname{Lang}^{c}(A)\right)$ is limit-closed. This is because a singular automaton $A$ has a finite number of initial states, and all edge steps as well as time steps are deterministic; that is, for all states $q$ of $A$, and every label $\pi \in E \cup \mathbb{R}_{\geqslant 0}$, the zone $\operatorname{Post}^{\pi}(q)$ is either empty or a singleton. Next, we consider a class of nonsingular, and therefore nondeterministic, automata whose $\omega$-languages with convergent words are limit-closed.

## The Case of Compact Nondeterminism

The rectangular automaton $A$ has compact nondeterminism if it has bounded nondeterminism, and all rectangular regions that appear in the definition of $A$ are closed. More precisely, $A$ has compact nondeterminism if (1) for every vertex $v$ of $A$, the regions $\operatorname{init}(v)$ and $\operatorname{flow}(v)$ are compact, and $\operatorname{inv}(v)$ is closed, and (2) for every edge $e$ of $A$, the regions $\operatorname{pre}(e)$ and post (e) are closed, and for every coordinate $i \in\{1, \ldots, n\}$ with $i \in j u m p(e)$, the interval post $(e)_{i}$ is bounded. We show that if $A$ has compact nondeterminism, then the $\omega$-language Lang $^{c}(A)$ with convergent words is limit-closed. This result is a consequence of the following basic property of compact zones, which is inherited from $\mathbb{R}^{n}$.

Proposition 3.20. Let $A$ be a rectangular automaton, and let $\left(Z_{i}\right)_{i \in \mathbb{N}}$ be an infinite decreasing sequence of nonempty compact zones of $A$; that is, $Z_{i} \supset Z_{i+1}$ for all $i \geqslant 0$. 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 the vertex set of $A$ is finite.

The next fact points out the compactness of all zones that will appear in the proof of the main theorem. It follows immediately from the proof of Proposition 2.2.

Fact 3.21. Let $A$ be a rectangular automaton with compact nondeterminism, and let $Z$ and $Z^{\prime}$ be two compact rectangular zones of $A$. For every label $\pi \in E \cup \mathbb{R}_{\geqslant 0}$, the zones $\operatorname{Post}^{\pi}(Z)$ and $\operatorname{Pre}^{\pi}\left(Z^{\prime}\right) \cap Z$ are compact and rectangular.

Note the asymmetry in Fact 3.21; the intersection of $\operatorname{Pre}{ }^{\pi}\left(Z^{\prime}\right)$ with the compact zone $Z$ is required for compactness, because the preguard regions of automata with compact nondeterminism are not necessarily bounded. The next lemma gives the heart of the limit-closure argument, showing that if all finite prefixes of an infinite timed edge word can be generated from a given zone $Z$, then there is a single state in $Z$ from which each prefix can be generated.

Lemma 3.22. Let $A$ be a rectangular automaton with compact nondeterminism, and let $Z$ be a compact rectangular zone of $A$. Suppose that $\bar{\pi} \in(E \cup \mathbb{R} \geqslant 0)^{\omega}$ is a timed edge word
such that for every $m \in \mathbb{N}$, Post $^{\pi_{0} \pi_{1} \cdots \pi_{m}}(Z) \neq \varnothing$. Then there is a state $q \in Z$ such that for every $m \in \mathbb{N}$, Post $^{\pi_{0} \pi_{1} \cdots \pi_{m}}(q) \neq \varnothing$.

Proof. For each $m \in \mathbb{N}$, define $Z_{m}=\{q \in Z \mid$ Post $\left.^{\pi_{0} \pi_{1} \cdots \pi_{m}}(q) \neq \varnothing\right\}$. Since each Post $\pi_{0} \pi_{1} \cdots \pi_{m}(Z)$ is nonempty, each $Z_{m}$ is nonempty. Also, $Z_{m} \supset Z_{m+1}$ for all $m \geqslant 0$. We show that each zone $Z_{m}$ is compact; then the lemma follows from Proposition 3.20. By Fact 3.21 for each $m \in \mathbb{N}$, the zone Post ${ }^{\pi_{0} \pi_{1} \cdots \pi_{m}}(Z)$ is compact and rectangular. Therefore, again by Fact 3.21 , the zone $Z_{m}=Z \cap \operatorname{Pre} e^{\pi_{0} \pi_{1} \cdots \pi_{m}}$ (Post ${ }^{\pi_{0} \pi_{1} \cdots \pi_{m}}(Z)$ ) is compact as well.

Now we are ready to prove the main theorem.
Theorem 3.23. If $A$ is a rectangular automaton with compact nondeterminism, then the $\omega$-language Lang ${ }^{c}(A)$ with convergent words is limit-closed.

Proof. Let $A$ be a rectangular automaton with compact nondeterminism. By Proposition 3.19, it suffices to show that the edge $\omega$-language $\operatorname{Lang}_{E}^{c}(A)$ with convergent words is limit-closed. Suppose that $\bar{\pi} \in\left(E \cup \mathbb{R}_{\geqslant 0}\right)^{\omega}$ is an infinite timed edge word such that for every $m \in \mathbb{N}$, the prefix $\pi_{0} \pi_{1} \cdots \pi_{m}$ is a finite prefix of a timed edge word in $\operatorname{Lang}_{E}^{c}(A)$; that is, for each $m \geqslant 0$, Post $^{\pi_{0} \pi_{1} \cdots \pi_{m}}$ ( Init $) \neq \varnothing$. We need to show that $\bar{\pi} \in \operatorname{Lang}_{E}^{c}(A)$.

We define an infinite sequence of zones $\left(Z_{i}\right)_{i \in \mathbb{N}}$ of $A$. Let $Z_{0}=$ Init. Since $Z_{0}$ is compact and rectangular, by Lemma 3.22, there is a state $q_{0} \in Z_{0}$ such that for each $m \geqslant 0$, Post $^{\pi_{0} \pi_{1} \cdots \pi_{m}}\left(q_{0}\right) \neq \varnothing$. Let $Z_{1}=\operatorname{Post}^{\pi_{0}}\left(q_{0}\right)$. Then $Z_{1}$ is compact and rectangular, and for each $m \geqslant 1$, Post $\pi_{1} \pi_{2} \cdots \pi_{m}\left(Z_{1}\right) \neq \varnothing$. So by Lemma 3.22, there is a state $q_{1} \in Z_{1}$ such that for each $m \geqslant 1$, Post $^{\pi_{1} \pi_{2} \cdots \pi_{m}}\left(q_{1}\right) \neq \varnothing$. Proceed inductively in this manner, with $Z_{i+1}=\operatorname{Post}^{\pi_{i}}\left(q_{i}\right)$ compact and rectangular, and $q_{i+1} \in Z_{i+1}$ given by Lemma 3.22 such that for each $m \geqslant i+1$, Post $\pi^{\pi_{i+1} \pi_{i+2} \cdots \pi_{m}}\left(q_{i+1}\right) \neq \varnothing$. Then $q_{0} \xrightarrow{\pi_{0}} q_{1} \xrightarrow{\pi_{1}} q_{2} \xrightarrow{\pi_{2}} \cdots$ is an edge run of $A$ that accepts $\bar{\pi}$.

Corollary 3.24. The $\omega$-language emptiness problem for initialized rectangular automata with compact nondeterminism is complete for PSPACE.

## The Case of Bounded Nondeterminism

For rectangular automata $A$ with bounded, noncompact nondeterminism, the $\omega$-language $\operatorname{Lang}^{c}(A)$ with convergent words may not be limit-closed. To see this, consider the 1D initialized rectangular automaton $\bar{A}$ from Fig. 10 (with the trivial invariant function $\lambda v . \mathbb{R}$ ). While every finite prefix of the infinite timed word $\bar{\tau}=\frac{1}{2} \sigma_{1} \frac{1}{4} \sigma_{1} \frac{1}{8} \sigma_{1} \cdots$ is accepted by $\bar{A}$, the convergent word $\bar{\tau}$ is not in $\operatorname{Lang}^{c}(\bar{A})$. However, we show that the $\omega$-language $\operatorname{Lang}(A)$ is still closed under divergent limits for all rectangular automata $A$ that have bounded nondeterminism and are initialized, like the sample automaton $\bar{A}$. Bounded regions have no analogue to Proposition 3.20, and this greatly complicates the proof, which now relies on a detailed case analysis of the flow


FIG. 10. The initialized rectangular automaton $\bar{A}$ for which $\operatorname{Lang}^{c}(\bar{A})$ is not limit-closed.
function. The following fact points out the boundedness of all zones that will appear in the proof of the main theorem.

Fact 3.25. Let $A$ be a rectangular automaton with bounded nondeterminism, and let $Z$ be a bounded rectangular zone of $A$. For every label $\pi \in E \cup \mathbb{R}_{\geqslant 0}$, the zone $\operatorname{Post}^{\pi}(Z)$ is bounded and rectangular.

The next lemma gives the heart of the argument. It shows that for 1D rectangular automata $A$ with constant, bounded flow functions, $\operatorname{Lang}(A)$ is closed under those divergent limits that are accepted by runs without discontinuous jumps in the continuous state.

Lemma 3.26. Let $\mathscr{I}$ be a finite set of intervals, and let flow be a bounded interval. Let $\left(t_{i}\right)_{i \in \mathbb{N}}$ be an infinite sequence of positive reals with $\sum_{i=0}^{\infty} t_{i}=\infty$, and let $\left(I_{i}\right)_{i \in \mathbb{N}}$ be an infinite sequence of intervals such that $I_{0}$ is bounded and each $I_{i}$ is the intersection of one or more members of $\mathscr{I}$. Suppose that for each $m \in \mathbb{N}$, there is a finite sequence $x_{0}, x_{1}, \ldots, x_{m}$ of reals such that for all $i \in\{0, \ldots, m\}$, we have $x_{i} \in I_{i}$, and for all $i \in\{0, \ldots, m-1\}$, we have $\left(x_{i+1}-x_{i}\right) / t_{i} \in$ flow. Then there is an infinite sequence $\left(x_{i}\right)_{i \in \mathbb{N}}$ of reals such that for all $i \geqslant 0$, we have $x_{i} \in I_{i}$ and $\left(x_{i+1}-x_{i}\right) / t_{i} \in$ flow.

Proof. We call a finite sequence $x_{0}, x_{1}, \ldots, x_{m} m$-admissible if for all $i \in\{0, \ldots, m\}$, we have $x_{i} \in I_{i}$, and for all $i \in$ $\{0, \ldots, m-1\}$, we have $\left(x_{i+1}-x_{i}\right) / t_{i} \in$ flow. We call an infinite sequence $\left(x_{i}\right)_{i \in \mathbb{N}}$ admissible if for all $i \geqslant 0$, we have $x_{i} \in I_{i}$ and $\left(x_{i+1}-x_{i}\right) / t_{i} \in$ flow. Think of these sequences as values of a variable $a$, with $\dot{a} \in f$ flow, in a run with time steps of durations $t_{i}$ and without discontinuous jumps. Let $\mathscr{F}$ be the set $\left\{J \subset \mathbb{R} \mid J=I_{i}\right.$ for infinitely many $\left.i\right\}$ of intervals.

Case $0 \notin$ closure (flow). Suppose that flow $\subset(\delta, \infty)$ for some $\delta>0$. The case of flow $\subset(-\infty, \delta)$ is handled symmetrically. Let $k \in \mathbb{R}$ be larger than all of the finite endpoints of the intervals in $\mathscr{I}$. The point here is that the slope of $a$ is bounded from below by $\delta$, so that once $\left(k-\inf \left(I_{0}\right)\right) / \delta$ time units have passed, no matter what the initial value of $a$, the value of $a$ will be greater than all of the finite endpoints of intervals from $\mathscr{I}$ (note that $\inf \left(I_{0}\right)$ is finite, because $I_{0}$ is bounded). Let $j$ be large enough so that $\sum_{i=0}^{j-1} t_{i}>$ $\left(k-\inf \left(I_{0}\right)\right) / \delta$; such a $j$ exists because the sum $\sum_{i=0}^{\infty} t_{i}$
diverges. Then, for every $m$-admissible sequence $x_{0}, x_{1}, \ldots$, $x_{m}$, we have $x_{i}>k$ for all $i \in\{j+1, \ldots, m\}$. Since each $I_{i}$ is an intersection of intervals from $\mathscr{I}$, it follows that for every $i>j$, we have $I_{i} \supset(k, \infty)$. Consider a $j$-admissible sequence $x_{0}, x_{1}, \ldots, x_{j}$ and choose $\delta^{\prime} \in$ flow. Then $x_{0}, x_{1}, \ldots, x_{j}, x_{j}+$ $\delta^{\prime} \cdot t_{j}, x_{j}+\delta^{\prime} \cdot\left(t_{j}+t_{j+1}\right) \cdots$, is an admissible sequence.

Case $0=\inf ($ flow $)$ or $0=\sup ($ flow $)$. We suppose that $0=\inf ($ flow $)$; the case $0=\sup ($ flow $)$ is handled symmetrically. Among the intervals $I_{i}$ are only finitely many distinct intervals. Therefore, $\cap \mathscr{J} \neq \varnothing$, because flow $\cap(-\infty, 0)=\varnothing$, so $a$ can never descend from an interval $I_{i}$ to an interval $I_{j}$ all of whose elements are less than those of $I_{i}$. Let $j_{1}$ be large enough so that (1) for every $i \in \mathbb{N}$, we have $I_{j_{1}+i} \in \mathscr{J}$, and (2) for every $J \in \mathscr{J}$, there is an $i<j_{1}$ with $I_{i}=J$; that is, $j_{1}$ is large enough so that all elements of $\mathscr{\mathscr { L }}$ have been met in the past, and only elements of $\mathscr{J}$ will be met in the future. Let $j_{2}>j_{1}$ be large enough so that all elements of $\mathscr{J}$ are represented among $I_{j_{1}+1}, \ldots, I_{j_{2}-1}$. Let $x_{0}, x_{1}, \ldots, x_{j_{2}}$ be a $j_{2}$-admissible sequence. Then $x_{j_{1}} \in \bigcap \mathscr{F}$, because $a$ cannot decrease, each interval in $\mathscr{F}$ contains at least one $x_{i}$ with $i<j_{1}$, and each interval in $\mathscr{J}$ contains at least one $x_{i}$ with $j_{1}<i<j_{2}$. If $0 \in$ flow, then $x_{0}, x_{1}, \ldots, x_{j_{1}}^{\omega}$ is an admissible sequence. If $0 \notin$ flow, then $x_{j_{1}}$ $<\sup (\cap \mathscr{J})$. Let $\delta=\sup (\bigcap \mathscr{J})-x_{j_{1}}$. For each $i>j_{1}$, choose $\delta_{i}$ so that $0<\delta_{i}<\delta /\left(2^{i} \cdot t_{i}\right)$. Then $x_{0}, x_{1}, \ldots, x_{j_{1}}, x_{j_{1}}+$ $\delta_{j_{1}+1} \cdot t_{j_{1}+1}, x_{j_{1}}+\delta_{j_{1}+1} \cdot t_{j_{1}+1}+\delta_{j_{1}+2} \cdot t_{j_{1}+2}, \ldots$ is admissible.

Case $0 \in \operatorname{interior}($ flow $)$ and $\cap \mathscr{J} \neq \varnothing$. Since 0 is in the interior of flow, every $m$-admissible sequence can be slowed down to give another $m$-admissible sequence. Let $j_{1}$ be as in the previous case. Since $0 \in$ flow, whenever a $\left(j_{1}+i\right)$ admissible sequence, $i \geqslant 0$, terminates in $\cap \mathscr{J}$, then it can be extended to an admissible sequence by repeating the last value ad infinitum. Such a $\left(j_{1}+i\right)$-admissible sequence terminating in $\bigcap \mathscr{J}$ exists, because there exist two intervals $J_{1}, J_{2} \in \mathscr{J}$ such that $\inf \left(J_{1}\right)=\inf (\cap \mathscr{J})$ with identical strictness, and $\sup \left(J_{2}\right)=\sup (\cap \mathscr{J})$ with identical strictness. Let $j_{2}$ be large enough so that both $J_{1}$ and $J_{2}$ each appear twice in $I_{j_{1}+1}, \ldots, I_{j_{2}-1}$. Let $x_{0}, x_{1}, \ldots, x_{j_{2}}$ be a $j_{2}$-admissible sequence. There must be two positions $i_{1}$ and $i_{2}$ with $j_{1}<i_{1}$ $<i_{2}<j_{2}$ such that $x_{i_{1}} \in J_{1}$ and $x_{i_{2}} \in J_{2}$. By slowing down the $x_{i}$ sequence, $\cap \mathscr{J}$ can be reached: if $x_{i_{1}} \geqslant \sup (\cap \mathscr{F})$ and $x_{i_{2}} \leqslant \inf (\cap \mathscr{F})$, then for some $j$ with $i_{1} \leqslant j<i_{2}$, we have $x_{j} \geqslant \sup (\cap \mathscr{J})$ and $x_{j+1}<\sup (\cap \mathscr{J})$. Choose $y \in \bigcap \mathscr{J}$ so that $y>x_{j+1}$. Then $x_{0}, x_{1}, \ldots, x_{j}, y^{\omega}$ is an admissible sequence.

Case $0 \in$ interior (flow) and $\cap \mathscr{J}=\varnothing$. Let $J_{1}, J_{2} \in \mathscr{J}$ be such that every element of $J_{1}$ is greater than every element of $J_{2}$. Let $j_{1}$ be as in the previous two cases. Let $j_{1}<p_{1}<q_{1}$ $<p_{2}$ be so that $I_{p_{1}}=I_{p_{2}}=J_{1}$ and $I_{q_{1}}=J_{2}$. Let $x_{0}, x_{1}, \ldots, x_{p_{2}}$ be a $p_{2}$-admissible sequence. We first show that for every $m \in \mathbb{N}$, there is an $m$-admissible sequence starting from $x_{0}$. This is obvious for $m \leqslant p_{2}$, so suppose that $m>p_{2}$. Let
$y_{0}, y_{1}, \ldots, y_{m}$ be an $m$-admissible sequence. We have three subcases.

Subcase $x_{p_{1}}=y_{p_{1}}$. In this case $x_{0}, x_{1}, \ldots, x_{p_{1}}, y_{p_{1}+1}$, $y_{p_{1}+2}, \ldots, y_{m}$ is $m$-admissible.

Subcase $x_{p_{1}}<y_{p_{1}}$. 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 $J_{1}$ to $J_{2}$. If $x_{q_{1}}>y_{q_{1}}$, then for some $j \in\left\{p_{1}+1, \ldots, q_{1}\right\}$, we have $x_{j}<y_{j}$ and $x_{j+1} \geqslant y_{j+1}$. Since $\left(y_{j+1}-y_{j}\right) / t_{j} \in$ flow and $\left(x_{j+1}-x_{j}\right) / t_{j} \in$ flow, and $y_{j+1}-y_{j}$ $<y_{j+1}-x_{j}<x_{j+1}-x_{j}$, it must be that $\left(y_{j+1}-x_{j}\right) / t_{j} \in$ flow. Hence, the sequence $x_{0}, x_{1}, \ldots, x_{j}, y_{j+1}, y_{j+2}, \ldots, y_{m}$ is $m$-admissible.

Subcase $x_{p_{1}}>y_{p_{1}}$. 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 $J_{1}$ to $J_{2}$. So suppose that $x_{q_{1}}>y_{q_{1}}$. 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 $J_{2}$ to $J_{1}$. If $x_{p_{2}}<y_{p_{2}}$, then the $y_{i}$ sequence must cross the $x_{i}$ sequence as above, and the $x_{0}, x_{1}, \ldots, x_{j}, y_{j+1}$, $y_{j+2}, \ldots, y_{m}$ construction from the previous subcase provides an $m$-admissible sequence. This completes the case analysis.

It remains to construct an infinite admissible sequence. Let $x_{0} \in I_{0}$ be such that for every $m \in \mathbb{N}$, there is an $m$-admissible sequence starting from $x_{0}$. Let $R$ be the set of $t_{0}$-successors of $x_{0}$; that is, $R=\left\{y \in \mathbb{R} \mid\left(y-x_{0}\right) / t_{0} \in\right.$ flow $\}$. Since flow is bounded, $R$ is bounded. Let $\left(t_{i}^{\prime}\right)_{i \in \mathbb{N}}$ be the sequence of reals such that $t_{i}^{\prime}=t_{i+1}$ for all $i \geqslant 0$. Let $\left(I_{i}^{\prime}\right)_{i \in \mathbb{N}}$ be the sequence of intervals such that $I_{0}^{\prime}=I_{1} \cap R$, and $I_{i}^{\prime}=I_{i+1}$ for all $i \geqslant 1$. If we apply what we have already proven to the duration sequence $\left(t_{i}^{\prime}\right)_{i \in \mathbb{N}}$ and the interval sequence $\left(I_{i}^{\prime}\right)_{i \in \mathbb{N}}$, we conclude that there is an $x_{1} \in I_{0}^{\prime}$ such that for every $m \in \mathbb{N}$, there is an $m$-admissible (with respect to $\left(t_{i}^{\prime}\right)_{i \in \mathbb{N}}$ and $\left.\left(I_{i}^{\prime}\right)_{i \in \mathbb{N}}\right)$ sequence starting from $x_{1}$. Continuing inductively, we form an admissible sequence starting from $x_{0}$.

Now the proof of the main theorem consists of reducing to one dimension, eliminating discontinuous jumps, and applying Lemma 3.26.

Theorem 3.27. If $A$ is an initialized rectangular automaton with bounded nondeterminism, then the $\omega$-language $\operatorname{Lang}(A)$ closed under divergent limits.

Proof. Let $A$ be an $n$-dimensional initialized rectangular automaton with bounded nondeterminism. By Proposition 3.19, it suffices to show that the edge $\omega$-language $\operatorname{Lang}_{E}(A)$ is closed under divergent limits. Suppose that $\bar{\pi} \in\left(E \cup \mathbb{R}_{\geqslant 0}\right)^{\omega}$ is a divergent timed edge word such that for
every $m \in \mathbb{N}$, there is a finite edge run of $A$ of the form $q_{0}^{m} \xrightarrow{\pi_{0}} q_{1}^{m} \xrightarrow{\pi_{1}} \cdots \xrightarrow{\pi_{m}} q_{m+1}^{m}$. We need to show that there is an infinite edge run $\rho$ of $A$ of the form $q_{0} \xrightarrow{\pi_{0}} q_{1} \xrightarrow{\pi_{1}} q_{2} \xrightarrow{\pi_{2}} \cdots$. First, observe that each of the $n$ coordinates of a rectangular automaton is independent of the other coordinates. Hence, $A$ has an edge run $\rho$ iff each of the $n 1 \mathrm{D}$ rectangular automata defined by projecting the continuous state of $A$ to one of the coordinates has the corresponding projection of $\rho$ as an edge run. Therefore, without loss of generality, we assume that $n=1$.

Let $\theta=\left\{i \in \mathbb{N} \mid 1 \in \operatorname{jump}\left(\pi_{i}\right)\right\}$ be the set of positions with discontinuous jumps in the first (and only) coordinate. If $i \in \theta$, then the value of the continuous state after the $i$ th step is independent of its previous value. So if $\theta$ is infinite, then we can string together $\rho$ from infinitely many segments that lead from one discontinuous jump to the next: for all $j \geqslant 0$, choose $q_{j}=q_{j}^{f(j)}$ for $f(j)=\min \{i \in \theta \mid i \geqslant j\}$. If $\theta$ is finite, then we can string together $\rho$ from $|\theta|$ many segments that lead, as in the previous case, from one discontinuous jump to the next, followed by an application of Lemma 3.26. In Lemma 3.26, let $\mathscr{I}$ be the set of all invariant, preguard, and postguard intervals of $A$; let flow be the flow interval of the vertex associated with the state $q_{j}^{k}$ for any $k \geqslant j>\max (\theta)$; and let $I_{0}$ be the interval associated with the zone Post ${ }^{\pi_{0} \pi_{1} \cdots \pi_{\max }(\theta)}($ Init $)$. The initialization of $A$ ensures that after the first $\max (\theta)$ steps, the flow interval flow remains constant. The bounded nondeterminism of $A$ ensures, by Fact 3.25, that the zone Post $^{\pi_{0} \pi_{1} \cdots \pi_{\max }(\theta)}($ Init $)$ is bounded and rectangular.

Corollary 3.28. The $\omega$-language emptiness problem for initialized rectangular automata with bounded nondeterminism is complete for PSPACE.

### 3.4. Simulation Relations

We introduced several mappings between the state spaces of rectangular automata. We were interested only that the mappings preserve reachability and $\omega$-languages. Now we study the mappings in greater detail and show that they are timed (bi)simulations [LV96] on the underlying labeled transition systems. In particular the map $\alpha$ from Section 3.1 specifies a timed bisimulation between an initialized stopwatch automaton $C$ and the timed automaton $D_{C}$, the map $\beta$ from Section 3.1 specifies a timed bisimulation between an initialized singular automaton $B$ and the initialized stopwatch automaton $C_{B}$, and the map $\gamma$ from Section 3.2 specifies a timed forward simulation of an initialized rectangular automaton $A$ by the initialized singular automaton $B_{A}$, as well as a timed backward simulation of $B_{A}$ by $A$.

Let $A_{1}$ and $A_{2}$ be two rectangular automata with $\varepsilon$ moves and the same observation alphabet, $\Sigma$. A binary relation $\chi \subset \operatorname{Reach}\left(A_{1}\right) \times \operatorname{Reach}\left(A_{2}\right)$ is a timed forward simulation of $A_{2}$ by $A_{1}$ if the following two conditions are met:

1. For every initial state $r$ of $A_{2}$, there is an initial state $q$ of $A_{1}$ such that $(q, r) \in \chi$.
2. For all states $r, r^{\prime} \in \operatorname{Reach}\left(A_{2}\right)$, every state $q \in \operatorname{Reach}\left(A_{1}\right)$ with $(q, r) \in \chi$, and every label $\tau \in \Sigma \cup \mathbb{R}_{\geqslant 0}$, if $r \xrightarrow{\tau} A_{2} r^{\prime}$, then there exists a state $q^{\prime} \in \operatorname{Reach}\left(A_{1}\right)$ such that $\left(q^{\prime}, r^{\prime}\right) \in \chi$ and $q \xrightarrow{\tau} A_{1} q^{\prime}$.

The relation $\chi$ is a timed backward simulation of $A_{2}$ by $A_{1}$ if the following three conditions are met:

1. For every state $r \in \operatorname{Reach}\left(A_{2}\right)$, there exists a state $q \in \operatorname{Reach}\left(A_{1}\right)$ with $(q, r) \in \chi$.
2. For every initial state $r$ of $A_{2}$ and every state $q \in$ $\operatorname{Reach}\left(A_{1}\right)$, if $(q, r) \in \chi$, then $q$ is an initial state of $A_{1}$.
3. For all states $r, r^{\prime} \in \operatorname{Reach}\left(A_{2}\right)$, every state $q^{\prime} \in$ $\operatorname{Reach}\left(A_{1}\right)$ with $\left(q^{\prime}, r^{\prime}\right) \in \chi$, and every label $\tau \in \Sigma \cup \mathbb{R}_{\geqslant 0}$, if $r \xrightarrow{\tau} A_{A_{2}} r^{\prime}$, then there exists a state $q \in \operatorname{Reach}\left(A_{1}\right)$ such that $(q, r) \in \chi$ and $q \xrightarrow{\tau} A_{1} q^{\prime}$.

If $\chi$ is a timed forward simulation of $A_{2}$ by $A_{1}$, and $\chi^{-1}$ is a timed forward simulation of $A_{1}$ by $A_{2}$, then $\chi$ is a timed bisimulation between $A_{1}$ and $A_{2}$. Notice that, if there is a timed forward simulation of $A_{2}$ by $A_{1}$, then every timed word accepted by $A_{2}$ is accepted also by $A_{1}$; if there is a timed backward simulation of $A_{2}$ by $A_{1}$, then every finite timed word accepted by $A_{2}$ is accepted also by $A_{1}$.

For a map $\kappa$ : $Q_{A_{1}} \rightarrow Q_{A_{2}}$ from the states of $A_{1}$ to the states of $A_{2}$, define the relation $\hat{\kappa} \subset \operatorname{Reach}\left(A_{1}\right) \times \operatorname{Reach}\left(A_{2}\right)$ such that $(q, r) \in \hat{\kappa}$ iff $r=\kappa(q)$. In this way, we obtain the relations $\hat{\alpha}$ and $\hat{\beta}$. For a map $\kappa: Q_{A_{1}} \rightarrow 2^{Q_{A_{2}}}$ from the states of $A_{1}$ to zones of $A_{2}$, define the relation $\hat{\kappa} \subset \operatorname{Reach}\left(A_{1}\right) \times \operatorname{Reach}\left(A_{2}\right)$ such that $(q, r) \in \hat{\kappa}$ iff $r \in \kappa(q)$. In this way, we obtain the relation $\hat{\gamma}$. The next proposition follows from Lemma 3.2 in the case of $\hat{\alpha}$, from Lemma 3.4 in the case of $\hat{\beta}$, and from Lemmas 3.13, 3.14, and 3.17 in the case of $\hat{\gamma}$.

Proposition 3.29. For every initialized stopwatch automaton $C$ with $\varepsilon$ moves, the relation $\hat{\alpha}$ is a timed bisimulation between $C$ and the timed automaton $D_{C}$. For every initialized singular automaton $B$ with $\varepsilon$ moves, the relation $\hat{\beta}$ is a timed bisimulation between B and the initialized stopwatch automaton $D_{C}$. For every initialized rectangular automaton $A$, the relation $\hat{\gamma}$ is a timed forward simulation of $A$ by $B_{A}$, and $\hat{\gamma}^{-1}$ is a timed backward simulation of $B_{A}$, restricted to its upperhalf space, by $A$.

Simulation relations compose, as summarized in Figs. 11 and 12. In particular, the timed automaton $D_{C_{B_{A}}}$ timed forward simulates the initialized rectangular automaton $A$ via the relation $\hat{\alpha} \circ \hat{\beta} \circ \hat{\gamma}$, and $A$ timed backward simulates $D_{C_{B_{A}}}$ via $\hat{\gamma}^{-1} \circ \hat{\beta} \circ \hat{\alpha}$. It is not difficult to check that $A$ does not timed forward simulate $B_{A}$, and $B_{A}$ does not timed backward simulate $A$ [Kop96].


FIG. 11. Chain of timed simulations from $A$ to $D_{C_{B_{A}}}$.

### 3.5. Symbolic Reachability Analysis

Consider an $n$-dimensional rectangular automaton $A$, and a rectangular zone $Z_{f}$ of $A$. To solve the reachability problem for $A$ and $Z_{f}$ symbolically, by computing with multirectangular zones, we may attempt to compute the sequence Init, Post(Init), $\operatorname{Post}^{2}($ Init $), \ldots$ of zones, until either the intersection with $Z_{f}$ is nonempty, or a fixpint of Post is reached within a finite number of steps (that fixpoint, then, is Reach $(A)$ ). This procedure, which we call the symbolic execution of $A\left[\mathrm{ACH}^{+} 95\right]$, will terminate if the zone $Z_{f}$ is reachable or if there is a natural number $i \in \mathbb{N}$ such that $\operatorname{Reach}(A)=\operatorname{Post}^{i}($ Init $)$, but it will not terminate if $Z_{f}$ is not reachable and no such $i$ exists. Symbolic execution therefore constitutes a semidecision procedure for the reachability problem of rectangular automata. The procedure has been implemented in the automatic verification tool НyTech [HHWT97], and successfully applied to examples of practical interest [HHWT95, HW95, NS95, Cor96, HWT96, SMF97].

While the reachability problem is decidable for all initialized rectangular automata, even for timed automata symbolic execution does not always terminate. To see this, consider the 2D timed automaton $\hat{B}$ from Fig. 13, with first coordinate $c$ and second coordinate $d$. The only vertex of $\hat{B}$, called $v$, has the invariant region $[0, \infty) \times[0,1]$. For each $i \geqslant 0, \operatorname{Post}_{\hat{B}}^{2 i}\left(\operatorname{Init}_{\hat{B}}\right)=\{(v,(x, y)) \mid 0 \leqslant x \leqslant i$, and either
$y=x-\lfloor x\rfloor$ or $y=x \in \mathbb{N}\}$. Hence the fixpoint computation does not converge. To blame is the unbounded invariant region. This is because symbolic execution is known to terminate for every timed automaton with bounded invariant regions [HNSY94] (where $A$ has bounded invariant regions if for every vertex $v$ of $A, \operatorname{inv}(v)$ if bounded). It follows that symbolic execution terminates also for initialized rectangular automata with bounded invariant regions. This is the content of the next proposition.

Proposition 3.30. For every initialized rectangular automaton $A$ with $\varepsilon$ moves and bounded invariant regions, there is a natural number $i \in \mathbb{N}$ such that $\operatorname{Reach}(A)=\operatorname{Post}^{i}($ Init $)$.

Proof. Consider an initialized rectangular automaton $A$ with $\varepsilon$ moves and bounded invariant regions. The construction of the singular automaton $B_{A}$ from Section 3.2 can be modified in two respects. First, it is a simple matter to accommodate inherited $\varepsilon$ edges from $A$. Second, $B_{A}$ is equipped with bounded invariant regions as follows, by introducing additional $\varepsilon$ edges. Whenever a lower-bound variable $b_{\ell(i)}$ falls to 1 below the lower boundary of the bounded invariant interval for $a_{i}$, the slope of $b_{\ell(i)}$ is changed to 0 via an $\varepsilon$ edge. Upper-bound variables are treated symmetrically. These modifications do not affect the mapping $\gamma$ and its properties. Now the proposition follows


FIG. 12. Putting together the commuting diagrams.


FIG. 13. The timed automaton $\hat{B}$ for which $\operatorname{Reach}(\hat{B}) \ni$ Post $^{i}($ Init $)$ for all $i \geqslant 0$.
immediately from the relationships shown in Fig. 12 and the statement of the proposition for timed automata.

In this section, we remedy the problems arising from unbounded invariant regions by preprocessing the given automaton $A$. For initialized $A$, we construct in linear time an initialized rectangular automaton $A_{b d}$ with $\varepsilon$ moves such that (1) $\operatorname{Reach}(A) \cap Z_{f} \neq \varnothing$ iff $\operatorname{Reach}\left(A_{b d}\right) \cap Z_{f} \neq \varnothing$, and (2) there is a natural number $i \in \mathbb{N}$ such that $\operatorname{Reach}\left(A_{b d}\right)=$ Post $_{A_{b d}}^{i}\left(\right.$ Init $\left._{A_{b d}}\right)$. The second condition implies that the symbolic execution of $A_{b d}$ terminates, and the first condition implies that it gives the correct answer to the reachability problem for $A$. Consequently, the reachability problem for $A$ can be solved, rather than by translating $A$ into a timed automaton, by direct symbolic execution of $A_{b d}$. While translation doubles the dimension, the dimension of $A_{b d}$ remains $n$, which alleviates a major practical bottleneck in the verification of hybrid systems [HHWT97].

To facilitate the proof of condition (2), we first introduce a third automaton $A_{b d}^{\prime}$, which ratifies both (1) and (2) but is exponentially larger than $A$. The automaton $A_{b d}^{\prime}$ will have bounded invariant regions, and therefore satisfy condition (2) by Proposition 3.30. For the remainder of this section, assume that the given automaton $A$ is initialized, and that its variables are $a_{1}, \ldots, a_{n}$.

## An Exponential Preprocessing Step

We define an $n$-dimensional initialized rectangular automaton $A_{b d}^{\prime}$ with $\varepsilon$ moves and bounded invariant regions, and a rectangular zone $Z_{f}^{\prime}$ of $A_{b d}^{\prime}$, such that $\operatorname{Reach}(A) \cap Z_{f} \neq \varnothing$ iff $\operatorname{Reach}\left(A_{b d}^{\prime}\right) \cap Z_{f}^{\prime} \neq \varnothing$. Let $h$ be 1 more than the largest rational constant that appears in the definitions of $A$ and $Z_{f}$ as a finite endpoint of an interval. Let $g$ be 1 less than the smallest such constant. The idea is to truncate all invariant, preguard, and postguard regions of $A$ by intersection with $[g, h]^{n}$. When a variable reaches the upper or lower boundary of the rectangular region $[g, h]^{n}$, we change its slope to 0 . The automaton $A_{b d}^{\prime}$ has the vertex set $V_{A_{b d}^{\prime}}=V_{A} \times\{0,1,2\}^{n}$. Put low $=0$, ok $=1$, and $\operatorname{high}=2$, and let $o k^{n}$ be the $n$-vector $(o k, o k, \ldots, o k)$. Each state $((v, \boldsymbol{\lambda}), \mathbf{x})$ of $A_{b d}^{\prime}$ is intended to represent all states of $A$ of the form $(v, \mathbf{y})$ with $y_{i} \leqslant g$ if $\lambda_{i}=$ low,
and $y_{i}=x_{i} \in[g, h]$ if $\lambda_{i}=o k$, and $y_{i} \geqslant h$ if $\lambda_{i}=h i g h$. All of these states will be shown equivalent for reachability purposes.

The remaining components of $A_{b d}^{\prime}$ are defined as follows. The observation alphabet of $A_{b d}^{\prime}$ is the same as for $A$. The initial, invariant, and flow functions of $A_{b d}^{\prime}$ are defined by

$$
\begin{aligned}
& \text { init }_{A_{b d}^{\prime}}(v, \lambda)_{i}= \begin{cases}\text { init }_{A}(v)_{i} \cap[g, h], & \text { if } \lambda_{i}=o k, \\
\varnothing, & \text { otherwise; }\end{cases} \\
& \operatorname{inv}_{A_{b d}^{\prime}}(v, \boldsymbol{\lambda})_{i}=\left\{\begin{array}{l}
{[h, h],} \\
\operatorname{inv}_{A}(v)_{i} \cap[g, h], \\
{[g, g],}
\end{array}\right. \\
& \text { if } \lambda_{i}=h i g h, \\
& \text { if } \quad \lambda_{i}=o k \text {, } \\
& \text { if } \lambda_{i}=\text { low; } \\
& \text { flow }_{A_{b d}^{\prime}}(v, \boldsymbol{\lambda})_{i}=\left\{\begin{array}{l}
\text { flow }_{A}(v)_{i}, \\
{[0,0],}
\end{array}\right. \\
& \text { if } \quad \lambda_{i}=o k \text {, } \\
& \text { otherwise. }
\end{aligned}
$$

For each edge $e=(v, w)$ of $A$, the automaton $A_{b d}^{\prime}$ has an edge $e^{\prime}=\left(\left(v, o k^{n}\right),\left(w, o k^{n}\right)\right)$ with $o b s_{A_{b d}^{\prime}}\left(e^{\prime}\right)=o b s_{A}(e)$, $\operatorname{pre}_{A_{b d}^{\prime}}\left(e^{\prime}\right)=\operatorname{pre}_{A}(e) \cap[g, h]^{n}, \operatorname{post}_{A_{b d}^{\prime}}\left(e^{\prime}\right)=\operatorname{post}_{A}(e) \cap$ $[g, h]^{n}$, and jump $A_{A_{b d}^{\prime}}\left(e^{\prime}\right)=j u m p_{A}(e)$. Define trunc: $\mathbb{R}^{n} \rightarrow$ $[g, h]^{n}$ by $\operatorname{trunc}(\mathbf{x})_{i}=g$ if $x_{i}<g, \operatorname{trunc}(\mathbf{x})_{i}=x_{i}$ if $g \leqslant x_{i} \leqslant h$, and $\operatorname{trunc}(\mathbf{x})_{i}=h$ if $x_{i}>h$. A rectangular region $R \subset \mathbb{R}^{n}$ is gh-limited if for every coordinate $i \in\{1, \ldots, n\}$, (1) either $\inf \left(R_{i}\right)$ $=-\infty$ or $g+1 \leqslant \inf \left(R_{i}\right) \leqslant h-1$, and (2) either $\sup \left(R_{i}\right)=\infty$ or $g+1 \leqslant \sup \left(R_{i}\right) \leqslant h-1$. By definition of $g$ and $h$, all initial, invariant, preguard, and postguard regions of $A$, as well as $\left[Z_{f}\right]^{v}$ for all vertices $v$ of $A$, are $g h$-limited regions. The following fact ensures that the guards of $A_{b d}^{\prime}$ have the same effect as the guards of $A$.

Fact 3.31. Let $\mathbf{x} \in \mathbb{R}^{n}$, and let $R \subset \mathbb{R}^{n}$ be a $g h$-limited rectangular region. Then $\mathbf{x} \in R$ iff $\operatorname{trunc}(\mathbf{x}) \in R \cap[g, h]^{n}$.

In addition to the edges inherited from $A$, the automaton $A_{b d}^{\prime}$ has $\varepsilon$ edges for toggling the components of $\lambda$. For every vertex $v$ of $A$, every coordinate $i \in\{1, \ldots, n\}$, and every vector $\lambda$ with $\lambda_{i}=o k$, there is an $\varepsilon$ edge from $(v, \lambda)$ to $\left(v, \lambda\left[\lambda_{i}:=l o w\right]\right)$, and another $\varepsilon$ edge in the reverse direction, from ( $\left.v, \lambda\left[\lambda_{i}:=l o w\right]\right)$ to $(v, \lambda)$, each annotated with the guarded command $a_{i}=g \rightarrow a_{i}:=g$. Here $\lambda\left[\lambda_{i}:=\right.$ low $]$ stands for the vector that agrees with $\lambda$ on all components except for the $i$ th component, which is low. The trivial assignment is for initialization. Similarly, there are $\varepsilon$ edges from $(v, \lambda)$ to $\left(v, \lambda\left[\lambda_{i}:=\operatorname{high}\right]\right)$ and from $\left(v, \lambda\left[\lambda_{i}:=\operatorname{high}\right]\right)$ to $(v, \lambda)$, each annotated with the guarded command $a_{i}=h \rightarrow a_{i}:=h$. This completes the definition of $A_{b d}^{\prime}$. Note that the rectangular automaton $A_{b d}^{\prime}$ is initialized and has bounded invariant regions.

Define the function $\zeta: Q_{A} \rightarrow Q_{A_{b d}^{\prime}}$ by $\zeta(v, \mathbf{x})=\left(\left(v, o k^{n}\right)\right.$, $\operatorname{trunc}(\mathbf{x}))$, and extend $\zeta$ to zones in the natural way. Notice that Init $_{A_{b d}^{\prime}}=\zeta\left(\right.$ Init $\left._{A}\right)$. Define $Z_{f}^{\prime}=\zeta\left(Z_{f}\right)$.

Lemma 3.32. Let $A$ be an initialized rectangular automaton, and let $Z_{f}$ be a rectangular zone of $A$. Then $\operatorname{Reach}(A)$ $\cap Z_{f} \neq \varnothing$ iff $\operatorname{Reach}\left(A_{b d}^{\prime}\right) \cap Z_{f}^{\prime} \neq \varnothing$.

Proof. It suffices to prove the lemma for one-dimensional $A$. For the "only if" direction, we show that for all labels $\pi \in E_{A} \cup \mathbb{R}_{\geqslant 0}$, if $q \xrightarrow{\pi}{ }_{A} r$ then $\zeta(q) \xrightarrow{\pi} A_{b d}^{\prime} \zeta(r)$. For time steps, suppose that $\left(v, x^{1}\right) \xrightarrow{t} A\left(v, x^{2}\right)$ for $t>0$. Then $\left(x^{2}-x^{1}\right) / t \in$ flow $_{A}(v)$. If $g \leqslant x^{1}, x^{2} \leqslant h$, then $\operatorname{trunc}\left(x^{1}\right)=x^{1}$ and $\operatorname{trunc}\left(x^{2}\right)=x^{2}$. Since flow $_{A_{b d}^{\prime}}(v, o k)=$ flow $_{A}(v)$, we have $\zeta\left(v, x^{1}\right) \xrightarrow{t}_{A_{b d}^{\prime}} \zeta\left(v, x^{2}\right)$. Now suppose that $x^{1} \leqslant g \leqslant h \leqslant x^{2}$. Then there exist three durations $t_{1}, t_{2}, t_{3} \in \mathbb{R}_{\geqslant 0}$ such that $t=t_{1}+t_{2}+t_{3}$ and $\left(v, x^{1}\right) \xrightarrow{t_{1}}{ }_{A}(v, g) \xrightarrow{t_{2}}(v, h) \xrightarrow{t_{3}}{ }_{A}\left(v, x^{2}\right)$. In this case, $\operatorname{trunc}\left(x^{1}\right)=g$ and $\operatorname{trunc}\left(x^{2}\right)=h$. Since flow $_{A_{b d}^{\prime}}(v, o k)=$ flow $_{A}(v)$, we have

$$
\begin{aligned}
\left((v, \text { ok }), \operatorname{trunc}\left(x^{1}\right)\right) & \xrightarrow{\varepsilon} A_{A_{b d}^{\prime}}((v, \text { low }), g) \xrightarrow{t_{1}} A_{b d}^{\prime}((v, \text { low }), g) \\
& \xrightarrow{\varepsilon} A_{b d}^{\prime}((v, o k), g) \xrightarrow{t_{2}} A_{b d}^{\prime}((v, o k), h) \\
& \left.\xrightarrow{\varepsilon} A_{b d}^{\prime}(v, \text { high }), h\right) \xrightarrow{t_{3}} A_{b d}^{\prime}((v, \text { high }), h) \\
& \xrightarrow{\varepsilon} A_{b d}^{\prime}\left((v, o k), \operatorname{trunc}\left(x^{2}\right)\right) .
\end{aligned}
$$

Again $\zeta\left(v, x^{1}\right) \xrightarrow{t}{ }_{A_{b d}^{\prime}} \zeta\left(v, x^{2}\right)$. Other positions of $x^{1}$ and $x^{2}$ relative to $g$ and $h$ are handled similarly. For edge steps, the key fact is that all preguard and postguard regions are $g h$-limited. Suppose that $\left(v^{1}, x^{1}\right) \xrightarrow{e}{ }_{A}\left(v^{2}, x^{2}\right)$ for $e \in E_{A}$. Then $x^{1} \in \operatorname{pre}_{A}(e)$ and $x^{2} \in \operatorname{post}_{A}(e)$, and so by Fact 3.31, $\operatorname{trunc}\left(x^{1}\right) \in \operatorname{pre}_{A_{b d}^{\prime}}\left(e^{\prime}\right)$ and $\operatorname{trunc}\left(x^{2}\right) \in \operatorname{post}_{A_{b d}^{\prime}}\left(e^{\prime}\right)$, where $e^{\prime}$ is the edge of $A_{b d}^{\prime}$ inherited from $e$. In addition, jump $A_{A_{b d}^{\prime}}\left(e^{\prime}\right)=$ $j u m p_{A}(e)$, and so $\zeta\left(v^{1}, x^{1}\right) \xrightarrow{e^{\prime}} A_{b d}^{\prime} \zeta\left(v^{2}, x^{2}\right)$.

For the "if" direction we apply a more global argument. Suppose that

$$
\begin{gathered}
\left(\left(v^{0}, o k\right), y^{0}\right) \xrightarrow{\stackrel{\pi}{1}^{A_{b d}^{\prime}}}\left(\left(v^{1}, o k\right), y^{1}\right) \xrightarrow{\pi_{A_{b d}}} \ldots \\
\\
\xrightarrow[A_{b d}^{\prime}]{\pi_{m}}\left(\left(v^{m}, o k\right), y^{m}\right),
\end{gathered}
$$

where $\left(\left(v^{0}, o k\right), y^{0}\right) \in$ Init $_{A_{b d}^{\prime}}$ and $\left(\left(v^{m}, o k\right), y^{m}\right) \in Z_{f}^{\prime}$. We find $x^{0}, \ldots, x^{m}$ such that

$$
\left(v^{0}, x^{0}\right) \xrightarrow{\pi_{1}}\left(v^{1}, x^{1}\right) \xrightarrow{\pi_{2}}{ }_{A} \ldots \xrightarrow{\pi_{m}}\left(v^{m}, x^{m}\right)
$$

and for each $i \in\{0, \ldots, m\}, \operatorname{trunc}\left(x^{i}\right)=y^{i}$. Then, by Fact 3.31, $\left(v^{0}, x^{0}\right) \in \operatorname{Init}_{A}$ and $\left(v^{m}, x^{m}\right) \in Z_{f}$. First, since $A$ is initialized, it suffices to assume that each $\pi_{i} \in E_{A_{b d}^{\prime}}$ has $j \operatorname{jump}\left(\pi_{i}\right)=\varnothing$ (otherwise we string together the solutions obtained for the segments between discontinuous jumps); consequently, flow $_{A}\left(v^{i}\right)=$ flow $_{A}\left(v^{j}\right)$ for all $i, j \in\{0, \ldots, m\}$. Second, by Fact 3.31, it suffices to assume that $\pi_{i} \in \mathbb{R}_{\geqslant 0}$ for all $i$; that is, all $m$ steps are time steps. Let flow be the common value of the flow region flow $_{A}\left(v^{i}\right)$, for $i \in\{0, \ldots, m\}$. If $0 \in$ flow, then

$$
\left(v^{0}, y^{0}\right) \xrightarrow{\pi_{1}}\left(v^{1}, y^{1}\right) \xrightarrow{\pi_{2}} A \cdots \xrightarrow{\pi_{m}}\left(v^{m}, y^{m}\right),
$$

so putting $x^{i}=y^{i}$ for each $i$, we are finished. Now suppose that flow $\subset(0, \infty)$. Here the most interesting case is given by $y^{0}=g$ and $y^{m}=h$ for $g<h$. In this case, there exist $0 \leqslant j_{1} \leqslant j_{2}<m$ such that $y^{i}=g$ for $i \leqslant j_{1}$, and $g<y^{i}<h$ for $j_{1}<i \leqslant j_{2}$, and $y^{i}=h$ for $i>j_{2}$. We put $x^{i}=y^{i}$ for $j_{1}<i \leqslant j_{2}$. To set the $x^{i}$ for $i>j_{2}$, we need only determine a suitable slope. Let $k_{2}$ be such that for some $h^{\prime} \geqslant h$, we have $k_{2}=$ $\left(h^{\prime}-y^{j_{2}}\right) / \pi_{j_{2}+1} \in$ flow. Such a $k_{2}>0$ exists because $\left(\left(v^{j_{2}}, o k\right), y^{j_{2}}\right) \xrightarrow{\pi_{j_{2}+1}} A_{b_{b d}^{\prime}}\left(\left(v^{j_{2}+1}, o k\right), h\right)$. Put $x^{i}=y^{j_{2}}+k_{2}$. $\left(i-j_{2}\right)$ for each $i>j_{2}$. Then $\left(v^{i}, x^{i}\right) \xrightarrow{\pi_{i+1}}{ }_{A}\left(v^{i+1}, x^{i+1}\right)$ for all $i \in\left\{j_{2}, j_{2}+1, \ldots, m-1\right\}$. It remains to set the $x^{i}$ for $i \leqslant j_{1}$, which is done in the same way. Since $\left(\left(v^{j_{1}}, o k\right), g\right)$ $\xrightarrow{\pi_{j_{1}+1}}\left(\left(v^{j_{1}+1}, o k\right), y^{j_{1}+1}\right)$, there exists a $k_{1} \in f l o w ~ s u c h ~$ that for some $g^{\prime} \leqslant g$, we have $k_{1}=\left(y^{j_{1}+1}-g^{\prime}\right) / \pi_{j_{1}+1} \in$ flow. For each $i \in\left\{0, \ldots, j_{1}\right\}$, put $x^{i}=y^{j_{1}+1}-k_{1} \cdot\left(j_{1}+1-i\right)$. Then $\left(v^{i}, x^{i}\right) \xrightarrow{\pi_{i+1}}\left(v^{i+1}, x^{i+1}\right)$ for all $i \in\left\{0, \ldots, j_{1}\right\}$, and we are finished. All other cases are handled in a similar fashion.

## A Linear Preprocessing Step

The automaton $A_{b d}^{\prime}$ uses the discrete part of the state to store information about variables whose values are too small or too large to be relevant. This causes the vertex set of $A_{b d}^{\prime}$ to be exponentially larger than the vertex set of $A$. We now define an initialized rectangular automaton $A_{b d}$, with the same dimension and vertex set as $A$, which uses the continuous part of the state for the same purpose. Instead of stopping a variable when it reaches $g$ (resp. $h$ ), the automaton $A_{b d}$ allows a nondeterministic jump to any value below $g$ (resp. above $h$ ). Formally, $A_{b d}$ is identical to $A$, except for $2 n \cdot\left|V_{A}\right|$ additional $\varepsilon$ edges. For every vertex $v$ of $A$, and every coordinate $i \in\{1, \ldots, n\}$, the automaton $A_{b d}$ has two $\varepsilon$ edges from $v$ to $v$, which are annotated respectively with the two guarded commands $a_{i} \leqslant g \rightarrow a_{i}: \in(-\infty, g]$ and $a_{i} \geqslant h \rightarrow a_{i}: \in[h, \infty)$.

The following two lemmas prove that the simple modification $A_{b d}$ can be used to decide reachability on $A$ by symbolic execution. The connection between $A_{b d}$ and $A$ is established via the automaton $A_{b d}^{\prime}$. The first lemma implies that $A_{b d}$ is timed bisimilar to $A_{b d}^{\prime}$.

Lemma 3.33. For all states $q$ and $r$ of the rectangular automaton $A_{b d}$, and every label $\tau \in \Sigma \cup \mathbb{R}_{\geqslant 0}$, we have $q \xrightarrow{\tau} A_{b d} r$ iff $\zeta(q) \xrightarrow{\tau} A_{A_{b d}^{\prime}} \zeta(r)$.

Proof. For $\tau \in \Sigma$, the statement of the lemma follows from Fact 3.31. For $\tau \in \mathbb{R}_{\geqslant 0}$, it suffices to prove the result for one-dimensional $A$. Suppose that $\left(v, x^{1}\right) \xrightarrow{\tau} A_{b d}\left(v, x^{2}\right)$. If $g<x^{1}, x^{2}<h$, then immediately $\zeta\left(v, x^{1}\right)=\left((v, o k), x^{1}\right)$ $\xrightarrow{\tau} A_{b d}^{\prime}\left((v, o k), x^{2}\right)=\zeta\left(v, x^{2}\right)$, The most interesting case is $x^{1}<g<h<x^{2}$. In this case, there exists a duration $t \leqslant \tau$ such that $(h-g) / t \in$ flow $_{A}(v)$. Therefore,

$$
\begin{aligned}
\zeta\left(v, x^{1}\right)=((v, o k), g) & \xrightarrow{\varepsilon} A_{b d}^{\prime}((v, l o w), g) \\
& \xrightarrow{\tau-t} A_{b d}^{\prime}((v, l o w), g) \\
& \xrightarrow{\varepsilon} A_{b d}^{\prime}((v, o k), g) \\
& \xrightarrow[A_{b d}^{\prime}]{ }((v, o k), h)=\zeta\left(v, x^{2}\right) .
\end{aligned}
$$

Similar arguments apply to all relative positions of $x^{1}, x^{2}, g$, and $h$. The reverse implication follows from the $\varepsilon$ edges of $A_{b d}$.

Let $Q_{o k} \subset Q_{A_{b d}^{\prime}}$ be the set of states of $A_{b d}^{\prime}$ that have the form $\left(\left(v, o k^{n}\right), \mathbf{x}\right)$. Then $\zeta$ is an onto function from $Q_{A_{b d}}$ to $Q_{A_{b d}^{\prime}}$. From Lemma 3.33, it follows that for every zone $Z$ of $A_{b d}$, Post $_{A_{b d}^{\prime}}(\zeta(Z)) \cap Q_{o k}=\zeta\left(\right.$ Post $\left._{A_{b d}}(Z)\right)$. Since $\zeta(q) \in$ Init $_{A_{b d}^{\prime}}$ if $q \in$ Init $_{A_{b d}}$ (by Fact 3.31), we conclude that $\operatorname{Reach}\left(A_{b d}^{\prime}\right) \cap Q_{o k}=\zeta\left(\operatorname{Reach}\left(A_{b d}\right)\right)$. Another consequence of Lemma 3.33 is the next lemma, which implies that the symbolic execution of $A_{b d}$ must terminate within at most one more step than the symbolic execution of $A_{b d}^{\prime}$ (whose bounded invariant regions guarantee termination).

Lemma 3.34. For all zones $Z$ of the rectangular automaton $A$, if Post $_{A_{b d}^{\prime}}(\zeta(Z))=\zeta(Z)$, then Post $_{A_{b d}}^{2}(Z)=$ Post $_{A_{b d}}(Z)$.

Proof. Consider a zone $Z$ of $A_{\tau}$ such that Post $_{A_{b d}^{\prime}}(\zeta(Z))$ $=\zeta(Z)$. We assume that $q \xrightarrow{\tau}_{A_{b d}} r$ for two states $q \in$ Post $_{A_{b d}}(Z)$ and $r \notin$ Post $_{A_{b d}}(Z)$, and show a contradiction. From Lemma 3.33 it follows that for all states $r^{\prime}$ with $\zeta\left(r^{\prime}\right)=\zeta(r)$, also $r^{\prime} \notin \operatorname{Post}_{A_{b d}}(Z)$. On the other hand, $\zeta(q) \in$ $\zeta\left(\right.$ Post $\left._{A_{b d}}(Z)\right) \subset$ Post $_{A_{b d}^{\prime}}(\zeta(Z))=\zeta(Z)$, using Lemma 3.33 a second time. Since $r \in$ Post $_{A_{b d}}(q)$, we have $\zeta(r) \in$ Post $_{A_{b d}^{\prime}}(\zeta(Z))$ $\cap Q_{o k}=\zeta\left(\right.$ Post $\left._{A_{b d}}(Z)\right)$ by a third and fourth application of Lemma 3.33. So there exists a state $r^{\prime} \in$ Post $_{A_{b d}}(Z)$ with $\zeta\left(r^{\prime}\right)=\zeta(r)$, which gives a contradiction.

From Lemmas 3.32 and 3.33 it follows that reachability in $A$ is equivalent to reachability in $A_{b d}$ : we have $\operatorname{Reach}(A) \cap Z_{f}$ $\neq \varnothing$ iff $\operatorname{Reach}\left(A_{b d}^{\prime}\right) \cap Z_{f}^{\prime} \neq \varnothing$ iff $\zeta\left(\operatorname{Reach}\left(A_{b d}\right)\right) \cap \zeta\left(Z_{f}\right) \neq \varnothing$ iff $\operatorname{Reach}\left(A_{b d}\right) \cap Z_{f} \neq \varnothing$ (the final equivalence follows from Fact 3.31, because all regions of $Z_{f}$ are $g h$-limited). From Proposition 3.30, Lemma 3.33, and Lemma 3.34 it follows that the symbolic execution of $A_{b d}$ terminates.

Theorem 3.35. Let $A$ be an initialized rectangular automaton, and let $Z_{f}$ be a rectangular zone of $A$. Then there exists a rectangular automaton $A_{b d}$, obtained from $A$ by adding $\varepsilon$ moves, such that (1) $\operatorname{Reach}(A) \cap Z_{f} \neq \varnothing$ iff $\operatorname{Reach}\left(A_{b d}\right) \cap Z_{f} \neq \varnothing$, and (2) there is a natural number $i \in \mathbb{N}$ with Reach $\left(A_{b d}\right)=$ Post $_{A_{b d}}^{i}\left(\right.$ Init $\left._{A_{b d}}\right)$.

## 4. UNDECIDABILITY

In Section 3, we showed that the 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, brings undecidability already with a single nonclock 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, which are of interest for the specification of duration properties [KPSY93].

An $n$-dimensional rectangular automaton $A$ is simple if it meets the following restrictions:

1. Exactly one variable of $A$ is not a clock.
2. The automaton $A$ has only one initial state $q_{0}$, and $q_{0}$ has the form $(v,(0,0, \ldots, 0))$.
3. For every edge $e$ of $A$, and every coordinate $i \in\{1, \ldots, n\}$, if $i \in j u m p(e)$ then $\operatorname{post}(e)_{i}=[0,0]$, and if $i \notin j u m p(e)$ then $\operatorname{post}(e)_{i}=\operatorname{pre}(e)_{i}$.
4. For every vertex $v$ of $A$, the invariant region $\operatorname{inv}(v)$ and the flow region flow $(v)$ are compact (by restriction (2), the initial region $\operatorname{init}(v)$ is compact as well). For every edge $e$ of $A$, the preguard region pre( $e$ ) is compact (by restriction (3), the postguard region post(e) is compact as well).
5. The observation alphabet of $A$ is a one-letter alphabet, and the invariant function inv of $A$ is constant. (One can require that $A$ has the fixed invariant function $\lambda v \cdot[0,1]^{n}$ or $\lambda v . \mathbb{R}^{n}$, with only minor modifications of our proofs.)
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. Restrictions (2) and (3) ensure that every simple automaton has deterministic jumps, which eliminates the nondeterminism of jumps as a possible source of undecidability. Many limited decidability results are based on a technique, called digitization, which discretizes time steps with noninteger durations [HMP92, BES93, BER94, PV94]. Since the digitization technique requires closed guard and invariant regions, restriction (4) implies that the technique does not generalize beyond very special cases. Many limited decidability results apply to automata with a single stopwatch [BES93, KPSY93, BER94, MV94, BR95, ACH97]. Restriction (1) implies that these results do not generalize either.

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 $M$ consists of a finite control and two unbounded nonnegative integer variables called counters. Initially both counter values are 0 . Three types of instructions are used: branching based upon whether a specific counter has the value 0 , incrementing a counter, and decrementing a counter
(which leaves unchanged a counter value of 0 ). When a specified halting location is reached, the machine halts. In our reductions, the finite control of $M$ is encoded in the finite control of a simple rectangular automaton $A_{M}$; in particular, there is a vertex $v$ such that that $M$ halts iff the zone $\{v\} \times \operatorname{inv}(v)$ is reachable for $A_{M}$. Each counter is encoded by a clock of $A_{M}$, and we supply widgets for performing the operations that correspond to incrementing or decrementing a counter. Typically, the counter value $u$ corresponds to the clock value $k_{1} \cdot\left(k_{2} / k_{1}\right)^{u}$, where $k_{1}$ and $k_{2}$ are the slopes of a two-slope variable of $A_{M}$, with $k_{1}$ being the larger. When $k_{1}=2 k_{2}$, 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 encode 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 a two-slope variable of slopes $k_{1}$ and $k_{2}$.

We first prove three lemmas that are basic to all of our undecidability proofs. Let $W$ be a positive rational number. A simple rectangular automaton $A$ is $W$-wrapping if its invariant function is defined as follows:

- For every variable $a$ of $A$ that is a clock, and every vertex $v$ of $A, \operatorname{inv}(v)(a)=[0, W]$.
- If $b$ is the nonclock variable of $A$, and $b$ takes only nonnegative slopes (i.e., flow $(v)(b) \subset[0, \infty)$ for all $v$ ), then for each vertex $v$ of $A, \operatorname{inv}(v)(b)=[0, W$. $\max \{\sup (f l o w(w)(b)) \mid w \in V\}]$.
- If $b$ is the nonclock variable of $A$, and $b$ takes only nonpositive slopes, then for each vertex $v$ of $A, \operatorname{inv}(v)(b)=$ $[W \cdot \min \{\inf (f l o w(w)(b)) \mid w \in V\}, 0]$.
- If $b$ is the nonclock variable of $A$, and $b$ takes both positive and negative slopes, then for each vertex $v \quad$ of $A, \quad \operatorname{inv}(v)(b)=[W \cdot \min \{\inf (f l o w(w)(b)) \mid w \in V\}$, $W \cdot \max \{\sup (f l o w(w)(b)) \mid w \in V\}]$.

A $W$-wrapping edge for a clock $a$ and a vertex $v$ is an edge from $v$ to itself that is annotated with the guarded command $a=W \rightarrow a:=0$. A $W$-wrapping edge for a nonclock variable $b$ and a vertex $v$ with $\operatorname{flow}(v)(b)=[k, k]$ is an edge from $v$ to itself that is annotated with the guarded command $b=k \cdot W \rightarrow b:=0$. The invariant of a wrapping automaton forces wrapping edges to be taken when 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 nonwrapping edge. This is the content of the wrapping lemma. A similar wrapping technique can be found in [Cer92].

In figures of simple automata, we use the following conventions. First, all variables whose slopes are not listed are clocks, i.e., they have slope 1. Second, wrapping conditions are left implicit; in particular, we omit invariants from every figure after those regarding the three basic lemmas, and we omit wrapping edges beginning with Fig. 20.

Wrapping Lemma. Let $W$ be a positive rational number. Let $k_{1} \in \mathbb{Q} \backslash\{0\}$, and consider the simple $W$-wrapping automaton fragment of Fig. 14. (Figure 14 assumes that $k_{1}>0$. If $k_{1}<0$, replace $c \leqslant k_{1} W$ by $c \geqslant k_{1} W$.) Suppose that the value of $c$ is $x$ when the edge $e_{1}$ is traversed, where $0<x<k_{1} \cdot W$ if $k_{1}>0$, and $k_{1} \cdot W<x<0$ if $k_{1}<0$. Then the next time $e_{3}$ is traversed, the value of $c$ is again $x$.

Proof. Figure 15 contains a time portrait that illustrates 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 traversed in the future, the following series of events must occur: (1) $e_{1}$ is traversed; (2) exactly $\left(W \cdot k_{1}-x\right) / k_{1}$ time units elapse, after which $c$ has the value $W \cdot k_{1}$, and $a$ has the value $\left(W \cdot k_{1}-x\right) / k_{1} ;(3)$ the wrapping edge $e_{2}$ is traversed, after which $c$ has the value 0 , and $a$ has the value $\left(W \cdot k_{1}-x\right) /$ $k_{1}$; (4) exactly $W-\left(W \cdot k_{1}-x\right) / k_{1}=x / k_{1}$ time units elapse, after which $a$ has the value $W$ and $c$ has the value $x$.

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

Equality Lemma. Let $W$ be a positive rational number. Consider the simple $W$-wrapping automaton fragment of Fig. 16, in which all variables are clocks. Suppose that the value of $c$ is $x$ and the value of $d$ is $y$ when the edge $e_{1}$ is traversed, where $0<x, y<W$. Then the edge $e_{3}$ can be traversed later iff $x=y$. Furthermore, the next time $e_{3}$ is traversed, the value of both $c$ and $d$ is $x$ (which is equal to $y$ ).


FIG. 14. Wrapping lemma: the skewed clock $c$ retains its entry value upon exit.


FIG. 15. Proof of the wrapping lemma for slope 1.

Similarly, by assigning the skewed clock $d$ to 0 at the same time as wrapping the skewed clock $c$ to 0 , we perform the assignment $d:=\left(k_{2} / k_{1}\right) \cdot c$, where $\dot{c}=k_{1}$ and $\dot{d}=k_{2}$.

Assignment Lemma. Let $W$ be a positive rational number. Let $k_{1}, k_{2} \in \mathbb{Q}$ with $k_{1} \neq 0$, and consider the simple $W$-wrapping automaton fragment of Fig. 17. ${ }^{3}$ Suppose that the value of $c$ is $x$ when the edge $e_{1}$ is traversed, where $0<x<k_{1} \cdot W$ if $k_{1}>0$, and $k_{1} \cdot W<x<0$ if $k_{1}<0$. Then the next time $e_{3}$ is traversed, the value of $c$ is again $x$ and the value of $d$ is $\left(k_{2} / k_{1}\right) \cdot x$.

Now we are ready to prove Theorem 4.1.
Proof of Theorem 4.1. We reduce the halting problem for two-counter machines to the reachability problem for simple wrapping automata with a two-slope variable taking slopes $k_{1}$ and $k_{2}$. Let $M$ be a two-counter machine with counters $C$ and $D$. We describe the construction of a simple wrapping automaton $A_{M}$ with the following six variables: $a$, $b, b^{\prime}, c$, and $d$ are clocks, and $z$ is a two-slope variable with slopes $k_{1}$ and $k_{2}$. The values of the counters $C$ and $D$ are encoded in the values of the clocks $c$ and $d$, respectively.

Case $k_{1}>k_{2}>0$ or $k_{1}<k_{2}<0$. The automaton $A_{M}$ is $W$-wrapping, where $W$ may be chosen to be any number larger than $\left|k_{1}\right|$. We encode counter value $u$ by clock value $\left|k_{1}\right| \cdot\left(k_{2} / k_{1}\right)^{u}$. The relationships $c=\left|k_{1}\right| \cdot\left(k_{2} / k_{1}\right)^{C}$ and $d=\left|k_{1}\right| \cdot\left(k_{2} / k_{1}\right)^{D}$ hold when $a=0$ or $a=W$, except when more than one time interval of duration $W$ is needed to simulate one computation step of $M$. The initialization of $C$ and $D$ is implemented by the initial vertex of $A_{M}$ and an outgoing edge $e$ with the preguard $\operatorname{pre}(e)(c)=\operatorname{pre}(e)(d)=$ $\left[\left|k_{1}\right|,\left|k_{1}\right|\right]$. The test $C=0$ is implemented by two edges $e_{1}$ and $e_{2}$, where $\operatorname{pre}\left(e_{1}\right)(c)=\left[\left|k_{1}\right|,\left|k_{1}\right|\right]$ (corresponding to $C=0$ ) and $\operatorname{pre}\left(e_{2}\right)(c)=\left[0,\left|k_{2}\right|\right]$ (corresponding to $C \neq 0$ ). Decrementing a counter corresponds to first checking if its value is 0 as above, and if not, then multiplying the corresponding clock value by $k_{1} / k_{2}$. This is implemented by concatenating two assignment-lemma constructions, as shown in Fig. 18 for counter $C$. In the first, $\dot{z}=k_{1}$; it performs $z:=k_{1} \cdot c$. In the second, $\dot{z}=k_{2}$; it performs

[^1]

FIG. 16. Equality lemma: testing if $c=d$.
$c:=\left(1 / k_{2}\right) \cdot z$. The bottom portion of Fig. 18 shows a time portrait of the decrementation fragment for $W=4, k_{1}=2$, and $k_{2}=1$. Notice that the value of $d$, which represents the unchanged counter $D$, is not affected by the decrementation fragment. Incrementing a counter corresponds to multiplying the corresponding clock value by $k_{2} / k_{1}$. This is done by reversing the two assignments, as shown in Fig. 19 for counter $C$. First $z:=k_{2} \cdot c$ is performed, and then $c:=\left(1 / k_{1}\right) \cdot z$. The bottom portion of Fig. 19 shows a time portrait of the incrementation fragment for $W=4, k_{1}=2$, and $k_{2}=1$.

Each instruction of $M$ can be implemented as outlined above, with the terminal edge of the widget for instruction $i$ coinciding with the initial edge of the widget for instruction $i+1$ (with the obvious modifications for branch instructions). It follows that the vertex of $A_{M}$ that corresponds to the halting location of $M$ is reachable iff $M$ halts.

Case $k_{1} \neq 0$ and $k_{2}=0$. The construction is insensitive to the sign of $k_{1}$. We use the wrapping constant $W=4$ and encode counter value $u$ by clock value $2^{1-u}$. Initialization and test for zero are implemented easily. Decrementing a counter corresponds to doubling the corresponding clock. The doubling procedure is shown in Fig. 20 for the clock $c$; in this and all remaining figures, we omit the wrapping edges required to maintain the value of $d$. The idea is to perform $z:=k_{1} \cdot c$ 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 $W$, we have $z=2 k_{1} \cdot x$, where $x$ is the original value of $c$. Finally, we perform $c:=\left(1 / k_{1}\right) \cdot z$ with the assignment lemma. The bottom portion of Fig. 20 shows a time portrait for $k_{1}=2$. Incrementing a counter corresponds to halving the corresponding clock, which can be


FIG. 17. Assignment lemma: performing the assignment $d:=\left(k_{2} / k_{1}\right) \cdot c$.


FIG. 18. Counter decrement: multiplying $c$ by $k_{1} / k_{2}$ using the twoslope variable $z$.
done with two auxiliary clocks $b$ and $b^{\prime}$. To halve the value of $c$, first a value is nondeterministically guessed in $b$. Then $b^{\prime}:=2 b$ is performed using the above doubling procedure. Then $c=b^{\prime}$ is checked by the equality lemma, and if this succeeds, then $c:=b$ is performed using the assignment lemma.

Case $k_{1}>0>k_{2}$. First suppose that $\left|k_{2}\right|<k_{1}$. The wrapping constant $W$ can be any number larger than $k_{1}$. We encode counter value $u$ by clock value $k_{1} \cdot\left(\left|k_{2}\right| / k_{1}\right)^{u}$. Now we need two synchronization clocks, $a$ and $b$. Clock $c$ is synchronized with $a$, and clock $d$ is synchronized with $b$.



FIG. 19 Counter increment: multiplying $c$ by $k_{2} / k_{1}$ using the twoslope variable $z$.

The relationship $c=k_{1} \cdot\left(\left|k_{2}\right| / k_{1}\right)^{c}$ holds when $a=0$ or $a=W$, and the relationship $d=k_{1} \cdot\left(\left|k_{2}\right| / k_{1}\right)^{D}$ holds when $b=0$ or $b=W$. To multiply $c$ by $k_{1} /\left|k_{2}\right|$, we first perform $z:=k_{1} \cdot c$ 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 $c=\left(k_{1} /\left|k_{2}\right|\right) \cdot x$, where $x$ is the original value of $c$. See Fig. 21. The bottom portion of the figure shows a time portrait for $W=4, k_{1}=2$, and $k_{2}=-1$. To multiply $c$ by $\left|k_{2}\right| / k_{1}$, simply reverse the slopes of $z$. Specifically, perform $z:=k_{2} \cdot c$, reset $c$ to 0 , then put $\dot{z}=k_{1}$, and when $z$ reaches 0 , reset $a$ to 0 .

If $\left|k_{2}\right|>k_{1}$, we use a wrapping constant larger than $\left|k_{2}\right|$ and encode counter value $u$ by clock value $\left|k_{2}\right| \cdot\left(k_{1} /\left|k_{2}\right|\right)^{u}$. This simply switches the roles of multiplying by $k_{1} /\left|k_{2}\right|$ and multiplying by $\left|k_{2}\right| / k_{1}$.

Finally, suppose that $k_{2}=-k_{1}$. In this case we use the wrapping constant 4 and encode counter value $u$ by clock value clock value $2^{1-u}$. Again we use separate synchronization clocks $a$ and $b$ for $c$ and $d$. To double $c$, perform $z:=k_{1} \cdot c$, and then put $\dot{z}=k_{2}=-k_{1}$, resetting $a$ when $z$ reaches 0 . See Fig. 22, 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 when $c$ reaches 4 , keeping $z$ at slope $k_{1}$ for the first half of the interval, and at 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 Fig. 23, which gives the construction, and a time portrait for $k_{1}=5$.

### 4.2. Generalized Automata

A slight generalization of the invariant, flow, preguard, postguard, or jump function leads to the undecidability of rectangular automata, even under the stringent restrictions of simplicity and initialization. For the remainder of this section, fix an $n$-dimensional rectangular automaton $A$.

## Rectangular Automata with Assignments

The jump function of $A$ can be generalized to allow, during edge steps, the value of one variable to be assigned to another variable. A jump function with assignments for $A$ assigns to each edge $e$ of $A$ both a jump set $j u m p(e) \subset\{1, \ldots, n\}$ and an assignment function $\operatorname{assign}(e):\{1, \ldots, n\} \rightarrow\{1, \ldots, n\}$. The edge-step relation $\xrightarrow{\sigma}$, for $\sigma \in \Sigma$, is then redefined as follows: $(v, \mathbf{x}) \xrightarrow{\sigma}(w, \mathbf{y})$ iff there is an edge $e=(v, w)$ of $A$ such that $\operatorname{obs}(e)=\sigma$, and $\mathbf{x} \in \operatorname{pre}(v)$, and $\mathbf{y} \in \operatorname{post}(w)$, and for all $i \in\{1, \ldots, n\}$ with $i \notin j \operatorname{jump}(e)$, we have $y_{i}=x_{\text {assign }(i)}$. A rectangular automaton with assignments is a rectangular automaton whose jump function is replaced by a jump function with assignments.

Using a jump function with assignments, the proof of Theorem 4.1 can be replicated even if the two-slope variable is replaced by a memory cell (with slope 0 ) or by a skewed clock (with any slope different from 0 and 1). The former



FIG. 20. Doubling $c$ using the two-slope variable $z$ with the slopes 0 and $k_{1}$.


FIG. 21. Multiplying $c$ by $k_{1} /\left|k_{2}\right|$ when $k_{1}>0>k_{2}$ and $\left|k_{2}\right|<k_{1}$.



FIG. 22. Doubling $c$ when $k_{2}=-k_{1}$.



FIG. 23. Halving $c$ when $k_{2}=-k_{1}$.
gives a new proof of a result from [Cer92]. In the following theorem, notice that every simple rectangular automaton whose nonclock variable is a one-slope variable is necessarily initialized.

Theorem 4.2. For every slope $k \in \mathbb{Q} \backslash\{1\}$, the reachability problem is undecidable for simple rectangular automata with assignments and a one-slope variable of slope $k$.

Proof. Consider $k \in \mathbb{Q} \backslash\{1\}$. We repeat the construction given in the proof of Theorem 4.1 for the case $k_{1}=1$ and $k_{2}=k$ with the following modifications. The two-slope variable $z$ is replaced by a clock $z_{1}$ and a one-slope variable $z_{2}$ with slope $k$. Each vertex of $A_{M}$ is augmented with a bit $s l p \in\{1,2\}$ that indicates if the value of $z$ corresponds to the current value of $z_{1}$ or to the current value of $z_{2}$. Assigments are used to copy the value of $z_{1}$ into $z_{2}$, or vice versa, whenever the bit $s l p$ changes. More precisely, for each edge $e=(v, w)$ of $A_{M}$, we have an edge $e^{\prime}$ from $(v, s l p)$ to $\left(w, s l p^{\prime}\right)$, where $s l p=i$ iff the slope of $z$ in $v$ is $k_{i}$, and $s l p^{\prime}=i$ iff the slope of $z$ in $w$ is $k_{i}$. For all coordinates other than $z, z_{1}$, and $z_{2}$, the preguard and postguard intervals and the jump sets of $e$ and $e^{\prime}$ coincide. In addition, $\operatorname{pre}\left(e^{\prime}\right)\left(z_{s l p}\right)=\operatorname{pre}(e)(z)$, $\operatorname{post}\left(e^{\prime}\right)\left(z_{s l p^{\prime}}\right)=\operatorname{post}(e)(z)$, and $z_{s l p^{\prime}}$ is in the jump set of $e^{\prime}$ iff $z$ is in the jump set of $e$. Finally, the assignment function of $e^{\prime}$ assigns to $z_{s l p^{\prime}}$ the value of $z_{s l p}$.

## Triangular Preguard, Postguard, and Invariant Constraints

The preguard, postguard, and invariant functions of $A$ can be generalized to allow comparisons between the values of variables. We call preguard, postguard, and invariant constraints of the form $a \leqslant b$, where $a$ and $b$ are variables, triangular, because they define triangular regions of $\mathbb{R}^{n}$. A triangular restriction $\leqslant$ for $A$ is a reflexive and transitive binary relation on $\{1, \ldots, n\}$. A triangular preguard (resp. postguard) function for $A$ assigns to each edge $e$ of $A$ both a rectangular region $\operatorname{pre}(e)$ (resp. post $(e)$ ) and a triangular restriction $\leqslant_{e}$. The edge-step relation $\xrightarrow{\sigma}$, for $\sigma \in \Sigma$, is then redefined as follows: $(v, \mathbf{x}) \xrightarrow{\sigma}(w, \mathbf{y})$ iff there is an edge $e=(v, w)$ of $A$ such that $o b s(e)=\sigma, \mathbf{x} \in \operatorname{pre}(v), \mathbf{y} \in \operatorname{post}(w)$, for all $i \in\{1, \ldots, n\}$ with $i \notin j u m p(e)$, we have $x_{i}=y_{i}$, and for
all $i, j \in\{1, \ldots, n\}$ with $i \leqslant{ }_{e} j$, we have $x_{i} \leqslant x_{j}$ (resp. $y_{i} \leqslant y_{j}$ ). A triangular invariant function for $A$ assigns to each vertex $v$ of $A$ both a rectangular region $\operatorname{inv}(v)$ and a triangular restriction $\leqslant_{v}$. The state space $Q$ of $A$ is then redefined to contain a pair $(v, \mathbf{x}) \in V \times \mathbb{R}^{n}$ iff $\mathbf{x} \in \operatorname{inv}(v)$ and for all $i, j \in\{1, \ldots, n\}$ with $i \leqslant_{v} j$, we have $x_{i} \leqslant x_{j}$. An automaton with triangular preguards (resp. postguards; invariants) is a rectangular automaton whose preguard (resp. postguard; invariant) function is replaced by a triangular preguard (resp. postguard; invariant) function.

Using any of the three types of triangular constraints, the proof of Theorem 4.2 can be replicated without assignments.

Theorem 4.3. For every slope $k \in \mathbb{Q} \backslash\{1\}$, the reachability problem is undecidable for simple automata with triangular preguards or postguards or invariants, and a one-slope variable of slope $k$.

Proof. Triangular preguards, postguards, or invariants permit comparisons between variables of the form $a=b$. This allows an assignment $a:=b$ to be simulated by a reset $a:=0$ at a nondeterministically chosen point in time, followed later by the test $a=b$ (assuming positive values and slopes). As usual, wrapping ensures that all other variables, as well as $b$, maintain their values while the assignment is simulated. In this way, the construction outlined in the proof of Theorem 4.2 can be modified appropriately.

## Triangular Flow Constraints

The flow functions of $A$ can be generalized to impose an ordering on the first derivatives of variables. For example, the triangular flow constraint $1 \leqslant \dot{a} \leqslant \dot{b} \leqslant 2$ says that both $a$ and $b$ may increase at any slopes between 1 and 2, but $b$ increases always as least as fast as $a$. A triangular flow function for $A$ assigns to each vertex $v$ of $A$ both a rectangular region flow $(v)$ and a triangular restriction $\leqslant_{v}$. For $t>0$, the time-step relation $\xrightarrow{[t]}$ is then redefined as follows: $(v, \mathbf{x}) \xrightarrow{[t]}(w, \mathbf{y})$ iff $v=w$, and $(\mathbf{y}-\mathbf{x}) / t \in$ flow $(v)$, and for all $i, j \in\{1, \ldots, n\}$ with $i \leqslant_{v} j$, we have $y_{i}-x_{i} \leqslant y_{j}-x_{j}$. The triangular flow function is called constant if the functions
flow and $\lambda v . \leqslant_{v}$ are both constant functions on the set of vertices. An automaton with constant triangular flow is a rectangular automaton whose flow function is replaced by a constant triangular flow function.

Using a constant triangular flow constraint, the proof of Theorem 4.1 can be replicated. This is nontrivial, because Theorem 4.1 permits a two-slope variable that is not governed by a constant flow function. For the simulation of the twoslope variable, we use three infinite-slope variables that do follow a constant flow function, albeit a triangular one.

Theorem 4.4. The reachability problem is undecidable for 3 -simple automata with constant triangular flow.

Proof. We use three nonclock variables $z, z_{1}$, and $z_{2}$ with the constant triangular flow constraint $1 \leqslant \dot{z}_{1} \leqslant z \leqslant$ $\dot{z}_{2} \leqslant 2$. The idea is to repeat the construction given in the proof of Theorem 4.1 for the case $k_{1}=1$ and $k_{2}=2$ with two additional variables, $z_{1}$ and $z_{2}$, which enforce that the slope of $z$ is always either 1 or 2 . The slope 1 of $z$ is enforced by resetting $z_{2}$ to 0 whenever $a$ wraps to 0 , and later checking that $a=4 \wedge z_{2}=4$. Similarly, the slope 2 of $z$ is enforced by resetting $z_{1}$ to 0 whenever $a$ wraps to 0 , and later checking that $a=4 \wedge z_{1}=8$.

## 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 and $\omega$-language emptiness problems for timed automata are decidable. Linear hybrid automata [AHH96] equip finitestate 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 НуТесн [HНWT97]. 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. On one hand, initialized rectangular automata generalize timed automata without incurring a complexity penalty. Their reachability problem is PSPACEcomplete and, under the natural restriction of bounded nondeterminism, so is their $\omega$-language emptiness problem. (We do not know the complexity of the $\omega$-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 nonrectangular systems have undecidable reachability problems.

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

Initialized rectangular automata are also interesting from a practical perspective. First, reachability analysis using НyТесн terminates on every initialized rectangular automaton with bounded invariants, and on every initialized rectangular automaton after a linear-time preprocessing step. Second, many distributed communication protocols assume that local clocks have bounded drift. Such protocols are naturally modeled as initialized rectangular hybrid automata. For example, HyTech has been applied successfully to verify one such protocol used in Philips audio components [HW95]. Third, initialized rectangular automata can be used to conservatively approximate, arbitrarily closely, hybrid systems with general dynamical laws [OSY94, PBV96, HHWT98].

## ACKNOWLEDGMENT

We thank Howard Wong-Toi for a careful reading and for $A_{b d}$.

## REFERENCES

[ $\left.\mathrm{ACH}^{+} 95\right]$ 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, Theor. Comput. Sci. 138 (1995), 3-34.
[ACH97] R. Alur, C. Courcoubetis, and T. A. Henzinger, Computing accumulated delays in real-time systems, Formal Methods System Design 11, No. 2 (1997), 137-156.
[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 "Hybrid Systems" (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds.), Lecture Notes in Computer Science, Vol. 736, pp. 209-229, Springer-Verlag, New York/Berlin, 1993.
[AD94] R. Alur and D. L. Dill, A theory of timed automata, Theor. Comput. Sci. 126 (1994), 183-235.
[AHH96] R. Alur, T. A. Henzinger, and P.-H. Ho, Automatic symbolic verification of embedded systems, IEEE Trans. Software Eng. 22, No. 3 (1996), 181-201.
[AHV93] R. Alur, T. A. Henzinger, and M. Y. Vardi, Parametric realtime reasoning, in "Proceedings of the 25th Annual Symposium on Theory of Computing," pp. 592-601, ACM Press, New York, 1993.
[BER94] A. Bouajjani, R. Echahed, and R. Robbana, Verifying invariance properties of timed systems with duration variables, in "FTRTFT 94: Formal Techniques in Real-time and Fault-tolerant Systems" (H. Langmaack, W.-P. de Roever, and J. Vytopil, Eds.), Lecture Notes in Computer Science, Vol. 863, pp. 193-210, Springer-Verlag, New York/ Berlin, 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," pp. 147-159, IEEE Comput. Soc. Press, New York, 1993.
[BR95] A. Bouajjani and R. Robbana, Verifying $\omega$-regular properties for subclasses of linear hybrid systems, in "CAV 95:

Computer-aided Verification" (P. Wolper, Ed.), Lecture Notes in Computer Science, Vol. 939, pp. 437-450, SpringerVerlag, New York/Berlin, 1995.
[Cer92] K. Čerāns, "Algorithmic Problems in Analysis of Real-time System Specifications," Ph.D. thesis, University of Latvia, 1992.
[Cor96] J. C. Corbett, Timing analysis of Ada tasking programs, IEEE Trans. Software Eng. 22, No. 7 (1996), 461-483.
[HHWT95] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, HyTech: The next generation, in "Proceedings of the 16th Annual Realtime Systems Symposium," pp. 56-65, IEEE Comput. Soc. Press, New York, 1995.
[HHWT97] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, HyTech: A model checker for hybrid systems, Software Tools Technology Transfer 1 (1997), 110-122.
[HHWT98] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, Algorithmic analysis of nonlinear hybrid systems, IEEE Trans. Automatic Control 43, No. 4 (1998), 540-554.
[HMP92] T. A. Henzinger, Z. Manna, and A. Pnueli, What good are digital clocks?, in "ICALP 92: Automata, Languages, and Programming" (W. Kuich, Ed.), Lecture Notes in Computer Science, Vol. 623, pp. 545-558, Springer-Verlag, New York/ Berlin, 1992.
[HNSY94] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, Symbolic model checking for real-time systems, Inform. \& Comput. 111, No. 2 (1994), 193-244.
[HRP94] N. Halbwachs, P. Raymond, and Y.-E. Proy, Verification of linear hybrid systems by means of convex approximation, in "SAS 94: Static Analysis Symposium" (B. LeCharlier, Ed.), Lecture Notes in Computer Science, Vol. 864, pp. 223-237, Springer-Verlag, New York/Berlin, 1994.
[HW95] P.-H. Ho and H. Wong-Toi, Automated analysis of an audio control protocol, in "CAV 95: Computer-aided Verification" (P. Wolper, Ed.), Lecture Notes in Computer Science, Vol. 939, pp. 381-394, Springer-Verlag, New York/Berlin, 1995.
[HWT96] T. A. Henzinger and H. Wong-Toi, Using HyTech to synthesize control parameters for a steam boiler, in "Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control" (J.-R. Abrial, E. Börger, and H. Langmaack, Eds.), Lecture Notes in Computer Science, Vol. 1165, pp. 265-282, Springer-Verlag, New York/Berlin, 1996.
[Kop96] P. W. Kopke, "The Theory of Rectangular Hybrid Automata," Ph.D. thesis, Cornell University, 1996.
[KPSY93] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine, Integration graphs: A class of decidable hybrid systems, in "Hybrid

Systems" (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds.), Lecture Notes in Computer Science, Vol. 736, pp. 179-208, Springer-Verlag, New York/Berlin, 1993.
[LV96] N. A. Lynch and F. Vaandrager, Forward and backward simulations, Part II: Timing-based systems, Inform. \& Comput. 128, No. 1 (1996), 1-25.
[MV94] J. McManis and P. Varaiya, Suspension automata: A decidable class of hybrid automata, in "CAV 94: Computeraided Verification" (D. L. Dill, Ed.), Lecture Notes in Computer Science, Vol. 818, pp. 105-117, Springer-Verlag, New York/Berlin, 1994.
[NOSY93] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, An approach to the description and analysis of hybrid systems, in "Hybrid Systems" (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds.), Lecture Notes in Computer Science, Vol. 736, pp. 149-178, Springer-Verlag, New York/Berlin, 1993.
[NS95] S. Nadjm-Tehrani and J.-E. Strömberg, Proving dynamic properties in an aerospace application, in "Proceedings of the 16th Annual Real-time Systems Symposium," pp. 2-10, IEEE Comput. Soc. Press, New York, 1995.
[OSY94] A. Olivero, J. Sifakis, and S. Yovine, Using abstractions for the verification of linear hybrid systems, in "CAV 94: Computer-aided Verification" (D. L. Dill, Ed.), Lecture Notes in Computer Science, Vol. 818, pp. 81-94, SpringerVerlag, New York/Berlin, 1994.
[PBV96] A. Puri, V. Borkar, and P. Varaiya, $\varepsilon$-Approximation of differential inclusions, in "Hybrid Systems III" (R. Alur, T. A. Henzinger, and E. D. Sontag, Eds.), Lecture Notes in Computer Science, Vol. 1066, pp. 362-376, Springer-Verlag, New York/Berlin, 1996.
[PV94] A. Puri and P. Varaiya, Decidability of hybrid systems with rectangular differential inclusions, in "CAV 94: Computeraided Verification" (D. L. Dill, Ed.), Lecture Notes in Computer Science, Vol. 818, pp. 95-104, Springer-Verlag, New York/Berlin, 1994.
[SMF97] T. Stauner, O. Müller, and M. Fuchs, Using HyTech to verify an automotive control system, in "HART 97: Hybrid and Real-time Systems" (O. Maler, Ed.), Lecture Notes in Computer Science, Vol. 1201, pp. 139-153, Springer-Verlag, New York/Berlin, 1997.
[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," pp. 322-331, IEEE Comput. Soc. Press, New York, 1986.


[^0]:    ${ }^{1}$ A preliminary version of this paper appeared in the Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC 1995), pp. 373-382. 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 Grant CCR-9504469, by the Air Force Office of Scientific Research Contract F49620-93-1-0056, by the Army Research Office MURI Grant DAAH-04-96-1-0341, by the Army Research Office Contract DAAH-04-94-G-0026, by the Defense Advanced Research Projects Agency Grant NAG2-892, and by the California PATH program.
    ${ }^{2}$ Corresponding author.

[^1]:    ${ }^{3}$ Figure 17 assumes that $k_{1}, k_{2} \geqslant 0$. If $k_{1}<0$, replace $c \leqslant k_{1} \cdot W$ by $c \geqslant k_{1} \cdot W$; if $k_{2}<0$, replace $d \leqslant k_{2} \cdot W$ by $d \geqslant k_{2} \cdot W$.

