ELSEVIER Theoretical Computer Science 221 (1999) 369–392

Theoretical Computer Science

www.elsevier.nl/locate/tcs

Discrete-time control for rectangular hybrid automata☆

Thomas A. Henzinger[a,*], Peter W. Kopke[b]

[a] *Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, USA*

[b] *William H. Kopke, Jr. Inc., Lake Success, NY 11042, USA*

Abstract

Rectangular hybrid automata model digital control programs of analog plant environments. We study rectangular hybrid automata where the plant state evolves continuously in real-numbered time, and the controller samples the plant state and changes the control state discretely, only at the integer points in time. We prove that rectangular hybrid automata have finite bisimilarity quotients when all control transitions happen at integer times, even if the constraints on the derivatives of the variables vary between control states. This is in contrast with the conventional model where control transitions may happen at any real time, and already the reachability problem is undecidable. Based on the finite bisimilarity quotients, we give an exponential algorithm for the symbolic sampling-controller synthesis of rectangular automata. We show our algorithm to be optimal by proving the problem to be EXPTIME-hard. We also show that rectangular automata form a maximal class of systems for which the sampling-controller synthesis problem can be solved algorithmically. © 1999 Published by Elsevier Science B.V. All rights reserved.

Keywords: Hybrid systems; Computer-aided verification; Controller synthesis

1. Introduction

Hybrid systems are dynamical systems with both discrete and continuous components. A paradigmatic example of a hybrid system is a digital control program for an analog plant environment, like a furnace or an airplane: the controller state moves discretely between control modes, and in each control mode, the plant state evolves continuously according to physical laws. A natural mathematical model for hybrid

* Corresponding author.

E-mail address: tah@eecs.berkeley.edu. (Thomas A. Henzinger), pkpk@kopkefruit.com. (Peter W. Kopke)

☆ This research was supported in part by the ONR YIP award N00014-95-1-0520, by the NSF CAREER award CCR-9501708, by the NSF grant CCR-9504469, by the AFOSR contract F49620-93-1-0056, by the ARO MURI contract DAAH-04-96-1-0341, by the ARO contract DAAL03-91-C-0027 through the MSI at Cornell University, by the DARPA/NASA grant NAG2-1214, and by the SRC contract 97-DC-324.041.

systems is the *hybrid automaton*, which represents discrete components using finite-state machines and continuous components using real-numbered variables [1]. A particularly important subclass of hybrid automata are the *rectangular automata*, where in each control mode v, the given n variables follow a nondeterministic differential equation of the form $\mathrm{d}x/\mathrm{d}t \in B(v)$, for an n-dimensional rectangle $B(v) \subset \mathbb{R}^n$ [14]. Rectangular automata are useful for two reasons. First, they can be made to approximate, arbitrarily closely, complex continuous behavior using lower and upper bounds on derivatives [11, 22]. Second, they can be executed and reverse executed symbolically using polyhedral constraints to represent the possible values of variables [4, 12].

Symbolic execution stands in contrast to explicit execution, where each new state is computed individually. For systems that are reverse executable symbolically (r.e.s.), verification and control yield to a (semi)algorithmic approach even if the state space is infinite [10]. For r.e.s. systems, many model checking and controller synthesis problems can be solved by computing, using iterative approximation, certain fixpoints of predecessor operators on state sets [6, 21]. Various fixpoint computations are guaranteed to terminate in the presence of suitable finite quotient spaces. For example, r.e.s. systems with finite bisimilarity quotients allow LTL and CTL model checking, and safety controller synthesis. While rectangular automata are r.e.s., they do not necessarily have finite (time-abstract) bisimilarity quotients [9]; indeed, simple reachability questions are undecidable for rectangular automata [14]. A noted subclass of rectangular automata with finite bisimilarity quotients are the *timed automata*, where all variables are clocks with derivative 1 [2]. As a consequence, symbolic algorithms for model checking and controller synthesis are known for timed automata [15, 21].

While previous results on timed and hybrid automata allow edge transitions (i.e., control mode switches) to occur at any real-numbered points in time, this is not necessarily a natural assumption for controller synthesis, as it permits controllers that, in a single time unit, can interact with the plant an unbounded number of times (even infinitely often, if no special care is taken [3]). By contrast, we study the safety control problem under the assumption that while the plant evolves continuously, the controller samples the plant state discretely, at the integer points in time only.[2] This leads to the following formulation of the *sampling-controller synthesis problem* for rectangular automata: given a continuous-time rectangular automaton, is there a discrete-time controller that samples the automaton state at integer times and switches the control mode accordingly so that the resulting closed-loop system satisfies a given invariant?

To solve this problem, we study the *discrete-time transition systems* of rectangular automata, where all time transitions have unit duration. It should be noticed that all variables still evolve continuously, in real-numbered time; only edge transitions are restricted to discrete time. We prove that unlike in the case of dense time, the discrete-

[2] The sampling rate of the controller may be any rational, but without loss of generality we assume it to be 1.

time transition system of every rectangular automaton has a finite bisimilarity quotient.[3] As a corollary, we conclude that the standard approaches to symbolic model checking and controller synthesis are guaranteed to terminate when all control switches must occur at integer times. The running times of the verification and control algorithms depend on the number of bisimilarity equivalence classes, which, while exponential in the description of the automaton, is less by a multiplicative exponential factor than the number of region equivalence classes used for the dense-time verification and control of timed automata. Thus, the often more realistic sampling-controller synthesis problem can be solved for a wider class of hybrid systems than dense-time control (rectangular vs. timed), at a smaller cost.

We prove that our exponential sampling-control algorithm is optimal, by giving lower bounds on the control problem for timed and hybrid systems: we show that the safety control decision problem (does there exist a controller that maintains an invariant?) is complete for EXPTIME already in the restricted case of discrete-time timed automata. We also identify the boundary of sampling controllability by proving that several generalizations of rectangular automata lead to an undecidable reachability problem, even in discrete time. The undecidability of dense-time reachability for rectangular automata has led [23] to consider the restriction that the flow rectangle $B(v)$ must be the same for each control mode v. For the resulting class of *initialized* rectangular automata, reachability is decidable [14]. Our work can be viewed as pointing out an orthogonal restriction of rectangularity, namely, that the flow rectangle may change only at integer points in time. Unlike initialization, our restriction guarantees not only a finite language equivalence quotient but a finite bisimilarity quotient on the infinite state space of a rectangular automaton.

2. Prerequisite discussion of transition systems

We begin by presenting some general results about (labeled) transition systems, which will be applied later to the transition systems defined by rectangular automata.

2.1. Definition of transition systems

Definition 2.1 (*Transition system*). A *transition system* $S=(Q,\Sigma,\rightarrow,Q_\mathrm{I},\Pi,\models)$ consists of a set Q of *states*, a finite set Σ of *events*, a multiset $\rightarrow\ \subset Q\times\Sigma\times Q$ called the *transition relation*, a set $Q_\mathrm{I}\subset Q$ of *initial states*, a set Π of *observations*, and a *satisfaction relation* $\models\ \subset Q\times\Pi$. We write $q\xrightarrow{\sigma}r$ instead of $(q,\sigma,r)\in\ \rightarrow$, and $q\models\pi$ instead of $(q,\pi)\in\ \models$. The transition system S is *finite* if Q is finite. We say that the event σ is *enabled* in the state q if $q\xrightarrow{\sigma}r$ for some state r. We assume for simplicity that S is deadlock-free; that is, for each state $q\in Q$, there exists an event $\sigma\in\Sigma$ that

[3] Under the technical restriction that each variable is either nonnegative and nondecreasing, or bounded from below and above.

is enabled in q. A *region* is a subset of Q. Given an observation $\pi \in \Pi$, we write $R_\pi = \{q \in Q \mid q \models \pi\}$ for the region of states that satisfy π.

2.2. Verification as reachability

Definition 2.2 (*Weakest precondition*). Let S be a transition system. For each event $\sigma \in \Sigma$, the *σ-predecessor operator* $Pre_\sigma : 2^Q \to 2^Q$ is defined by $Pre_\sigma(R) = \{q \in Q \mid \exists r \in R. q \xrightarrow{\sigma} r\}$. In particular, $Pre_\sigma(Q)$ is the set of states in which the event σ is enabled. The *weakest-predecessor operator* $Pre : 2^Q \to 2^Q$ is defined by $Pre(R) = \bigcup_{\sigma\in\Sigma} Pre_\sigma(R)$. A region $R \subset Q$ is *reachable* in S if $Q_I \cap Pre^i(R) \neq \emptyset$ for some $i \in \mathbb{N}$; otherwise, R is *unreachable*.

The basic verification problem for transition systems asks whether an unsafe state is unreachable.

Definition 2.3 (*Safety verification*). Let $\mathscr{C}$ be a class of transition systems. The *safety verification problem* for $\mathscr{C}$ is stated in the following way: given a transition system $S \in \mathscr{C}$ and an observation $\pi \in \Pi$, determine whether the region R_π is unreachable in S.

For finite transition systems, the safety verification problem is the complement of graph reachability; it thus can be solved in linear time and is complete for NLOGSPACE. The safety verification problem can be generalized to the safety control problem.

2.3. Control as alternating reachability

We use the following model for control: for each state q of a transition system, a (memory-free) controller chooses an enabled event σ so that in state q, the controlled system always proceeds via event σ. Since q may have several σ-successors, the controlled system may still be nondeterministic. Alternative models for memory-free control are equivalent.

Definition 2.4 (*Control map*). Let S be a transition system. A *control map* for S is a function $\kappa : Q \to \Sigma$ such that for each state $q \in Q$, there exists a state $r \in Q$ with $q \xrightarrow{\kappa(q)} r$. The *closed-loop system* $\kappa(S)$ is the transition system $(Q, \Sigma, \Rightarrow, Q_I, \Pi, \models)$, where $q \overset{\sigma}{\Rightarrow} r$ iff $q \xrightarrow{\sigma} r$ and $\kappa(q) = \sigma$.

The basic control problem for transition systems asks whether an unsafe state is avoidable by applying some control map.

Definition 2.5 (*Safety control*). Let $\mathscr{C}$ be a class of transition systems. The *safety control decision problem* for $\mathscr{C}$ is stated in the following way: given a transition system $S \in \mathscr{C}$ and an observation $\pi \in \Pi$, determine whether there exists a control map κ such that the region R_π is unreachable in the closed-loop system $\kappa(S)$. If so, then we say that π is *avoidable* in S; otherwise, π is *unavoidable*. The *safety controller synthesis problem* requires the construction of a witnessing control map κ when π is avoidable.

Notice that the safety verification problem is the special case of the safety control decision problem where $|\Sigma| = 1$. Like safety verification, also safety control can be solved by iterating a predecessor operator on regions. For this purpose, the weakest-predecessor operator is replaced by the (more general) uncontrollable-predecessor operator [21].

Definition 2.6 (*Uncontrollable precondition*). Let S be a transition system. The *uncontrollable-predecessor operator* $UPre : 2^Q \to 2^Q$ is defined by

$$UPre(R) = \bigcap_{\sigma \in \Sigma} (Pre_\sigma(R) \cup (Q \backslash Pre_\sigma(Q))).$$

That is, for each region $R \subset Q$, the region $UPre(R)$ is the set of states that no control map can keep out of R for even one transition.

Then, the observation $\pi \in \Pi$ is unavoidable in the transition system S iff $Q_I \cap UPre^i(R_\pi) \neq \emptyset$ for some $i \in \mathbb{N}$. We call $U_\pi = \bigcup_{i \in \mathbb{N}} UPre^i(R_\pi)$ the *uncontrollable region* for π, because it contains all states that, in the long run, no control map can keep out of R_π. So π is avoidable iff U_π contains no initial state. For finite transition systems, $U_\pi = \bigcup_{0 \leqslant i \leqslant |Q|} UPre^i(R_\pi)$ can be computed in linear time [5]. If π is avoidable, then a witnessing control map can be constructed by choosing for each state $q \in Q \backslash U_\pi$ an event σ such that $q \in Pre_\sigma(Q) \backslash Pre_\sigma(U_\pi)$. This gives us the following upper bound on safety controller synthesis.

Theorem 2.1 (Thatcher and Wright [25], Beeri [5], Ramadge and Wonham [24]). *The safety controller synthesis problem for finite transition systems can be solved in linear time.*

To determine a lower bound, we reduce AND–OR graph reachability to the safety control decision problem.

Definition 2.7 (*Alternating reachability*). An *AND–OR graph* $G = (V_A, V_O, V_I, \Rightarrow)$ consists of a finite set $V = V_A \cup V_O$ of vertices that is partitioned into a set V_A of AND vertices and a set V_O of OR vertices, a set $V_I \subset V$ of initial vertices, and a multiset $\Rightarrow\ \subset V \times V$ of edges. We assume deadlock freedom, namely, that for each vertex $v \in V$, there exists a vertex $w \in V$ such that $v \Rightarrow w$. The *alternating-predecessor operator* $APre : 2^V \to 2^V$ is defined by

$$APre(R) = \{q \in V_O \mid (\exists r \in R)(q \Rightarrow r)\} \cup \{q \in V_A \mid (\forall r \in V)(q \Rightarrow r \text{ implies } r \in R)\}.$$

A set $R \subset V$ of vertices is *alternating reachable* in G if $V_I \cap APre^i(R) \neq \emptyset$ for some $i \in \mathbb{N}$. The *alternating reachability problem* asks whether a given set of vertices is alternating reachable in a given AND–OR graph.

Theorem 2.2 (Immerman [17]). *The alternating reachability problem is complete for PTIME.*

The following lemma relates the alternating-predecessor operator of AND–OR graphs to the uncontrollable-predecessor operator of transition systems.

Lemma 2.1. *Let $G=(V_A, V_O, V_I, \Rightarrow)$ be an AND–OR graph, and let R be a set of vertices of G. Put $V=V_A \cup V_O$, and define the transition system $S_G=(V, V, \rightarrow, V_I, \{\pi\}, \models)$ such that (1) $v \models \pi$ iff $v \in R$, (2) for all AND vertices $v \in V_A$, we have $v \xrightarrow{\sigma} w$ iff $v \Rightarrow w$ and $\sigma = w$, and (3) for all OR vertices $v \in V_O$, we have $v \xrightarrow{\sigma} w$ iff $v \Rightarrow w$ and $\sigma = v$. Then R is alternating reachable in G iff π is unavoidable in S_G.*

Proof. By condition (2) of Lemma 2.1, for all AND vertices $v \in V_A$, if $v \xrightarrow{\sigma} w$ and $v \xrightarrow{\sigma'} w'$ and $w \neq w'$, then $\sigma \neq \sigma'$. By condition (3), for all OR vertices $v \in V_O$, if $v \xrightarrow{\sigma} w$ and $v \xrightarrow{\sigma'} w'$, then $\sigma = \sigma'$. Therefore, for every set $R \subset V$ of vertices,

$$\begin{aligned}
UPre(R) &= \bigcap_{\sigma \in V} (Pre_\sigma(R) \cup V \backslash Pre_\sigma(V)) \\
&= \{v \in V_A \mid (\forall \sigma \in V)(v \in Pre_\sigma(R) \vee v \in V \backslash Pre_\sigma(V))\} \\
&\quad \cup \{v \in V_O \mid (\forall \sigma \in V)(v \in Pre_\sigma(R) \vee v \in V \backslash Pre_\sigma(V))\} \\
&= \{v \in V_A \mid (\forall w \in V)(v \Rightarrow w \text{ implies } w \in R)\} \cup \{v \in V_O \mid (\exists w \in R)(v \Rightarrow w)\} \\
&= APre(R).
\end{aligned}$$

From this and condition (1), the lemma follows. □

Notice that Lemma 2.1 provides a LOGSPACE reduction from the alternating reachability problem to the complement of the safety control decision problem.

Corollary 2.1. *The safety control decision problem for finite transition systems is complete for PTIME.*

2.4. Effectively presented transition systems with finite bisimilarity quotients

The safety controller synthesis problem can be solved not only for finite transition systems, but also for effectively presented transition systems with finite bisimilarity quotients.

Definition 2.8 (*Effective presentation*). A *symbolic reverse-execution theory* for the transition system S consists of a set $\mathscr{F}$ of *formulas*, a formula $\varphi_I \in \mathscr{F}$, and a map $[\![\cdot]\!] : \mathscr{F} \rightarrow 2^Q$ such that (1) every observation $\pi \in \Pi$ is a formula: $[\![\pi]\!] = R_\pi$; (2) for all formulas $\varphi_1, \varphi_2 \in \mathscr{F}$, the three expressions $\varphi_1 \wedge \varphi_2$ and $\varphi_1 \vee \varphi_2$ and $\neg\varphi_1$ are formulas:

$[\![\varphi_1 \wedge \varphi_2]\!] = [\![\varphi_1]\!] \cap [\![\varphi_2]\!]$ and $[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$ and $[\![\neg\varphi_1]\!] = Q \backslash [\![\varphi_1]\!]$; (3) $[\![\varphi_I]\!] = Q_I$; (4) the set $\{\varphi \in \mathscr{F} \mid [\![\varphi]\!] = \emptyset\}$ is recursive; and (5) for each event $\sigma \in \Sigma$, there is a computable map $pre_\sigma : \mathscr{F} \to \mathscr{F}$ such that $[\![pre_\sigma(\varphi)]\!] = Pre_\sigma([\![\varphi]\!])$ for all formulas $\varphi \in \mathscr{F}$. We say that the formula φ *defines* the region $[\![\varphi]\!]$. An *effectively presented transition system* consists of a transition system S together with a symbolic reverse-execution theory for S.

For an effectively presented transition system S, the uncontrollable-predecessor operator can be computed. To wit, for every formula φ of the symbolic reverse-execution theory for S, the region $UPre([\![\varphi]\!])$ is defined by the formula

$$upre(\varphi) = \bigwedge_{\sigma \in \Sigma} (pre_\sigma(\varphi) \vee \neg pre_\sigma(true)). \qquad (\dagger)$$

Definition 2.9 (*Bisimilarity*). A *bisimulation* on the transition system S is an equivalence relation $\cong$ on the state set Q such that (1) if $q \cong r$ then for all observations $\pi \in \Pi$, we have $q \models \pi$ iff $r \models \pi$, and (2) if $q \cong r$ and $q \xrightarrow{\sigma} q'$, then there exists a state $r' \in Q$ such that $r \xrightarrow{\sigma} r'$ and $q' \cong r'$. The largest bisimulation on S is denoted $\equiv$ and called the *bisimilarity relation* of S. Two states q and r of S are *bisimilar* if $q \equiv r$. The equivalence classes of $\equiv$ are called *bisimilarity classes*. The *bisimilarity quotient* $S/_\equiv$ is the transition system $(Q/_\equiv, \Sigma, \to_\exists, Q_\exists, \Pi, \models_\exists)$, where $Q/_\equiv$ is the set of bisimilarity classes for S, where $R \xrightarrow{\sigma}_\exists R'$ iff there exist two states $q \in R$ and $q' \in R'$ such that $q \xrightarrow{\sigma} q'$, where $R \in Q_\exists$ iff $R \cap Q_I \neq \emptyset$, and where $R \models_\exists \pi$ iff $R \cap R_\pi \neq \emptyset$.

If a region R of a transition system is a union of bisimilarity classes, then the weakest-predecessor region $Pre(R)$ is also a union of bisimilarity classes. The following lemma asserts that the same is true for the uncontrollable-predecessor region $UPre(R)$.

Lemma 2.2. *Let S be a transition system, let R be a bisimilarity class of S, and let q and r be two states of S. If $q \in UPre(R)$ and $q \equiv r$, then also $r \in UPre(R)$.*

Proof. Consider a bisimilarity class R and two bisimilar states q and r of S. Suppose that for all events $\sigma \in \Sigma$, either $q \in Pre_\sigma(R)$ or $q \notin Pre_\sigma(Q)$. In the first case, since $q \equiv r$ and R is a bisimilarity class, also $r \in Pre_\sigma(R)$. In the second case, since $q \equiv r$, also $r \notin Pre_\sigma(Q)$. □

It follows that, for each observation π of a transition system S, the uncontrollable region U_π is a union of bisimilarity classes. In particular, if the transition system S has K bisimilarity classes, then $U_\pi = \bigcup_{0 \leqslant i \leqslant K} UPre^i(R_\pi)$ is a finite union. Furthermore, if S is effectively presented, then U_π can be computed by iterating equation (†); it is defined by the formula $\varphi_\pi = \bigvee_{0 \leqslant i \leqslant K} upre^i(\pi)$. Finally, π is avoidable in S iff $[\![\varphi_I \wedge \varphi_\pi]\!] = \emptyset$. For effectively presented S, this can be decided.

Theorem 2.3. *The safety control decision problem is decidable for effectively presented transition systems with finite bisimilarity quotients. Moreover, when an observation is avoidable, a witnessing control map can be computed.*

Proof. We have already proved the first part of the theorem. Once we have computed the uncontrollable region U_π, we can solve the second part as follows. Suppose that the observation π is avoidable. We compute the bisimilarity classes of S by partition refinement, using the symbolic reverse-execution theory of S [7]. Then we construct a control map κ by choosing, for each state in each bisimilarity class R disjoint from U_π, an event σ such that $R \cap Pre_\sigma(Q) \neq \emptyset$ and $R \cap Pre_\sigma(U_\pi) = \emptyset$. Then R_π is unreachable in the closed-loop system $\kappa(S)$. □

Note that in the computation of the uncontrollable region U_π, negation is applied only to atomic formulas of the form $pre_\sigma(true)$, one for each event $\sigma \in \Sigma$. This is important in practice, where negation is often an expensive operation. Theorem 2.3 can be generalized to liveness verification such as μ-calculus model checking [10], and to memory-free liveness control such as control-map synthesis for Rabin chain conditions [26].

3. Previous results about rectangular automata

We now define rectangular automata, our main object of investigation, and review what is known about them.

3.1. Definition of rectangular automata

Definition 3.1 (*Rectangle*). Let $X = \{x_1, \ldots, x_n\}$ be a set of real-valued variables. A *rectangular inequality* over X is a formula of the form $x_i \sim c$, where c is an integer constant, and $\sim$ is one of $<, \leqslant, >, \geqslant$. A *rectangular predicate* over X is a conjunction of rectangular inequalities. The set of all rectangular predicates over X is denoted $Rect(X)$. The rectangular predicate ϕ defines the set of vectors $[\![\phi]\!] = \{y \in \mathbb{R} \mid \phi[X := y]$ is true$\}$. A set of the form $[\![\phi]\!]$, where ϕ is a rectangular predicate, is called a *rectangle.* Every rectangle $B = \Pi_{1 \leqslant i \leqslant n} B_i$ is a product of n intervals of the real line. Given a nonnegative integer $m \in \mathbb{N}$, the rectangular predicate ϕ and the rectangle $[\![\phi]\!]$ are *m-definable* if $|c| \leqslant m$ for every conjunct $x_i \sim c$ of ϕ.

Definition 3.2 (*Rectangular automaton* [14]). A *rectangular automaton A* consists of the following components.

Variables. The finite set $X = \{x_1, \ldots, x_n\}$ of real-valued *variables* represents the continuous part of the system. The number n is the *dimension* of A. We write $\dot{X}$ for the set $\{\dot{x}_i \mid x_i \in X\}$ of dotted variables (which represent first derivatives), and X' for the set $\{x'_i \mid x_i \in X\}$ of primed variables.

Control graph. The finite directed multigraph (V, E) represents the discrete part of the system. The vertices in V are called *control modes*. The edges in E are called *control switches*.

Initial conditions. The function $init: V \rightarrow Rect(X)$ maps each control mode to its *initial condition*, a rectangular predicate. When the automaton control starts in mode v, the variables have initial values inside the rectangle $[\![init(v)]\!]$.

Invariant conditions. The function $inv: V \rightarrow Rect(X)$ maps each control mode to its *invariant condition*, a rectangular predicate. The automaton control may reside in mode v only as long as the values of the variables stay inside the rectangle $[\![inv(v)]\!]$.

Jump conditions. The function *jump* maps each control switch $e \in E$ to a predicate $jump(e)$ of the form $\phi \wedge \phi' \wedge \bigwedge_{i \notin update(e)} (x_i' = x_i)$, where $\phi \in Rect(X)$ and $\phi' \in Rect(X')$ are rectangular predicates, and $update(e) \subset \{1, \ldots, n\}$. The jump condition $jump(e)$ specifies the effect of the change in control mode on the values of the variables: each unprimed variable x_i refers to a value before the control switch e, and each primed variable x_i' refers to the corresponding value after the control switch. So the automaton control may switch across e if (1) the values of the variables are inside the rectangle $[\![\phi]\!]$, and (2) the value of each variable x_i with $i \notin update(e)$ is in the interval $[\![\phi']\!]_i$. Then, the value of each variable x_i with $i \notin update(e)$ remains unchanged by the switch, and the value of each variable x_i with $i \in update(e)$ is updated nondeterministically to an arbitrary new value in the interval $[\![\phi']\!]_i$.

Flow conditions. The function $flow: V \rightarrow Rect(\dot{X})$ maps each control mode v to a *flow condition*, a rectangular predicate that constrains the behavior of the first derivatives of the variables. While time passes with the automaton control in mode v, the values of the variables follow nondeterministically any differentiable trajectory whose first derivative stays inside the rectangle $[\![flow(v)]\!]$.

Events. Given a finite set A of *events* disjoint from $\mathbb{R}_{\geqslant 0}$, the function $event: E \rightarrow A$ maps each control switch to an event.

Thus a rectangular automaton A is a tuple $(X, V, E, init, inv, jump, flow, A, event)$. The automaton A is *m-definable* if every rectangular predicate in the definition of A is m-definable.

Notice that the different variables (coordinates) of a rectangular automaton are completely independent. Indeed, while this is not done here, an n-dimensional rectangular automaton can be viewed as the cartesian product of n one-dimensional rectangular automata [9].

Definition 3.3 (*Nondecreasing and bounded variables*). Let A be an n-dimensional rectangular automaton, and let $i \in \{1, \ldots, n\}$. The variable x_i of A is *nondecreasing* if for every control mode $v \in V$, the invariant interval $[\![inv(v)]\!]_i$ and the flow interval $[\![flow(v)]\!]_i$ are subsets of the nonnegative reals. The variable x_i is *bounded* if for every control mode $v \in V$, the invariant interval $[\![inv(v)]\!]_i$ is a bounded set. The automaton A has *nondecreasing* (resp. *bounded*; *nondecreasing or bounded*) *variables* if all n variables of A are nondecreasing (resp. bounded; either nondecreasing or bounded).

The state of a rectangular automaton has two parts: a discrete (or control) part, and a continuous (or plant) part. The discrete state is a control mode. The continuous state is a valuation for the variables.

Definition 3.4 (*States of rectangular automata*). Let A be a rectangular automaton. A *state* of A is a pair $(v,\mathbf{y})$, where $v \in V$ is a control mode and $\mathbf{y} \in [\![inv(v)]\!]$ is a vector satisfying the invariant condition of v. Thus the set of states is $Q = \{(v,\mathbf{y}) \in V \times \mathbb{R}^n \mid \mathbf{y} \in [\![inv(v)]\!]\}$. A subset of Q is called a *region* of A. A *rectangular state predicate* for A is a function ψ from V to $Rect(X)$. The rectangular state predicate ψ defines the region $[\![\psi]\!] = \{(v,\mathbf{y}) \in Q \mid \mathbf{y} \in [\![\psi(v)]\!]\}$. A region of the form $[\![\psi]\!]$, where ψ is a rectangular state predicate for A, is called a *rectangular region*. The initial condition map defines the rectangular region $Q_\mathrm{I} = [\![init]\!]$ of *initial states*. The rectangular state predicate ψ is *m-definable* if for every vertex $v \in V$, the rectangular predicate $\psi(v)$ is m-definable.

A rectangular automaton makes two types of transitions: jump (or edge, or control) transitions, and flow (or time, or plant) transitions. Jump transitions are instantaneous. They are characterized by a change in control mode, and are accompanied by discrete modifications to the variables in accordance with the jump condition of the control switch. During flow transitions, while time elapses, the control mode remains fixed and the variables evolve continuously via a trajectory that satisfies the flow condition of the active control mode.

Definition 3.5 (*Transitions of rectangular automata*). Let A be a rectangular automaton. For each event $\lambda \in A$, we define the *jump relation* $\xrightarrow{\lambda} \subset Q^2$ by $(v,\mathbf{y}) \xrightarrow{\lambda} (v',\mathbf{y}')$ iff there exists a control switch $e = (v,v') \in E$ such that $event(e) = \lambda$ and $(\mathbf{y},\mathbf{y}') \in [\![jump(e)]\!]$. For each nonnegative real $\delta \in \mathbb{R}_{\geqslant 0}$, we define the *flow relation* $\xrightarrow{\delta} \subset Q^2$ by $(v,\mathbf{y}) \xrightarrow{\delta} (v',\mathbf{y}')$ iff (1) $v = v'$, and (2) there exists a differentiable function $f : [0,\delta] \to [\![inv(v)]\!]$ such that $f(0) = \mathbf{y}$ and $f(\delta) = \mathbf{y}'$, and $\dot{f}(\varepsilon) \in [\![flow(v)]\!]$ for all reals $\varepsilon \in (0,\delta)$, where $\dot{f}$ is the first derivative of f. We say that δ is the *duration* of the flow transition. Since the rectangle $[\![inv(v)]\!]$ is a convex set, it follows that for $\delta > 0$, condition (2) is equivalent to $(\mathbf{y}' - \mathbf{y})/\delta \in [\![flow(v)]\!]$; that is, all flows can be thought of as straight lines.

Every rectangular automaton defines two transition systems.

Definition 3.6 (*Discrete time and dense time*). Let A be an m-definable rectangular automaton. Define the binary relation $\xrightarrow{time} \subset Q^2$ by $(v,\mathbf{y}) \xrightarrow{time} (v',\mathbf{y}')$ iff $(v,\mathbf{y}) \xrightarrow{\delta} (v',\mathbf{y}')$ for some duration $\delta \in \mathbb{R}_{\geqslant 0}$. Define Π to be the set of rectangular state predicates for A, and for all states $(v,\mathbf{y}) \in Q$ and state predicates $\pi \in \Pi$, define $(v,\mathbf{y}) \models \pi$ iff $(v,\mathbf{y}) \in [\![\pi]\!]$. The *discrete-time transition system* of A is defined by $S_A^{\mathrm{disc}} = (Q, A \cup \{1\}, \to, Q_\mathrm{I}, \Pi, \models)$. The *dense-time transition system* of A is defined by $S_A^{\mathrm{dense}} = (Q, A \cup \{time\}, \to, Q_\mathrm{I}, \Pi, \models)$. Thus all flow transitions in the discrete-time transition system are required to have duration 1, while flow transitions in the dense-time transition system can have any nonnegative real durations. We refer to the safety verification problem for transition systems of the form S_A^{disc} (resp. S_A^{dense}), for some rectangular automaton A, as the *discrete-time* (resp. *dense-time*) *safety verification problem* for rectangular automata, and similarly for the control decision and controller synthesis problems. For each instance of safety verification or control, only one specific observation is of interest. Hence we define the *m-observable discrete-time* (resp. *dense-time*) *transition system* $S_{A,m}^{\text{disc}}$ (resp. $S_{A,m}^{\text{dense}}$) to be the same as S_A^{disc} (resp. S_A^{dense}), except that the observation set Π is replaced by the set $\Pi_m \subset \Pi$ of m-definable rectangular state predicates for A.

It follows that the σ-predecessor maps pre_σ of a symbolic reverse-execution theory for the transition systems S_A^{disc} or S_A^{dense} must satisfy the following equivalences over the reals: for all state predicates ψ for A that are formulas of the theory, all control modes $v \in V$, and all events $\lambda \in \Lambda$,

$$pre_\lambda(\psi)(v) \Leftrightarrow \bigvee_{e=(v,v')\in E \text{ with } event(e)=\lambda} (\exists X')(jump(e) \wedge (\psi(v') \wedge inv(v'))[X := X']), \tag{\#1}$$

$$pre_1(\psi)(v) \Leftrightarrow (\exists \dot{X})(flow(v) \wedge (\psi(v) \wedge inv(v))[X := X + \dot{X}]), \tag{\#2}$$

$$pre_{time}(\psi)(v) \Leftrightarrow (\exists \delta \geqslant 0)(\exists \dot{X})(flow(v) \wedge (\psi(v) \wedge inv(v))[X := X + \delta \cdot \dot{X}]), \tag{\#3}$$

where $\varphi[X := X + \delta \cdot \dot{X}]$ stands for the predicate that results from φ by replacing every occurrence of each variable $x_i \in X$ by the term $x_i + \delta \cdot \dot{x}_i$, etc. Notice that both in the discrete-time and dense-time cases, every symbolic reverse-execution theory for the transition system S_A is also a symbolic reverse-execution theory for the m-observable transition system $S_{A,m}$. Furthermore, every bisimulation on S_A is also a bisimulation on $S_{A,m}$.

3.2. Dense-time undecidability results

In dense time, the verification and control of rectangular automata cannot be fully automated.

Theorem 3.1 (Alur et al. [1]). *For rectangular automata with nondecreasing and bounded variables, the dense-time safety verification problem (and thus the dense-time safety control decision problem) is undecidable.*

Research has therefore concentrated on subclasses of rectangular automata. In [14] it is shown that for *initialized* rectangular automata, whose flow condition map is a constant function (i.e., all control modes have the same flow condition), the dense-time safety verification problem (in fact, LTL model checking) can be decided.[4] These

[4] It is not known if the dense-time safety control decision problem is decidable for initialized rectangular automata.

automata, however, have no finite bisimilarity quotients in dense time [9], and therefore further restrictions are desirable.

3.3. *Timed automata*

An important special case of initialized rectangular automata are timed automata. All variables of a timed automaton are clocks, which are nondecreasing variables that advance uniformly at rate 1 while time elapses.

Definition 3.7 (*Timed automaton* [2]). A *timed automaton* is a rectangular automaton A with nondecreasing variables such that $\mathit{flow}(v) = \bigwedge_{1 \leqslant i \leqslant n}(\dot{x}_i = 1)$ for every control mode v. A *triangular inequality* over a set X of variables is a formula of the form $x_i - x_j \sim c$, where $x_i, x_j \in X$ are variables, c is an integer constant, and $\sim$ is one of $<, \leqslant, >, \geqslant$. A *triangular predicate* over X is a conjunction of rectangular and triangular inequalities. A *triangular state predicate* for a timed automaton A is a function that maps every control mode of A to a triangular predicate over the variables of A.

The fundamental theorem for timed automata states that the m-observable dense-time transition system $S^{\text{dense}}_{A,m}$ of an m-definable timed automaton A has a finite bisimilarity quotient and can be presented effectively using triangular state predicates.

Theorem 3.2. *For every m-definable n-dimensional timed automaton A with k control modes, the m-observable dense-time transition system $S^{\text{dense}}_{A,m}$ has a finite bisimilarity quotient with* $\mathrm{O}(k \cdot n! \cdot 2^n \cdot (m+1)^n)$ *many equivalence classes* [2]. *Moreover, the boolean combinations of the triangular state predicates for A form a symbolic reverse-execution theory for the dense-time transition system S^{dense}_{A}* [15].

In particular, for every triangular state predicate ψ for A and every event $\sigma \in A \cup \{\mathit{time}\}$, the σ-predecessor $\mathit{pre}_\sigma(\psi)$ is a triangular state predicate that can be found by quantifier elimination from equivalences (#1) and (#3), the latter of which simplifies in the case of timed automata to

$$\mathit{pre}_{\mathit{time}}(\psi)(v) \;\Leftrightarrow\; (\exists \delta \geqslant 0)((\psi(v) \wedge \mathit{inv}(v))[X := X + \delta]).$$

Corollary 3.1. *For timed automata, the dense-time safety verification problem (in fact, LTL and CTL model checking) can be solved in PSPACE* [2], *and the dense-time safety controller synthesis problem can be solved in EXPTIME* [16].

As in the case of finite transition systems, control is harder than verification. In [2] it is shown that the dense-time safety verification problem for timed automata is hard for PSPACE. From Theorem 4.2 below it will follow that the dense-time safety control decision problem for timed automata is hard for EXPTIME.

4. Discrete-time rectangular automata

4.1. Finite bisimilarity quotients and effective presentation

We show that for every m-definable rectangular automaton A with nondecreasing or bounded variables, the m-observable discrete-time transition system $S_{A,m}^{\mathrm{disc}}$ has a finite bisimilarity quotient and can be presented effectively using rectangular state predicates. More precisely, in discrete time, two states of a rectangular automaton are bisimilar if (1) they have the same control mode, (2) corresponding variable values agree on their integer parts, and (3) corresponding variable values agree on whether they are integral. Moreover, if a variable is nondecreasing, then the automaton cannot distinguish variable values greater than m. If a variable is bounded, then its value stays in the interval $[-m,m]$. It follows that if all variables are nondecreasing or bounded, then the bisimilarity quotient is finite.

Definition 4.1. We use the notation y_i to denote the ith component of the vector $y\in\mathbb{R}^n$. Define the equivalence relation $\approx_n$ on $\mathbb{R}^n$ by $y\approx_n z$ iff $\lfloor y_i\rfloor=\lfloor z_i\rfloor$ and $\lceil y_i\rceil=\lceil z_i\rceil$ for all $1\leqslant i\leqslant n$. Given $m\in\mathbb{N}$, define the equivalence relation $\approx_n^m$ on $\mathbb{R}^n$ by $y\approx_n^m z$ iff for each $1\leqslant i\leqslant n$, either $y_i\approx_1 z_i$, or both y_i and z_i are greater than m, or both y_i and z_i are less than $-m$. For an n-dimensional rectangular automaton A, define the equivalence relations $\cong_A$ and $\cong_A^m$ on the states of A by $(v,y)\cong_A(w,z)$ iff $v=w$ and $y\approx_n z$, and $(v,y)\cong_A^m(w,z)$ iff $v=w$ and $y\approx_n^m z$.

Notice that every equivalence class of $\approx_n$ is a rectangle, and every equivalence class of $\approx_n^m$ is an m-definable rectangle. The following easy lemma implies that every rectangle is a union of $\approx_n$-equivalence classes, and every m-definable rectangle is a union of $\approx_n^m$-equivalence classes.

Lemma 4.1. *Consider two vectors $y,z\in\mathbb{R}^n$. Then $y\approx_n z$ iff for every rectangle $B\subset\mathbb{R}^n$, we have $y\in B$ iff $z\in B$. Moreover, $y\approx_n^m z$ iff for every m-definable rectangle $B\subset\mathbb{R}^n$, we have $y\in B$ iff $z\in B$.*

It follows that every rectangular state predicate ψ for a rectangular automaton A defines a union of $\cong_A$-equivalence classes, and if ψ is m-definable, then $[\![\psi]\!]$ is a union of $\cong_A^m$-equivalence classes. The next theorem states the central observation of this paper.

Theorem 4.1. *Let A be an n-dimensional rectangular automaton with k control modes. The equivalence relation $\cong_A$ is the bisimilarity relation of the discrete-time transition system S_A^{disc}. If A is m-definable and has nondecreasing or bounded variables, then $\cong_A^m$ is the bisimilarity relation of the m-observable discrete-time transition system $S_{A,m}^{\mathrm{disc}}$. The number of equivalence classes of $\cong_A^m$ is $k\cdot(4m+3)^n$.*

Proof. We prove only the second claim of the theorem; the proof of the first claim is similar. So suppose that A is m-definable and has nondecreasing or bounded variables.

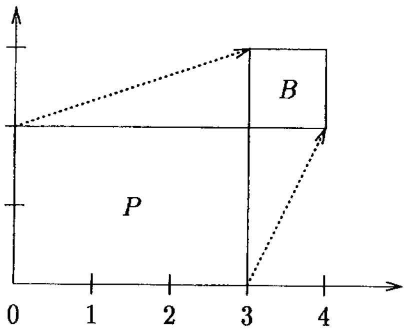

Fig. 1. Given a control mode v, consider the flow condition $flow(v) = (1 \leqslant \dot{x}_1 \leqslant 3 \wedge 1 \leqslant \dot{x}_2 \leqslant 2)$. Let $B = [\![3 \leqslant x_1 \leqslant 4 \wedge 2 \leqslant x_2 \leqslant 3]\!]$ and $P = [\![0 \leqslant x_1 \leqslant 3 \wedge 0 \leqslant x_2 \leqslant 2]\!]$. Then $Pre_1(\{v\} \times B) = \{v\} \times P$.

We show that $\cong_A^m$ is a bisimulation on $S_{A,m}^{\text{disc}}$. By Lemma 4.1 it follows that $\cong_A^m$ is the largest bisimulation, because $q \not\cong_A^m r$ implies that there is an m-definable rectangular state predicate ψ such that $q \in [\![\psi]\!]$ and $r \notin [\![\psi]\!]$. Suppose that $(v,\mathbf{y}) \cong_A^m (w,\mathbf{z})$. By Lemma 4.1, the two states $(v,\mathbf{y})$ and $(w,\mathbf{z})$ satisfy the same m-definable rectangular state predicates. So suppose that $(v,\mathbf{y}) \xrightarrow{\sigma} (v',\mathbf{y}')$. We must show that there exists a state $(w',\mathbf{z}')$ such that $(w,\mathbf{z}) \xrightarrow{\sigma} (w',\mathbf{z}')$ and $(v',\mathbf{y}') \cong_A^m (w',\mathbf{z}')$.

First, assume that $\lambda \in \Lambda$. In this case there exists a control switch e with source $v = w$ such that $event(e) = \lambda$ and $(\mathbf{y},\mathbf{y}') \in [\![jump(e)]\!]$, and $y_i = y_i'$ for each $i \notin update(e)$. Define $\mathbf{z}'$ by $z_i' = z_i$ for $i \notin update(e)$, and $z_i' = y_i'$ for $i \in update(e)$. By Lemma 4.1, since A is m-definable, $(\mathbf{z},\mathbf{z}') \in [\![jump(e)]\!]$ and $\mathbf{z}' \in [\![inv(v')]\!]$. It follows that $(w,\mathbf{z}) \xrightarrow{\lambda} (w',\mathbf{z}')$ and $(v',\mathbf{y}') \cong_A^m (w',\mathbf{z}')$.

Second, assume that $\sigma = 1$ (cf. Fig. 1). In this case $v' = v = w$, and $\mathbf{y}' - \mathbf{y} \in [\![flow(v)]\!]$. We must show that there exists a vector $\mathbf{z}'$ such that $\mathbf{z}' - \mathbf{z} \in [\![flow(v)]\!]$ and $\mathbf{y}' \approx_n^m \mathbf{z}'$ (notice that by Lemma 4.1, since A is m-definable, $\mathbf{y}' \approx_n^m \mathbf{z}'$ implies $\mathbf{z}' \in [\![inv(v)]\!]$). We do this one coordinate at a time. Fix $i \in \{1,\dots,n\}$. Suppose that $y_i > m$. In this case, the ith variable cannot be bounded and must therefore be nondecreasing. It follows that $y_i' > m$ and $z_i > m$. Choose any real $c \in [\![flow(v)]\!]_i$, and define $z_i' = z_i + c$. Since $c \geqslant 0$, we have $y_i' \approx_1^m z_i'$. Now suppose that $y_i \leqslant m$. Then $y_i \geqslant -m$, because the ith variable is nondecreasing or bounded. If $y_i \in \mathbb{N}$ then $z_i = y_i$, because $y_i \approx_1 z_i$. Define $z_i' = y_i'$. Then $z_i' - z_i = y_i' - y_i \in [\![flow(v)]\!]_i$. If $y_i \notin \mathbb{N}$, then $\lfloor y_i \rfloor < y_i$, $z_i < \lceil y_i \rceil$. The set $[\![flow(v)]\!]_i$ is an interval, say, with endpoints $a, b \in \mathbb{N}$ (it is easy to extend the argument to the case $b = \infty$). Thus $[\![flow(v)]\!]_i$ contains the open interval (a,b), and $y_i' \in [y_i + a, y_i + b]$. We show that there exists a real $c \in (a,b)$ such that $y_i' \approx_1 z_i + c$. Since $a, b \in \mathbb{N}$ and $y_i \approx_1 z_i$, it follows that $y_i + a \approx_1 z_i + a$ and $y_i + b \approx_1 z_i + b$. Thus the closed interval $[z_i + a, z_i + b]$ intersects the same $\approx_1$-equivalence classes as does $[y_i + a, y_i + b]$. Since neither $z_i + a$ nor $z_i + b$ is an integer, the same is true for the open interval $(z_i + a, z_i + b)$. Therefore there exists a number $c \in (a,b)$ such that $y_i' \approx_1 z_i + c$. $\square$

Corollary 4.1. *For every rectangular automaton A, the boolean combinations of the rectangular state predicates for A form a symbolic reverse-execution theory for the discrete-time transition system S_A^{disc}.*

Proof. Conditions (1)–(3) from Definition 2.8 are immediate. Condition (4) follows from the fact that the satisfiability problem for boolean combinations of rectangular state predicates is decidable (NP-complete [15]). Condition (5) is derived as follows. Lemma 4.1 and Theorem 4.1 imply that every rectangular state predicate ψ for A (and thus every boolean combination thereof) defines a union of bisimilarity classes for S_A^{disc}. By the definition of bisimilarity, if a region R of a transition system is a union of bisimilarity classes, then the σ-predecessor region $Pre_\sigma(R)$ is also a union of bisimilarity classes. Hence, for every rectangular state predicate ψ for A and every event $\sigma \in \Lambda \cup \{1\}$, there is a disjunction $pre_\sigma(\psi)$ of rectangular state predicates which defines the region $Pre_\sigma([\![\psi]\!])$. This disjunction is finite: it can be found by quantifier elimination from equivalences (#1) and (#2) [4]. □

Indeed, if the rectangular automaton A is m-definable and has nondecreasing or bounded variables, then the boolean combinations of the m-definable rectangular state predicates for A form a symbolic reverse-execution theory for the m-observable discrete-time transition system $S_{A,m}^{\text{disc}}$.

Corollary 4.2. *For rectangular automata with nondecreasing or bounded variables, the discrete-time safety verification problem (in fact, LTL and CTL model checking) can be solved in PSPACE, and the discrete-time safety controller synthesis problem can be solved in EXPTIME.*

Proof. The LTL and CTL parts of the corollary follow from the fact that the space requirement of either model-checking problem is logarithmic in the size of the bisimilarity quotient of the transition system, and polynomial in the size of the temporal formula [19, 20]. The controller synthesis statement follows from Theorem 2.1 and the proof of Theorem 2.3. □

It should be noted that while in the same complexity class, the actual running times of the discrete-time algorithms for rectangular automata are better by a multiplicative exponential factor than the running times of the corresponding dense-time algorithms for timed automata. This is because there, the number of equivalence classes of the bisimilarity quotient is $\Omega(k \cdot n! \cdot m^n)$. By providing tight lower bounds, the following theorem shows that our algorithms are optimal. The second part of the theorem follows from Theorem 4.4 below.

Theorem 4.2. *For timed automata with bounded variables, the discrete-time safety verification problem is hard for PSPACE* [2], *and the discrete-time safety control decision problem is hard for EXPTIME.*

4.2. Sampling-controller synthesis

The dense-time and discrete-time control problems are not realistic, as a controller may enforce arbitrarily many (even infinitely many) consecutive instantaneous jumps. A more natural control model for hybrid systems involves a controller that samples the plant state once per time unit, and then issues a command based upon its measurement. The command may cause a switch in control mode, after which the plant state evolves continuously for one time unit, before receiving the next command. We call this model "sampling control" to distinguish it from discrete-time control. Moreover, we wish to ensure that an observation is avoided not only at the sampling points but also between sampling points. Given a rectangular automaton A, we define a third transition system, S_A^{sample}, such that (1) any control map behaves in a sampling manner, and (2) the observable regions are "large enough" so that they cannot be entered and left by a single flow transition of duration 1. For example, if π is a rectangular state predicate that maps each control mode of A to either *true* or *false*, then R_π is large enough. If the region of unsafe states is not large enough, this may be correctable by increasing the sampling rate (i.e., by reducing the unit of time).

Definition 4.2 (*Sampling control*). Let A be an m-definable rectangular automaton. A rectangular state predicate $\pi \in \Pi$ is *large enough* for A if there are no three states $(v,y),(v,y') \notin R_\pi$ and $(v,y'') \in R_\pi$ such that $(v,y) \xrightarrow{\delta} (v,y'')$ and $(v,y'') \xrightarrow{1-\delta} (v,y')$ for some real $\delta \in (0,1)$. Define $\Pi' \subset \Pi$ to be the set of rectangular state predicates that are large enough for A, and define $((v,y),\theta) \models' \pi$ iff $(v,y) \models \pi$. The *sampling-control transition system* of A is defined by $S_A^{\text{sample}} = (Q \times \{\mathit{control},\mathit{plant}\},\ A \cup \{1\}, \Rightarrow,$ $Q_I \times \{\mathit{control}\}, \Pi', \models')$, where the binary relation $\Rightarrow$ is defined by: (1) for each event $\lambda \in A$, we have $((v,y),\mathit{control}) \overset{\lambda}{\Rightarrow} ((v',y'),\mathit{plant})$ iff $(v,y) \xrightarrow{\lambda} (v',y')$, and (2) $((v,y),$ $\mathit{plant}) \overset{1}{\Rightarrow} ((v',y'),\mathit{control})$ iff $(v,y) \xrightarrow{1} (v',y')$. Thus, in the sampling-control transition system the controller and the plant take turns: first the controller specifies a jump transition, then one time unit passes in a flow transition, and so on. We refer to the safety control decision problem for transition systems of the form S_A^{sample}, for some rectangular automaton A, as the *sampling-control decision problem* for rectangular automata, and similarly for the sampling-controller synthesis problem. The *m-observable sampling-control transition system* $S_{A,m}^{\text{sample}}$ results from S_A^{sample} by restricting the observation alphabet to $\Pi_m \cap \Pi'$, where Π_m is the set of m-definable rectangular state predicates for A.

There is an easy reduction from the sampling-control model to the discrete-time control model.

Theorem 4.3. *For rectangular automata with nondecreasing or bounded variables, the sampling-controller synthesis problem can be solved in EXPTIME.*

Proof. Consider an n-dimensional rectangular automaton A. We reduce sampling-control problems to discrete-time control problems by constructing a rectangular automaton $Ctrl(A)$ such that S_A^{sample} is isomorphic to $S_{Ctrl(A)}^{\text{disc}}$. Second, if A is m-definable, then $Ctrl(A)$ is m-definable, and therefore $S_{A,m}^{\text{sample}}$ is isomorphic to $S_{Ctrl(A),m}^{\text{disc}}$. Third, if A has nondecreasing or bounded variables, then so does $Ctrl(A)$. The automaton $Ctrl(A)$ has dimension $n+1$. Let $X_{Ctrl(A)} = X_A \cup \{x_{n+1}\}$ for a clock $x_{n+1} \notin X_A$. The control graph and events of $Ctrl(A)$ are identical to those of A. For each control mode v, let $init_{Ctrl(A)}(v) = init_A(v) \wedge (x_{n+1} = 1)$, let $inv_{Ctrl(A)}(v) = inv_A(v) \wedge (0 \leqslant x_{n+1} \leqslant 1)$, and let $flow_{Ctrl(A)}(v) = flow_A(v) \wedge (\dot{x}_{n+1} = 1)$. For each control switch e, let $jump_{Ctrl(A)}(e) = jump_A(e) \wedge (x_{n+1} = 1) \wedge (x'_{n+1} = 0)$. It follows that in the discrete-time transition system $S_{Ctrl(A)}^{\text{disc}}$, jump transitions must alternate with flow transitions (of duration 1). Hence the map $f : Q_{Ctrl(A)} \to Q_A \times \{control, plant\}$, defined by $f(v,y,0) = (v,y,plant)$ and $f(v,y,1) = (v,y,control)$, is an isomorphism between the transition systems $S_{Ctrl(A)}^{\text{disc}}$ and S_A^{sample}. If A is m-definable with k control modes, by Theorem 4.1, the bisimilarity quotient of $S_{Crtl(A),m}^{\text{disc}}$ has no more than $k \cdot (4m+3)^{n+1}$ equivalence classes, which is singly exponential in the size of A. □

In order to show that our exponential control algorithms are optimal, we prove that the sampling-control decision problem (and hence also discrete-time control) is EXPTIME-hard already in the restricted case of timed automata with bounded variables.

Theorem 4.4. *For timed automata with bounded variables, the sampling-control decision problem is hard for EXPTIME.*

Proof. For the sake of clarity, we first prove the result for the larger class of rectangular automata with nondecreasing and bounded variables. Afterwards we modify the argument to produce the statement of the theorem.

We reduce the halting problem for alternating Turing machines using polynomial space, which is EXPTIME-hard [8], to the sampling-control decision problem for rectangular automata with nondecreasing and bounded variables. Let M be an alternating Turing Machine with input s so that M uses space $p(|s|)$, for some polynomial $p(\cdot)$. Then M accepts s iff the unique final state u_F is alternating reachable in an AND–OR graph whose vertices are the configurations of M. The set of configurations of M is $U \times \{1, \ldots, p(|s|)\} \times \Gamma^{p(|s|)}$, where U is the state set of M, the second component of the product gives the position of the tape head, and Γ is the tape alphabet. Without loss of generality, we assume that $\Gamma = \{0,1,2\}$, where 2 is the "blank" symbol. We first define a rectangular automaton A with nondecreasing and bounded variables so that the states of A are the configurations of M, and an observation π_F, large enough for A, that is true exactly in the configurations containing the final state u_F. This is done in a way consistent with Lemma 2.1, so that π_F is unavoidable in S_A^{sample} iff M accepts s.

The automaton A uses $p(|s|)$ variables $x_1, \ldots, x_{p(|s|)}$ to store the tape contents. The set of control modes of A is $U \times \{1, \ldots, p(|s|)\}$. The initial condition map of A is defined by $init(u,i) = false$ except when u is the initial state u_I of M and $i = 1$; in that case, $init(u_I, 1) = \bigwedge_{1 \leqslant j \leqslant |s|} (x_j = s_j) \wedge \bigwedge_{|s|+1 \leqslant j \leqslant p(|s|)} (x_j = 2)$. The invariant and flow conditions are constant functions: $inv(u,i) = \bigwedge_{1 \leqslant j \leqslant p(|s|)} (0 \leqslant x_j \leqslant 2)$ and $flow(u,i) =$

$\bigwedge_{1 \leqslant j \leqslant p(|s|)}(\dot{x}_j = 0)$ for all u and i; thus flow transitions have no effect. Each transition t of M consists of a source state $u \in U$, a tape symbol $\gamma \in \Gamma$, and a list of triples (u_j, γ_j, d_j), where $u_j \in U$ is a target state, $\gamma_j \in \Gamma$ is written on the current tape cell, and $d_j \in \{-1, 1\}$ gives the direction in which the tape head moves. For every transition $t = (u, \gamma, (u_j, \gamma_j, d_j)_{j \in J})$ of M, every tape position $1 \leqslant i \leqslant p(|s|)$, and every $j \in J$, we define in A a control switch $e_{t,i,j}$ with source (u, i) and target $(u_j, i + d_j)$. The jump condition *jump*$(e_{t,i,j})$ is $(x_i = \gamma) \wedge (x_i' = \gamma_j) \wedge \bigwedge_{k \neq i}(x_k' = x_k)$. If u is an AND state of M, then *event*$(e_{t,i,j})$ is the target state of $e_{t,i,j}$; if u is an OR state of M, then *event*$(e_{t,i,j})$ is the source state of $e_{t,i,j}$. It follows from Lemma 2.1 that π_F is unavoidable in S_A^{sample} iff M accepts s. This completes the reduction.

To turn A into a timed automaton, all variables are replaced by clocks. Since each jump transition in S_A^{sample} is followed by a flow transition of duration 1, which increases the value of each clock by 1, the relationship between the tape of M and the continuous state of A is violated. To compensate, we introduce between every two transitions of A a widget that decreases the values of the clocks. The widget for $p(|s|) = 2$ is displayed in Fig. 2, where *stable*$_i$ is the predicate $x_i' = x_i$. Inside the widget, every control switch has the same event, so that the control map has no effect. Exactly $p(|s|)$ units of time are spent in the widget, by moving through the sequence $v_1, \ldots, v_{p(|s|)+1}$ of control modes. It follows that if each clock is decreased by $p(|s|) + 1$ exactly once in the widget, then the relationship between the tape of M and the continuous state of A will be restored upon exiting the widget. To decrease the value of the variable x_i, there are three control switches between each pair of control modes v_i and v_{i+1}, one for every possible value of x_i. Since one time unit passes in each mode v_j, for $1 \leqslant j \leqslant i$, the value of x_i is i greater than the value of the ith tape cell of M when the automaton control is in mode v_i. Accordingly, to decrease the value of x_i by $p(|s|) + 1$, the jump conditions on the three control switches are $(x_i = k + i) \wedge (x_i' = k + i - p(|s|) - 1) \wedge \bigwedge_{j \neq i} \mathit{stable}_j$, for $k = 0, 1, 2$. Finally, in order to keep all clock values nonnegative, $p(|s|)$ is added to all constants in the definition of A. Clearly, the size of the resulting timed automaton is polynomial in $|s|$. □

5. Beyond rectangular automata

5.1. Discrete-time undecidability results

We show that the pleasant properties of discrete-time rectangular automata (Theorem 4.1) depend on both conditions, (1) nondecreasing or bounded variables, and (2) rectangularity. If either condition is violated, then already the discrete-time safety verification problem becomes undecidable.

Definition 5.1 (*Triangular automaton*). A *triangular automaton A* has the same components as a rectangular automaton, except that the predicates defining A may be triangular predicates, and need not necessarily be rectangular.

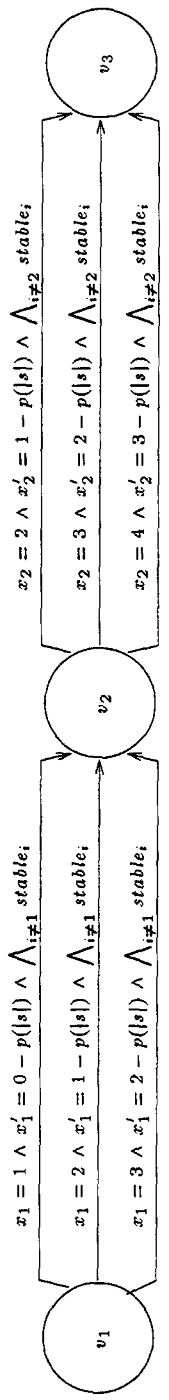
v_1
v_2
v_3
$x_1 = 1 \wedge x_1' = 0 - p(|s|) \wedge \bigwedge_{i \neq 1} stable_i$
$x_1 = 2 \wedge x_1' = 1 - p(|s|) \wedge \bigwedge_{i \neq 1} stable_i$
$x_1 = 3 \wedge x_1' = 2 - p(|s|) \wedge \bigwedge_{i \neq 1} stable_i$
$x_2 = 2 \wedge x_2' = 1 - p(|s|) \wedge \bigwedge_{i \neq 2} stable_i$
$x_2 = 3 \wedge x_2' = 2 - p(|s|) \wedge \bigwedge_{i \neq 2} stable_i$
$x_2 = 4 \wedge x_2' = 3 - p(|s|) \wedge \bigwedge_{i \neq 2} stable_i$

Fig. 2. Widget for decreasing each clock by $p(|s|) + 1$.

Theorem 5.1. *The discrete-time safety verification problem (and thus the discrete-time control decision problem) is undecidable for the class of all rectangular automata, and also for the class of triangular automata with nondecreasing and bounded variables.*

Proof. Both parts use a reduction from the halting problem for two-counter machines, which is undecidable. A two-counter machine M consists of a finite control and two counter variables c_1 and c_2 which range over the natural numbers. The machine has an initial location, a halting location, and three instruction types: branch on zero, increment, and decrement. Instructions of the first type test whether the value of a specified counter is 0, and make a change in the finite control according to the outcome. The increment and decrement instructions are defined in the natural way (if a counter has the minimum value 0, a decrement has no effect). For both parts of the theorem, we define an automaton A of the appropriate class having a control mode v such that the region $\{v\} \times \mathbb{R}^n$ is reachable in the discrete-time transition system S_A^{disc} iff M halts.

For the class of all rectangular automata, the reduction is simple, as the two counter values can be represented directly by the values of two variables. In this case, the dense-time undecidability proof from [18] for two variables, both with slopes $+1$ (for incrementation) and -1 (for decrementation), applies also to the discrete-time framework. Notice that the two variables can be neither nondecreasing (in order to decrement) nor bounded (in order to represent unbounded counter values).

For the class of triangular automata with nondecreasing and bounded variables, the unbounded set of counter values must be encoded within a bounded space. To do so, the dense-time wrapping technique of [14] can be adapted to discrete time. We use variable value $1/2^{\ell}$ to encode counter value ℓ. The rectangular automaton A is five-dimensional. The variables x_1 and x_2 represent the two counters c_1 and c_2, respectively. Then, branching on $c_i = 0$ corresponds to branching on $x_i = 1$, decrementing c_i corresponds to checking for 1 followed by doubling x_i, and incrementing c_i corresponds to halving x_i. The "wrapping variable" x_0 is reset to 0 whenever it reaches 4, and so partitions each run of A into segments. Each decrement instruction of M is simulated by two segments of A, and each increment instruction requires four segments. The variables x_3 and x_4 are auxiliary variables used to implement the increment and decrement instructions.

The crux of the proof is the doubling of a variable. In Fig. 3 we give a widget that doubles the value of x_1. The predicate $\mathit{stable}(i_1, i_2, \ldots, i_l)$ is shorthand for the predicate $\bigwedge_{1 \leqslant j \leqslant l} (x'_{i_j} = x_{i_j})$. Invariant conditions (not shown in the figure) bound all variables from below by 0 and from above by 4. The control modes v_1 and v_2 make up a segment that executes the assignment $x_3 := x_1$; that is, upon exit from v_2 the values of x_1 and x_3 are both equal to the original value of x_1, and the value of x_2 is still equal to the original value of x_2. Similarly, the control modes v_3 and v_4 make up a segment that executes the assignment $x_1 := 2x_1$. Thus, if the value of x_1 is α, for $0 < \alpha \leqslant 1$, when the jump transition into mode v_1 is made, then the next time that the jump transition out of mode v_4 is made, the value of x_1 is 2α (and the value of x_2 remains unchanged throughout the widget). Moreover, such a next time exists, because the variables can

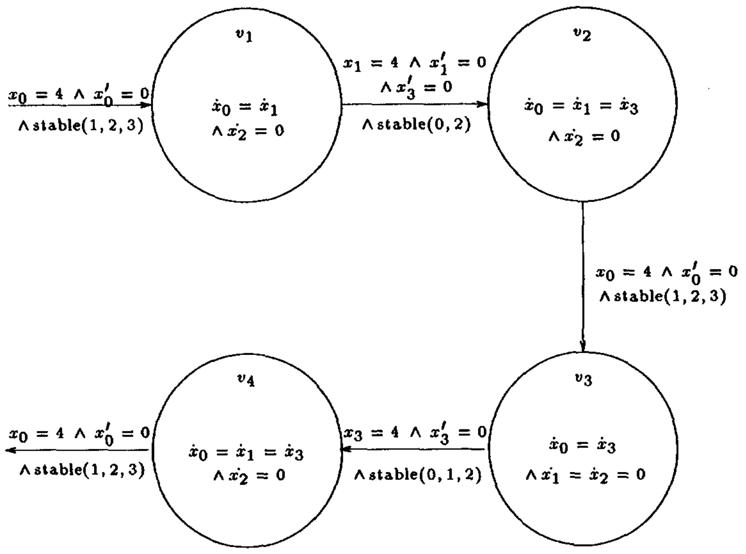
v_1
$\dot{x}_0 = \dot{x}_1 \wedge \dot{x}_2 = 0$
v_2
$\dot{x}_0 = \dot{x}_1 = \dot{x}_3 \wedge \dot{x}_2 = 0$
v_3
$\dot{x}_0 = \dot{x}_3 \wedge \dot{x}_1 = \dot{x}_2 = 0$
v_4
$\dot{x}_0 = \dot{x}_1 = \dot{x}_3 \wedge \dot{x}_2 = 0$
$x_0 = 4 \wedge x_0' = 0 \wedge \mathrm{stable}(1,2,3)$
$x_1 = 4 \wedge x_1' = 0 \wedge x_3' = 0 \wedge \mathrm{stable}(0,2)$
$x_0 = 4 \wedge x_0' = 0 \wedge \mathrm{stable}(1,2,3)$
$x_3 = 4 \wedge x_3' = 0 \wedge \mathrm{stable}(0,1,2)$
$x_0 = 4 \wedge x_0' = 0 \wedge \mathrm{stable}(1,2,3)$

Fig. 3. Doubling the value of x_1.

conspire to move at the correct speeds to make all of the jump transitions at integer time points. Notice that the widget requires triangular flow conditions.

Halving x_2 can be reduced to doubling: first use $x_3 = x_4$ to nondeterministically guess what half of x_2 should be (this requires one segment), then double x_3 while keeping x_4 unchanged (two segments), and finally check that $x_2 = x_3$ and assign to x_2 the value of x_4 (one segment). □

5.2. Generalized rectangular automata

The proof of Theorem 5.1 shows that triangular flow conditions lead to undecidability. On the other hand, it is well known that the pleasant properties of timed automata (Theorem 3.2) are preserved if rectangularity is relaxed to triangularity in initial, invariant, jump, and safety conditions. We conclude with a similar observation for discrete-time rectangular automata. The admission of triangular predicates, however, comes at a cost: the number of bisimilarity classes increases by a multiplicative exponential factor.

Definition 5.2 (*Generalized rectangular automaton*). A *generalized rectangular automaton A* is a triangular automaton whose flow conditions are rectangular predicates. The (*m-observable*) *discrete-time transition system* T_A^{disc} (resp. $T_{A,m}^{\mathrm{disc}}$) for A is defined like the (m-observable) discrete-time transition system for rectangular automata, except that the observation alphabet is the set of (m-definable) triangular state predicates for A.

Definition 5.3. For $y \in \mathbb{R}$, let $\langle y \rangle = y - \lfloor y \rfloor$ be the fractional part of y. Define the equivalence relation $\sim_n$ on $\mathbb{R}^n$ by $\mathbf{y} \sim_n \mathbf{z}$ iff (1) $\mathbf{y} \approx_n \mathbf{z}$ and (2) for all $i, j \in \{1, \ldots, n\}$, we have $\langle y_i \rangle \leqslant \langle y_j \rangle$ iff $\langle z_i \rangle \leqslant \langle z_j \rangle$. Given $m \in \mathbb{N}$, define the equivalence relation $\sim_n^m$ on $\mathbb{R}^n$ by $y \sim_n^m z$ iff (1) $y \approx_n^m z$ and (2) for all $i,j \in \{1,\dots,n\}$ with $y_i, y_j \in [-m,m]$, we have $\langle y_i \rangle \leqslant \langle y_j \rangle$ iff $\langle z_i \rangle \leqslant \langle z_j \rangle$. For an n-dimensional generalized rectangular automaton A, define the equivalence relations $\simeq_A$ and $\simeq_A^m$ on the states of A by $(v,y) \simeq_A (w,z)$ iff $v = w$ and $y \sim_n z$, and $(v,y) \simeq_A^m (w,z)$ iff $v = w$ and $y \sim_n^m z$.

The relation $\simeq_A^m$ is the bisimilarity relation of the m-observable dense-time transition system $S_{A,m}^{\text{dense}}$ for any m-definable timed automaton A [2]. The following statements about generalized rectangular automata correspond to statements made in Section 4 about rectangular automata, and their proofs are similar. We give only the most interesting part of the argument.

Theorem 5.2. *Let A be a generalized n-dimensional rectangular automaton with k control modes. The equivalence relation $\simeq_A$ is the bisimilarity relation of the discrete-time transition system T_A^{disc}. If A is m-definable and has nondecreasing or bounded variables, then $\simeq_A^m$ is the bisimilarity relation of the m-observable discrete-time transition system $T_{A,m}^{\text{disc}}$. The number of equivalence classes of $\simeq_A^m$ is $O(k \cdot n! \cdot 2^n \cdot (4m+3)^n)$.*

The factor $n!$ in the number of bisimilarity classes arises from considering all orderings of the fractional parts of variables. The factor 2^n arises from considering whether the fractional parts of different variables are equal.[5]

Proof. The interesting case of the proof concerns time transitions of duration 1 (cf. Fig. 4). Consider three vectors $y, y', z \in \mathbb{R}^n$, and an open flow rectangle $F = \Pi_{1 \leqslant i \leqslant n} (a_i, b_i)$ (other types of flow rectangles are handled similarly). Suppose that $y \sim_n z$ and $y' - y \in F$. We must show that there exists a vector $z' \in \mathbb{R}^n$ such that $z' - z \in F$ and $y' \sim_n z'$. Unlike in the case of rectangular automata, the n coordinates cannot be considered independently. Without loss of generality we assume that $\langle y'_1 \rangle \leqslant \langle y'_2 \rangle \leqslant \cdots \leqslant \langle y'_n \rangle$. Assuming that we have already chosen suitable components $z'_1, \dots, z'_{i-1}$ according to the following procedure, we choose the component z'_i so that after all choices have been made, the vector z' satisfies the desired criteria. If $y_i \in \mathbb{N}$, choose $z'_i = y'_i$. If $y_i \notin \mathbb{N}$, then according to the proof of Theorem 4.1, there exists a number $c_i \in (a_i, b_i)$ such that $y'_i \approx_1 z_i + c_i$. By the definition of $\approx_1$, it follows that $y'_i \approx_1 z_i + d_i$ also for all numbers $d_i \in (c_i, \lceil z_i + c_i \rceil - z_i)$. This allows us to choose $z'_i = z_i + d_i$ for some d_i such that if $i \geqslant 2$, then $\langle z'_i \rangle \geqslant \langle z'_{i-1} \rangle$. □

Corollary 5.1. *For every generalized rectangular automaton A, the boolean combinations of the triangular state predicates for A form a symbolic reverse-execution theory for the discrete-time transition system T_A^{disc}.*

[5] An exact formula for the number of bisimilarity classes can be found in [13].

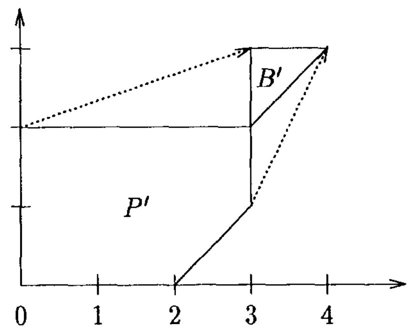

Fig. 4. Given a control mode v, consider the flow condition $flow(v) = (1 \leqslant \dot{x}_1 \leqslant 3 \wedge 1 \leqslant \dot{x}_2 \leqslant 2)$. Let $B' = [\![3 \leqslant x_1 \leqslant 4 \wedge 2 \leqslant x_2 \leqslant 3 \wedge x_1 \leqslant x_2 + 1]\!]$ and $P' = [\![0 \leqslant x_1 \leqslant 3 \wedge 0 \leqslant x_2 \leqslant 2 \wedge x_1 \leqslant x_2 + 2]\!]$. Then $Pre_1(\{v\} \times B') = \{v\} \times P'$.

Corollary 5.2. *For generalized rectangular automata with nondecreasing or bounded variables, the discrete-time safety verification problem (in fact, LTL and CTL model checking) can be solved in PSPACE, and the discrete-time safety controller synthesis problem (in fact, sampling-controller synthesis) can be solved in EXPTIME.*

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine, The algorithmic analysis of hybrid systems, Theoret. Comput. Sci. 138 (1995) 3–34.

[2] R. Alur, D.L. Dill, A theory of timed automata, Theoret. Comput. Sci. 126 (1994) 183–235.

[3] R. Alur, T.A. Henzinger, Modularity for timed and hybrid systems, in: Proc. CONCUR: Concurrency Theory, Lecture Notes in Computer Science, vol. 1243, Springer, Berlin, 1997, pp. 74–88.

[4] R. Alur, T.A. Henzinger, P.-H. Ho, Automatic symbolic verification of embedded systems, IEEE Trans. Software Eng. 22 (1996) 181–201.

[5] C. Beeri, On the membership problem for functional and multivalued dependencies in relational databases, ACM Trans. Database Systems 5 (1980) 241–259.

[6] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, L.J. Hwang, Symbolic model checking: 10^{20} states and beyond, Inform. Comput. 98 (1992) 142–170.

[7] A. Bouajjani, J.-C. Fernandez, N. Halbwachs, Minimal model generation, in: Proc. CAV: Computer-Aided Verification, Lecture Notes in Computer Science, vol. 531, Springer, Berlin, 1990, pp. 197–203.

[8] A.K. Chandra, D.C. Kozen, L.J. Stockmeyer, Alternation, J. ACM 28 (1981) 114–133.

[9] T.A. Henzinger, Hybrid automata with finite bisimulations, in: Proc. ICALP: Automata, Languages, and Programming, Lecture Notes in Computer Science, vol. 944, Springer, Berlin, 1995, pp. 324–335.

[10] T.A. Henzinger, The theory of hybrid automata, in: Proc. LICS: Logic in Computer Science, IEEE Computer Society Press, Silver Spring, Md, 1996, pp. 278–292.

[11] T.A. Henzinger, P.-H. Ho, H. Wong-Toi, Algorithmic analysis of nonlinear hybrid systems, IEEE Trans. Automat. Control 43 (1998) 540–554.

[12] T.A. Henzinger, P.-H. Ho, H. Wong-Toi, HyTech: a model checker for hybrid systems, Software Tools Technol. Transfer 1 (1997) 110–122.

[13] T.A. Henzinger, P.W. Kopke, State equivalences for rectangular hybrid automata, in: Proc. CONCUR: Concurrency Theory, Lecture Notes in Computer Science, vol. 1119, Springer, Berlin, 1997, pp. 530–545.

[14] T.A. Henzinger, P.W. Kopke, A. Puri, P. Varaiya, What's decidable about hybrid automata?, in: Proc. STOC: Theory of Computing, ACM Press, New York, 1995, pp. 373–382.

[15] T.A. Henzinger, X. Nicollin, J. Sifakis, S. Yovine, Symbolic model checking for real-time systems, Inform. Comput. 111 (1994) 193–244.

[16] G. Hoffmann, H. Wong-Toi, The input-output control of real-time discrete-event systems, in: Proc. RTSS: Real-time Systems Symp., IEEE Computer Society Press, Silver Spring, Md, 1992, pp. 256 –265.

[17] N. Immerman, Number of quantifiers is better than number of tape cells, J. Comput. System Sci. 22 (1981) 384–406.

[18] Y. Kesten, A. Pnueli, J. Sifakis, S. Yovine, Integration graphs: a class of decidable hybrid systems, in: Hybrid Systems, Lecture Notes in Computer Science, vol. 736, Springer, Berlin, 1993, pp. 179–208.

[19] O. Kupferman, Model checking for branching-time temporal logics, Ph.D. Thesis, The Technion, Haifa, Israel, 1995.

[20] O. Lichtenstein, A. Pnueli, Checking that finite-state concurrent programs satisfy their linear specification, in: Proc. POPL: Principles of Programming Languages, ACM Press, New York, 1985, pp. 97–107.

[21] O. Maler, A. Pnueli, J. Sifakis, On the synthesis of discrete controllers for timed systems, in: Proc. STACS: Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, vol. 900, Springer, Berlin, 1995, pp. 229–242.

[22] A. Puri, V. Borkar, P. Varaiya, ε-Approximation of differential inclusions, in: Hybrid Systems III, Lecture Notes in Computer Science, vol. 1066, Springer, Berlin, 1996, pp. 362–376.

[23] A. Puri, P. Varaiya, Decidability of hybrid systems with rectangular differential inclusions, in: Proc. CAV: Computer-Aided Verification, Lecture Notes in Computer Science, vol. 818, Springer, Berlin, 1994, pp. 95–104.

[24] P.J. Ramadge, W.M. Wonham, Supervisory control of a class of discrete-event processes, SIAM J. Control Optim. 25 (1987) 206–230.

[25] J.W. Thatcher, J.B. Wright, Generalized finite-automata theory with an application to a decision problem of second-order logic, Math. Systems Theory 2 (1968) 57–81.

[26] W. Thomas, On the synthesis of strategies in infinite games, in: Proc. STACS: Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, vol. 900, Springer, Berlin, 1995, pp. 1–13.