Proceedings of the 36th
Conference on Decision & Control
San Diego, California USA • December 1997

The Synthesis of Controllers for Linear Hybrid Automata

Howard Wong-Toi
Cadence Berkeley Labs*

Abstract

We present a semidecision procedure for synthesizing controllers for hybrid systems modeled as linear hybrid automata. The procedure is easily modified for partial observability, at the cost of completeness. The procedure has been implemented, and tested on the synthesis of controllers for various models of a steam boiler. Since the synthesis procedure may generate controllers that are Zeno, *i.e.* they prevent time from diverging, we provide sufficient, but not necessary, conditions on linear hybrid automata that guarantee that any synthesized controller is nonZeno.

1 Introduction

Controller synthesis for hybrid systems is being approached from two different angles. On one hand, the theory of linear and nonlinear control is being extended to cover switching mode controllers, where analysis is over a detailed model of continuous behavior [14]. On the other hand, the theory of supervisory control, in its earliest form, completely abstracts continuous behavior [15]. Plants are discrete event systems. A hybrid plant could be modeled as a discrete plant by placing the values of its continuous variables into discrete ranges [7, 5]. Discrete event control has been extended to handle models of the plant with real-valued variables. The controller reacts in real-time to the plant's behavior by applying discrete control actions. Initial consideration focused on *timed* systems, where the only continuous variables are clocks [17, 4, 13]. This paper extends controller synthesis to hybrid plants, as proposed in [13].

We take a symbolic approach to controller synthesis, as advocated in [5] for reducing the practical complexity of the synthesis algorithm, and in [13, 4] for handling the infinite state spaces encountered in timed systems. Our goal is to maximize the automation in controller synthesis, while retaining as much detail as possible in the model of continuous behavior. We therefore use linear hybrid automata (LHAs) as our model of hybrid plants [8]. They admit automated symbolic analysis, and have been used successfully for automated formal verification. These automata combine the discrete dynamics of finite-state transition systems with continuous dynamics specified by differential inclusions. They operate under a quite severe linearity restriction—essentially that the behavior of the continuous variables at each control mode be defined by differential inclusions of the form $\mathbf{A}\dot{\mathbf{x}} \geq \mathbf{b}$, for a constant matrix $\mathbf{A}$, vector of first derivatives $\dot{\mathbf{x}}$, and constant vector $\mathbf{b}$.[1]

The symbolic procedures we describe for synthesizing controllers can be viewed as solving a two-person game between a nondeterministic hybrid plant and a controller. The controller continually watches the plant configuration, and either chooses a control action to be performed instantaneously, or allows some non-zero amount of time to pass. The plant may either make uncontrollable discrete jumps in its configuration, or choose a flow for its continuous variables, should the controller be willing for time to pass. The goal for the controller is to ensure that all reachable configurations lie within a set of safe configurations. We give an explicit formulation for the set of configurations for which there exist winning controllers. The set is computable using operators that are implementable via geometric algorithms over convex linear sets. The procedure, however, is not guaranteed to terminate, as might be expected since even reachability analysis over linear hybrid automata is only semidecidable. We also describe a simple means for generating controllers that take into account partial observability of the plant's configuration. The synthesis procedure has been implemented, and tested on the synthesis of controllers for various models of a steam boiler [1].

A controller that performs infinitely many discrete control actions within a finite amount of time is clearly not physically realizable. The *nonZenoness* property for a system asserts that there is a divergent behavior originating from every reachable configuration. A controller is *nonZeno* for a hybrid plant if the closed-loop system is nonZeno. We provide a condition on the hybrid plant under which every synthesized controller is nonZeno. With this, we show how to synthesize nonZeno controllers for systems that are either sampled every Δ time units, or have a minimum delay of Δ time units between sampling points.

Related work

Henzinger and Kopke prove decidability results for the discrete-time control problem for the restricted class of *rectangular automata*, where the variable's rates of change and jump conditions are defined using upper and lower bounds only [10]. Heymann et al. study controller synthesis for a subclass of hybrid systems similar to rectangular automata [11]. They manually apply their synthesis procedure to obtain a controller for the steam boiler. The synthesis reported here, on the other hand, is automated and considers also partial observability and nonZenoness. Asarin et al. synthesize controllers that do not take infinitely many control actions in a finite period of time, but this restriction does not ensure that the closed system is nonZeno [4]. Alur and Henzinger synthesize a reactive controller for a reactive hybrid module, using a doubly-nested fixpoint characterization that appears to be computationally more challenging than the fixpoint we present here [3]. Along different lines, Tittus and Egardt synthesize control laws for hybrid automata with deterministic straight-line flows, subject to eventuality specifications [16].

* Cadence Design Systems, Inc., 2001 Addison St, Third Floor, Berkeley, CA 94704. Email: howard@cadence.com

[1] This definition of linearity differs from that commonly used in systems theory.

0-7803-3970-8/97 $10.00 © 1997 IEEE

2 Controller Synthesis Problem

We use LHAs as our model for the plant [2]. The requirement specification is a safety property, expressed as a subset of the plant's configurations. A subset of events are designated as control events. The goal of the controller is to continually observe the plant, and force control events at appropriate times, so that the plant always remains within the safe set of configurations.

2.1 Linear hybrid automata

Syntax: Given a finite set X of real-valued variables, we let $Conv(X)$ denote the set of finite conjunctions of linear inequalities over X. For a predicate ϕ over X, we write $[\![\phi]\!]$ for the set of states in $\mathbb{R}^n$ satisfying ϕ. A *linear hybrid automaton (LHA)* $H = (X, V, E, inv, init, jump, flow, \Sigma, event)$ is a system consisting of the following components:

Variables A finite set $X = \{x_1, \ldots, x_n\}$ of real-valued variables.

Control graph A finite directed multigraph (V, E) of *control modes* V and *control switches* E that represent the discrete component of H.

Invariant conditions A function $inv : V \to Conv(X)$ that assigns an *invariant condition* to each control mode.

Initial conditions A function $init : V \to Conv(X)$ that assigns an *initial condition* to each control mode.

Jump conditions A function $jump : E \to Conv(X \cup X')$ where $X' = \{x'_1, \ldots, x'_n\}$ and the variable x'_i represents the value of x_i after the switch. The function assigns to each control switch a *jump condition* that relates the values of the variables before the switch with their values immediately after the switch.

Flow conditions A function $flow : V \to Conv(\dot{X})$, where $\dot{X} = \{\dot{x}_1, \ldots, \dot{x}_n\}$ and $\dot{x}_i$ represents the first derivative of x_i. We assume that the set of states satisfying $flow(v)$ is bounded for each mode v.

Event set A finite set Σ of *events*, partitioned into a set Σ_u of *uncontrollable* events and a set Σ_c of *controllable* events.

Event labels A function $event : E \to \Sigma$ that assigns an event label to every control switch. Switches with labels in Σ_c (Σ_u *resp.*) are called *controllable* (*uncontrollable resp.*).

Semantics: A *configuration* of the LHA H is a pair $(v, \mathbf{a}) \in V \times \mathbb{R}^n$ such that $\mathbf{a} \in [\![inv(v)]\!]$. The set of configurations is denoted $\mathcal{C}$. The role of the invariant condition $inv(v)$ is to restrict the values of the variables while automaton control remains in mode v. In particular, there may be a bound on the duration of time that can elapse while control remains in a given mode. A configuration $(v, \mathbf{a})$ is *initial* if $\mathbf{a} \in [\![init(v)]\!]$. Given a set K of configurations, $\overline{K}$ denotes its complement. The dynamics of the LHA consists of *jump steps*, which correspond to instantaneous changes in the control mode and the values of the variables, and *flow steps*, which correspond to time passing while the continuous variables evolve and the control mode remains fixed. Let $q = (v, \mathbf{a})$ and $q' = (v', \mathbf{a}')$ be two configurations. A triple $(q, \sigma, q') \in \mathcal{C} \times \Sigma \times \mathcal{C}$, denoted $q \xrightarrow{\sigma}_H q'$, is a *jump step* iff there exists a control switch $e = (v, v') \in E$ such that $event(e) = \sigma$ and $(\mathbf{a}, \mathbf{a}') \in [\![jump(e)]\!]$. A triple $(q, \delta, q') \in \mathcal{C} \times \mathbb{R}_{\geq 0} \times \mathcal{C}$, denoted $q \xrightarrow{\delta}_H q'$, is a *flow step* iff (1) $v = v'$ and (2) there exists a differentiable function $\rho : [0, \delta] \to [\![inv(v)]\!]$ such that (a) $\rho(0) = \mathbf{a}$, (b) $\rho(\delta) = \mathbf{a}'$, and (c) for all $0 \leq d \leq \delta$, $\rho'(d) \in [\![flow(v)]\!]$, where ρ' is the first time derivative of ρ. Such a differentiable function ρ is called a *witness flow* for $q \xrightarrow{\delta}_H q'$.

The set $Post(W, a)$ of *successors* of the set W of configurations for the event $a \in \Sigma \cup \mathbb{R}_{\geq 0}$ is $\{q' \mid \exists q \in W.q \xrightarrow{a} q'\}$. Analogously, the predecessor set $Pre(W, a)$ is defined to be $\{q \mid \exists q' \in W.q \xrightarrow{a} q'\}$. For a subset of events $\Sigma' \subseteq \Sigma$, let $Pre(W, \Sigma') = \bigcup_{\sigma \in \Sigma'} Pre(W, \sigma)$. A predicate is *linear* if it is a finite disjunction of predicates in $Conv(X)$. A *linear region* is a subset of configurations of the form $\bigcup_{v \in V} \{v\} \times [\![\phi_v]\!]$ for linear predicates ϕ_v. It is *closed* iff it is definable using linear predicates ϕ_v with $[\![\phi_v]\!]$ closed for each $v \in V$. The set of linear regions for the LHA H is denoted $Regions(H)$. The event $a \in \Sigma \cup \mathbb{R}_{\geq 0}$ is *enabled* at configuration q iff $Post(\{q\}, a)$ is nonempty. A *trajectory* originating from a configuration q is a finite or infinite sequence of steps $q \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots$. The configuration q' is *reachable* from q iff there exists a finite trajectory from q ending at q'. The operators *Pre*, *Post*, and boolean operations are closed over linear regions, and effectively computable.

Example 2.1 Consider a simple model of a steam boiler plant [1], whose water volume is determined by the rate at which steam is being produced, whether each of two supply pumps is active, and whether a draining valve is open. LHAs depicting the system appear in Figure 1. We describe only the pump and water automata. The pump automaton P_i has two continuous variables: p_i for the volume of water being pumped into the boiler, and a clock t_i. It has three control modes. The flow condition for modes *off* and *going_on* is $\dot{p}_i = 0$, indicating zero water flow. Each pump, once active, delivers water at a rate of 4 liters per second, as specified by the flow condition $\dot{p}_i = 4$ for mode *on*. The event labels p_i*_on* and p_i*_off* are in Σ_c, and p_i*_active* is in Σ_u. There is a delay of 5 seconds between when the controller turns on the pump, and when the pump starts delivering water. This delay is enforced by the reset of the clock on the switch labeled p_i*_on* (modeled by the jump condition $t'_i = 0$), the invariant $t_i \leq 5$ forcing control to leave the mode *going_on* within 5 seconds, and the guard $t_i = 5$, ensuring that pumping does not commence until exactly 5 seconds delay. The flow condition in the water automaton relates the water level w to the pumping, draining, and steam emission rates.

The plant to be controlled consists of the parallel composition of the automata in the figure. We briefly indicate how LHAs are composed. See [2] for a formal definition. The composition $H_1 \parallel H_2$ is defined for two LHAs H_1 and H_2 only when no event label is controllable in one automaton but uncontrollable in the other. The set of control modes of $H_1 \parallel H_2$ is the crossproduct of the components' modes, with the initial, invariant, and flow conditions being the conjunction of those for the components. Switches are synchronized whenever

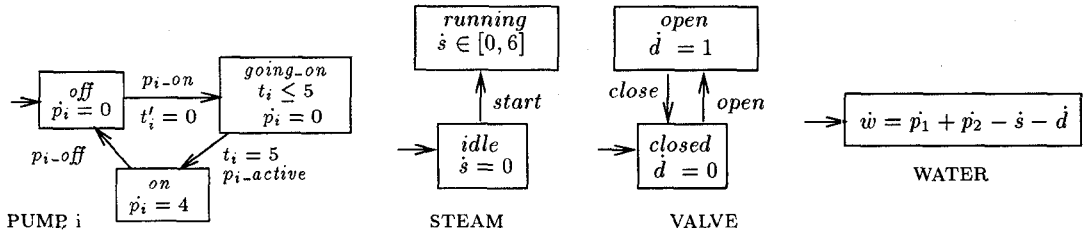

Figure 1: Linear hybrid automata for a steam boiler

their event labels appear in both components' event sets, and occur independently otherwise.

2.2 Controllers for linear hybrid automata

The controller continuously observes the plant with no time delay, and enforces its control decisions instantaneously, *i.e.* there is no delay for decision-making nor for triggering the control events. The controller may, of course, choose to activate no control events for a while, being content to observe the plant's evolution. A controller C for the LHA H is a function $f_C : \mathcal{C} \to \Sigma_c \cup \{\lambda\}$. The intended meaning is that if $f_C(q) = \sigma_c$, then the controller issues a command to take a control switch labeled σ_c; if $f_C(q) = \lambda$, then it is happy to take no action but to let time pass. We require that for all $q \in \mathcal{C}$ and for all $\sigma \in \Sigma$, $f_C(q) = \sigma$ implies that σ is enabled at q. We stipulate that for each controllable event $\sigma_c \in \Sigma_c$, the inverse image of σ is a closed linear region. The reason for this restriction is given later.

We define the coupling (H, C) of the LHA H with a controller C as an infinite-state transition system. Transition steps in (H, C) are either uncontrollable jump steps (the controller has no jurisdiction over the uncontrollable mode switches), controlled jump steps (corresponding to the forcing of controllable switches), or controlled flows (corresponding to witness flows that pass through configurations where the controller lets time elapse). Note that when the controller attempts to force a control event, no time is allowed to elapse: either a control step takes place, or an uncontrollable step occurs, but the controller cannot govern which. Let $q = (v, \mathbf{a})$ and $q' = (v', \mathbf{a}')$ be two configurations. For every controllable event $\sigma_c \in \Sigma_c$, the triple (q, σ_c, q'), denoted $q \xrightarrow{\sigma_c}_{(H,C)} q'$, is a *controlled jump step* for (H, C) iff $q \xrightarrow{\sigma_c}_H q'$ and $f_C(q) = \sigma_c$. For every uncontrollable event $\sigma_u \in \Sigma_u$, the triple (q, σ_u, q'), denoted $q \xrightarrow{\sigma_u}_{(H,C)} q'$, is an *uncontrollable jump step* iff $q \xrightarrow{\sigma_u}_H q'$. For each $\delta \in \mathbb{R}_{\geq 0}$, the triple (q, δ, q'), denoted $q \xrightarrow{\delta}_{(H,C)} q'$, is a *controlled flow step* iff (1) $v = v'$ and (2) there exists a differentiable function $\rho : [0, \delta] \to [\![inv(v)]\!]$ such that (a) $f_C(v, \rho(d)) = \lambda$ for all $0 \leq d < \delta$, and (b) ρ is a witness flow for $q \xrightarrow{\delta} q'$ in H. A *controlled trajectory* of (H, C) originating from the configuration q is a finite or infinite sequence $q = q_0 \xrightarrow{a_1}_{(H,C)} q_1 \xrightarrow{a_2}_{(H,C)} \cdots$ such that for each i, $q_i \xrightarrow{a_{i+1}}_{(H,C)} q_{i+1}$ is either a controlled jump step, an uncontrollable jump step, or a controlled flow step. A configuration is *initial* in (H, C) if it is initial in H; it is *control-reachable* from q if it is reachable from q via a controlled trajectory. The reason for the closure requirement on $f_C^{-1}(\sigma_c)$ is to avoid flows from being blocked because they cannot progress into an open region of configurations for which the controller chooses the action σ_c: there may be no earliest instant at which the control is enabled.

Control problem statement

A *safety requirement safe* for a LHA H is a linear region of H. A controller for H is *legal* if all configurations that are control-reachable from the initial configurations of H lie in *safe*. The supervisory control problem for a LHA H and a linear safety requirement *safe* is to determine whether there exists a legal controller C for H. Furthermore, a legal controller should be constructed if any exists.

3 Controller Synthesis Procedures

3.1 Synthesis with full observability

We extend well-known concepts from supervisory control theory to our hybrid control problem [15]. A configuration q of H is *controllable* w.r.t. the safety requirement *safe* iff there exists a controller C such that all configurations of (H, C) control-reachable from q lie within *safe*. A controller for H exists iff all initial configurations are controllable.

We generalize the familiar synthesis algorithm for discrete systems found in [15], and expressed symbolically in [5]. We successively prune away "bad" configurations from the set of potentially controllable configurations to yield the maximal set of controllable configurations. We define the *unavoidable predecessors* operator $\gamma : Regions(H) \to Regions(H)$ that maps a linear region K to the set of all configurations from which it is impossible for any controller to prevent the plant from reaching K in either one controlled flow step or one controlled flow followed by an uncontrollable jump step. We first define the set of configurations from which a controller could issue a control action that avoids K. For a given linear region K, let the configuration q be an *escape* configuration if it is not in K and there exists a controllable event σ_c enabled at q such that $Post(q, \sigma_c) \subseteq \overline{K}$, *i.e.* all σ_c-steps lead to configurations not in K. Let $escape(K)$ be the set of escape configurations for K, *i.e.* $escape(K) = \overline{K} \cap \bigcup_{\sigma_c \in \Sigma_c} (Pre(\overline{K}, \sigma_c) \cap \overline{Pre(K, \sigma_c)})$. Then $\gamma(K)$ consists of all configurations q such that there exists a configuration $q' \in K$, a real $\delta \in \mathbb{R}_{\geq 0}$ and a witness flow ρ for $q \xrightarrow{\delta} q'$ such that (a) for all $0 \leq t < \delta$, $\rho(t) \notin escape(K)$, and (b) $q' \in K \cup Pre(K, \Sigma_u)$. The

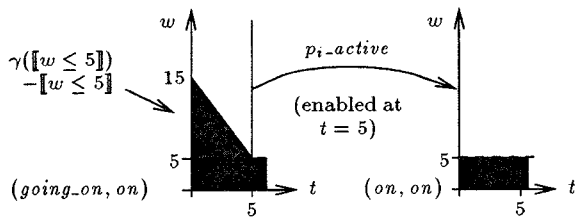

Figure 2: Unavoidable predecessors example

first condition requires the flow to avoid configurations from which a control strategy could divert the configuration into $\overline{K}$. The second condition requires that the ending configuration of the flow is either in K or has a successor in K via an uncontrollable jump step.

Example 3.1 The safety requirement for the steam boiler asserts $w > 5$. Unsafe configurations for part of the state space appear shaded in black in Figure 2. For simplicity, only the modes of the pumps are shown. For mode (*going_on*, *on*), the water flows with rate varying between -2 and 4 liters per second. The set $Pre([\![w \le 5]\!], \Sigma_u)$ is contained in $[\![w \le 5]\!]$. Hence it suffices to compute the uncontrollable predecessors due to controlled flows into the unsafe region $[\![w \le 5]\!]$. From mode (on, on), the flow is between 2 and 8 liters per second, so the only uncontrollable predecessors of $[\![w \le 5]\!]$ at this mode are the configurations that are themselves unsafe.

Theorem 3.1 *The set of controllable configurations of a LHA H subject to the safety requirement safe is the complement of the least fixpoint $\mu W.\overline{safe} \cup \gamma(W)$.*

Proof: A configuration q is *uncontrollable* iff the plant H has a strategy for visiting a configuration in $\overline{safe}$ iff the plant can force its configuration into $\overline{safe}$ within a finite number of steps. The set $\gamma(\overline{safe})$ consists of all configurations from which the plant can enter the set $\overline{safe}$ by a single flow step possibly followed immediately by a single uncontrollable jump step. Note that since the flow avoids all configurations in $escape(\overline{safe})$, it is impossible for any controller to issue a control command without the plant entering $\overline{safe}$. An inductive argument, together with monotonicity of γ, completes the proof. ■

While the above theorem characterizes the (maximal) set of controllable configurations, it does not imply the existence of a minimally restrictive controller. For example, consider a plant with a single control mode, a single variable x, subject to $\dot{x} = 1$, and a single mode switch with jump condition $x < 1 \wedge x' = 0$ and controllable event label. The initial condition is $x = 0$ and the safety requirement is $x < 1$. The set of controllable configurations is precisely the set of safe states. However, every legal controller must force the controllable event to occur at some point *before* x reaches 1, and hence not all configurations with $x < 1$ will be reachable.

It is possible to compute $\gamma(K)$ using primitive operations on linear regions. The apparent obstacle is the existence condition for witness flows into $K \cup Pre(K, \Sigma_u)$. The witness flows may swerve around regions of escape configurations.

Lemma 3.2 *There is a witness flow from q to q' that avoids configurations in the linear region E iff there exists a finite sequence of straightline witness flows for $q \xrightarrow{t_1} q_1 \xrightarrow{t_2} q_2 \xrightarrow{t_3} \cdots q'$, each avoiding E.*

Proof: The region E to avoid is linear, and its complement can be expressed as the union of finitely many convex regions $C_1, \ldots, C_m$ say. A witness flow ρ from q to q' avoiding E passes only through the regions $C_1, \ldots, C_m$. We can select a sequence of configurations corresponding to the last time ρ leaves each region C_i, and by convexity of the C_i and the flow conditions and the mean value theorem, we can construct straightline witness flows between them, each staying entirely within the C_i. ■

We define the operator $\gamma' : Regions(H) \to Regions(H)$ such that the definition of γ' differs from that of γ only in that the witness flows in the definition must also be straight lines.

Lemma 3.3 *Given a linear region K of a LHA H, both $escape(K)$ and $\gamma'(K)$ are linear and computable using Pre, a closure operation, and boolean operations over linear regions.*

Proof: For $escape(K)$, the result follows immediately from the definition. Let $escape(K)$ be the union of the convex sets $E_1, \ldots, E_n$, and $K \cup Pre(K, \Sigma_u)$ the union of the convex sets $A_1, \ldots, A_m$. Then the set $\gamma'(K)$ is given as $K \cup \bigcup_{\sigma_u \in \Sigma_u} Pre(K, \sigma_u) \cup \bigcup_{1 \le i \le m} \bigcap_{1 \le j \le n} flow\_avoid(A_m, E_n)$, where the set $flow\_avoid(A, E)$ is the set of configurations that are either in A or can flow into A along straightline witnesses while avoiding E. It can be shown that for convex sets A and E, the set $flow\_avoid(A, E)$ can be computed as the least fixpoint $\mu W.A \cup \bigcup_{\psi \in \Psi(E)} (\overline{E} \cap cl([\![\neg\psi]\!]) \cap Pre_t(W \cap [\![\neg\psi]\!]))$, where $cl(Y)$ is the closure of Y, $Pre_t(W)$ is $\bigcup_{\delta \in \mathbb{R}_{\ge 0}} Pre(W, \delta)$, and $\Psi(E)$ a set of linear inequalities such that $E = \bigcap_{\psi \in \Psi(E)} [\![\psi]\!]$. The idea is to construct $flow\_avoid(A, E)$ by iteratively adding configurations along flows that may reach but may not cross the extended faces $(cl([\![\psi]\!]) \cap cl([\![\neg\psi]\!]) \cap \overline{E})$ of E. Details are defered to the full paper. ■

Lemma 3.4 *Given a linear region Y for a LHA H, $\mu W.Y \cup \gamma(W) = \mu W.Y \cup \gamma'(W)$.*

Proof: Since $\gamma'(W) \subseteq \gamma(W)$, it follows that $\mu W.Y \cup \gamma'(W) \subseteq \mu W.Y \cup \gamma(W)$. For the reverse inclusion, first observe that the set of escape configurations for a linear region is itself a linear region by Lemma 3.3. Then by Lemma 3.2, configurations in $\gamma(W)$ are in $\mu Z.W \cup \gamma'(Z)$, and thus the inclusion follows. ■

Theorem 3.5 *The controller synthesis problem for LHAs with linear safety requirements is semidecidable.* ■

It is not difficult to see that controllers can be extracted from the maximal set of controllable configurations: control actions should only be taken at escape configurations, and they must be taken before the further passage of time could lead to an unsafe configuration. Termination of the synthesis procedure is not guaranteed; indeed even reachability for linear hybrid automata is already only semidecidable.

3.2 Partial observability

We have assumed so far that the controller has complete observability of the entire plant configuration. We now provide simple procedures to synthesize controllers under partial observability. We sacrifice completeness, *i.e.* the procedure may terminate without finding a controller, even though a controller does exist. In return, we maintain soundness and ease of computation. We assume that the controller views the plant configuration through an observability mask $vis : \mathcal{C} \to \mathcal{O}$, where $\mathcal{O}$ is the domain of controller observations of the configurations [6, 12]. A *partially observing* controller for a LHA H is a function $f : \mathcal{O} \to \Sigma_c \cup \{\lambda\}$, or equivalently a controller function $f' : \mathcal{C} \to \Sigma_c \cup \{\lambda\}$ such that $vis(q) = vis(q')$ implies $f'(q) = f'(q')$. For every observable configuration q_o and set P of potentially controllable configurations, let $Potential(q_o, P)$ be a set that contains all configurations in $vis^{-1}(q_o)$ that could be the real plant configuration when a legal controller sees q_o. Possible choices for $Potential(q_o, P)$ include (1) the entire set $vis^{-1}(q_o)$, (2) those configurations in $vis^{-1}(q_o)$ that are reachable in the uncontrolled plant H, and (3) the potentially controllable configurations $vis^{-1}(q_o) \cap P$. All three options can be computed using, at worst, reachability analysis over LHAs.

The set of *partial-observation escape* configurations for a set K via the event σ_c w.r.t. the mask vis and set P consists of all configurations q for which all configurations in $Potential(vis(q), P)$ are escape configurations for K via σ_c. The unavoidable predecessors operator $\gamma_o(vis)$ for the observability mask vis is defined as for γ except that escape configurations are replaced by partial observation escape configurations. The operator is computable using the same primitives required for computing γ. It is not difficult to see that if the least fixpoint $\mu W.safe \cup \gamma_o(vis)(W)$ includes the initial configurations, then a partially observing controller exists and can be extracted from the fixpoint.

4 Time divergence

The controllers synthesized in Section 3 could force trajectories to be safe by taking infinitely many control actions in a finite amount of time, thereby stopping time from progressing. Such a controller is clearly not realizable, and should be disallowed. As an extreme case, consider a plant with a single initial configuration. If the initial configuration is safe, and has a controllable jump returning to itself, then the controller that forces the jump to occur repeatedly at the initial state is legal, but nonsensical.

The *duration* of a step $q \xrightarrow{\sigma} q'$ for $\sigma \in \Sigma$ is 0, and the duration of a step $q \xrightarrow{\delta} q'$ for $\delta \in \mathbb{R}_{\geq 0}$ is δ. A trajectory is *divergent* iff the sum of the durations of the steps is unbounded. A LHA is *nonZeno* iff there is a divergent trajectory originating from every configuration that is reachable from an initial configuration. A controller C for a LHA H is *nonZeno* iff there is a divergent controlled trajectory originating from every configuration that is control-reachable from an initial configuration. A LHA H is *control-divergent* if every controller for H is nonZeno. We define a sufficient, but not necessary, condition for a LHA to be control-divergent. A LHA H is *conforming* iff the following three conditions hold:

1. *nonZenoness:* H is nonZeno,
2. *cycle δ-positivity:* There exists $\delta > 0$ such that every trajectory $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} q_2 \xrightarrow{e_3} \cdots q_m$ has duration no less than δ if q_0 and q_m share the same control mode and at least one of the events e_i belongs to Σ_c (such a trajectory is called a *controlled cycle*),
3. *control δ-positivity:* There is a $\delta' > 0$ such that for every controllable event σ_c and configuration q, if σ_c is enabled at q, then δ' time is also enabled at q.

The control δ-positivity condition implies that invariants never force control events to occur. In particular, if time gets "stuck" at an invariant boundary, then no control events are enabled.

Theorem 4.1 *Every conforming LHA H is control-divergent.*

Proof: Let C be a controller for H. Let q be a configuration that is control-reachable from an initial configuration of H. We show existence of a divergent controlled trajectory originating from q.

Case 1: Suppose that for all configurations q' control-reachable from q, there is no controllable $\sigma_c \in \Sigma_c$ that is enabled at q'. Then the set of all controlled trajectories originating from q is the set of all trajectories in H originating from q, since the controller C cannot prevent time from passing, nor any uncontrollable events from occurring. Since H is nonZeno, then q has a divergent trajectory.

Case 2: Suppose that there is a controlled trajectory from q to some configuration q' such that q' meets the requirement of Case 1. Then q has a divergent trajectory.

Case 3: Suppose otherwise, *i.e.* that from every configuration q' control-reachable from q there is some configuration q'' control-reachable from q' for which a controllable event is enabled in H. We exhibit a divergent controlled trajectory from q. First, follow a sequence of controlled steps to a configuration for which a controllable event is enabled in H. By control δ-positivity, either the controller C allows δ time to pass, or it forces a control event before δ time elapses. Our constructed trajectory follows the controller choice, leading to a new configuration q'. From q', it is possible to reach another configuration at which a controllable event is enabled. We can therefore construct a trajectory with either infinitely many controllable events or infinitely time steps of duration δ. By the cycle δ-positivity of H, the trajectory is divergent. ∎

Example 4.1 Many controllers issue instant control commands in response to sampling the system state every Δ time units. This control structure can be imposed by modifying the plant H to obtain a new automaton H': introduce a new variable t, and add the conjunct $\dot{t} = 1$ to every flow condition, the conjunct $t = 0$ to every initial condition, the conjunct $t \leq \Delta$ to every invariant condition, and the conjunct $t = \Delta \wedge t' = 0$ to the jump condition of every mode switch whose event label is in Σ_c. The synthesis of a sampling controller for H is solved by synthesizing a controller for H'.

	Observability	Nr. modes	Nr. itns	Time (s)
No sampling	Full	36	6	46
	Pumps, water	36	6	51
	control, water	36	7	61
Sampling	Full	44	5	152
	control, water	44	5	172
+valve	Full	52	6	222
+valve	control, water	52	6	321
Simplified [11]	Full	12	4	42
	control	12	4	73

Figure 3: Performance results for the steam boiler

Although H' is not control-divergent, it is easy to construct a control-divergent automaton A whose controllers are isomorphic to those of H'. The LHA A includes a slightly modified copy of H', together with a new control mode $\hat{v}$ of every control mode v. Every controllable mode switch $e = (v, v')$ in H' is removed and replaced by a control switch $(v, \hat{v})$ with an uncontrollable event label and the jump condition $t = \Delta$, and by a control switch $(\hat{v}, v')$ with the jump condition and event label of e. The invariant for each mode $\hat{v}$ is *True*. All configurations where $t > \Delta$ and the mode is some new mode $\hat{v}$ are considered unsafe. Thus in A, control events can only occur in the new vertices, and they must occur at time $t = \Delta$.

A similar construction can be used for generating nonZeno controllers that must allow a delay of *at least* Δ, and possibly more, between control events.

5 Conclusions

We presented symbolic procedures for synthesizing controllers for LHAs subject to safety requirements, under full and partial observability. Though not pursued here, the procedures can be extended to handle more general forms of requirement specifications by using the dual of our unavoidable predecessors operator as the controllable predecessors operator π in Maler et al.'s algorithms [13].

The synthesis procedures for full and partial observability have been implemented as an experimental module in the tool HyTech [9]. We report here on experiments performed on the steam boiler. We synthesized controllers with a fixed sampling rate of 5 seconds, as well as controllers that could continually monitor the plant configuration and issues control actions whenever they pleased. We also varied the observability of the controller, including observability of the entire plant configuration (full), observability of only a subset of the continuous variables (pumps, water), and observability of only the water and the control mode for a prespecified control structure (control, water). We also considered a simplified model of the boiler due to Heymann et al. [11]. Figure 3 shows the size (number of reachable control modes) of the plants, and performance data obtained on a Sun Sparcstation 5. Unless otherwise indicated, the valve automaton was omitted.

The early implementational results show that in practice the synthesis procedures are limited to control problems with only a small number of control modes. Improving efficiency can come from two means: better algorithms and data structures for implementing the unavoidable predecessors operator, and the consideration of special classes of automata for which more efficient synthesis procedures exist [10].

References

[1] J.-R. Abrial, E. Börger, and H. Langmaack. *Formal Methods for Industrial Applications*, LNCS 1165. Springer, 1996.

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

[3] R. Alur and T. A. Henzinger. Modularity for timed and hybrid systems. In *CONCUR 97: Concurrency Theory*, LNCS 1243, pp. 74–88. Springer, 1997.

[4] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In P. Antsaklis et al., ed., *Hybrid Systems II*, pp. 1–20. Springer, 1995.

[5] S. Balemi, G. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. Franklin. Supervisory control of a rapid thermal multiprocessor. *IEEE Trans. on Automatic Control*, 38(7):1040–1059, July 1993.

[6] R. Cieslak, C. Desclaux, A. S. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Trans. on Automatic Control*, 33(3):249–260, 1988.

[7] A. Göllü and P. Varaiya. Hybrid dynamical systems. In *Proc. of 28th IEEE Conf. on Decision and Control*, pp. 2708–2712, 1989.

[8] T. Henzinger. The theory of hybrid automata. In *Proc. of 11th IEEE Symposium on Logic in Computer Science*, pp. 278–292, 1996.

[9] T. Henzinger, P.-H. Ho and H. Wong-Toi. HyTech: a model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1). Springer, 1997.

[10] T. Henzinger and P. Kopke. Discrete-time control for rectangular hybrid automata. In P. Degano et al,. ed., *ICALP 97: Automata, Languages, and Programming*, LNCS 1256, pp. 582–593. Springer, 1997.

[11] M. Heymann, F. Lin, and G. Meyer. Control synthesis for a class of hybrid systems subject to configuration-based safety constraints. In O. Maler, ed., *HART 97: Hybrid and Real-Time Systems*, LNCS 1201, pp. 376–390. Springer, 1997.

[12] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(3):173–198, 1988.

[13] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In E. Mayr and C. Puech, ed., *STACS 95: Theoretical Aspects of Computer Science*, LNCS 900, pp. 229–242. Springer, 1995.

[14] A. S. Morse, editor. *Control using Logic-Based Switching*, LNCIS 222, Springer-Verlag, 1997.

[15] P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. of Control and Optimization*, 25(1):1202–1218, 1987.

[16] M. Tittus and B. Egardt. Control-law synthesis for linear hybrid systems. In *Proc. of 33rd IEEE Conf. on Decision and Control*, pp. 961–966, 1994.

[17] H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *Proc. of 30th IEEE Conf. on Decision and Control*, pp. 1527–1528, 1991.