Synthesis and Viability of Minimally Interventive Legal Controllers for Hybrid Systems*

Michael Heymann[1] Feng Lin[2] George Meyer[3]

[1]Department of Computer Science
Technion, Israel Institute of Technology
Haifa 32000, Israel
e-mail: heymann@cs.technion.ac.il

[2]Department of Electrical and Computer Engineering
Wayne State University
Detroit, MI 48202
e-mail: flin@ece.eng.wayne.edu

[3]NASA Ames Research Center
Moffett Field, CA 94035
e-mail: meyer@tarski.arc.nasa.gov

Abstract

In this paper, we study the control of *Composite Hybrid Machines* (CHMs) subject to safety specifications. CHMs are a fairly general class of hybrid systems modeled in modular fashion as the concurrent operation of *Elementary Hybrid Machines* (EHMs). The formalism has a well-defined synchronous-composition operation that permits the introduction of the controller as a component of the system. The task of a legal (safety) controller is to ensure that the system never exits a set of specified legal configurations. Among the legal controllers, we are particularly interested in designing a minimally-interventive (or minimally-restrictive) one, which interferes in the system's operation only when constraint violation is otherwise inevitable. Thus, a minimally interventive safety controller provides maximum flexibility in embedding additional controllers designed for other control objectives to operate concurrently, while eliminating the need to re-investigate or re-verify the legality of the composite controller with respect to the safety specification. We describe in detail an algorithm for controller synthesis and examine the viability of a synthesized controller as related to the possibility of Zenoness, where the system can undergo an unbounded number of transitions in a bounded length of time.

Keywords: Hybrid systems, safety control synthesis, Zenoness

*This research is supported in part by the National Science Foundation under grant ECS-9315344 and NASA under grant NAG2-1043 and in part by the Technion Fund for Promotion of Research. The work by the first author was completed while he was a Senior NRC Research Associate at NASA Ames Research Center, Moffett Field, CA 94035.

1 Introduction

1.1 Background and Objectives

Various formalism have been proposed in the literature to capture the intuitive idea that hybrid systems are dynamic systems in which discrete and continuous behaviors coexist and interact [3] [4] [10] [11] [28] [32]. Broadly speaking, hybrid systems are systems in which changes occur both in response to events that take place discretely, asynchronously and sometimes nondeterministically, and in response to dynamics that represents (causal) evolution as described by differential or difference equations of time. Thus, most physical systems that can be represented by formal behavior models are hybrid in nature.

In recent years there has been a rapidly growing interest in the computer-science community in modeling, analysis, formal specification and verification of hybrid systems (see, e.g. [4] [34]). This interest evolved progressively from logical systems, through "logically-timed" temporal systems [2] to real-time systems modeled as timed automata [29] and, most recently, to a restricted class of hybrid systems called *hybrid automata* [3] [28]. Thus, the computer-science viewpoint of hybrid systems can be characterized as one of discrete programs embedded in an "analog" environment.

In parallel, there has been growing interest in hybrid systems in the control-theory community [8] [10] [11], where traditionally systems have been viewed as "purely" dynamic systems that are modeled by differential or difference equations. More recently, control of purely discrete systems, modeled as discrete-event systems, also received attention in the literature [35] [36] [24]. The growing realization that neither the purely discrete nor the purely continuous frameworks are adequate for describing many physical systems, has been an increasing driving force to focus attention on hybrid systems. Contrary to the computer-science viewpoint that focuses interest in hybrid systems on issues of analysis and verification [31] [33], the control-theory viewpoint is to focus its interest on issues of design.

Typical hybrid systems interact with the environment both by sharing signals (i.e., by transmission of input/output data), and by event synchronization (through which the system is reconfigured and its structure modified). Control of hybrid systems can therefore be achieved by employing both interaction mechanisms simultaneously. Yet, while this flexibility adds significantly to the potential control capabilities, it clearly makes the problem of design much more difficult. Indeed, in view of the obvious complexity of hybrid control, even the question of what are tractable and achievable design objectives, is far from easy to resolve.

In the present paper we examine the control problem for a class of hybrid systems called *composite hybrid machines* (CHMs). These constitute hybrid systems consist of the concurrent operation of *elementary hybrid machines* (EHMs) using a well-defined synchronous composition formalism that allows both signal sharing and event synchronization. A controller can then be coupled with the plant by means of synchronous composition.

The goal of a legal controller considered in the present paper is to ensure the safety of the system in the sense that it will never violate its legal specification given by a set of (illegal) configurations that must be avoided. In other words, a *legal* controller must prevent the system from ever entering the illegal configurations. Among all legal controllers, we are interested in minimally interventive ones.

A legal controller is minimally interventive if, when composed to operate concurrently with any other controller, it will remain inactive except at the boundary of legal region where controller inaction would lead to inevitable safety violation. Therefore, such a controller can be composed to operate concurrently with any other controller that may be designed to achieve other requirements such as stability or optimality. There is no need to re-investigate or re-verify legality of the

composite controller with respect to safety.

We confine our attention to a special class of CHMs where system dynamics is *rate-bounded* and legal guards are conjunctions or disjunctions of atomic formulas specified by inequalities. We further confine our attention to a simplified control mechanism, where the controller is allowed to interact with the system only discretely; that is, the controller is permitted to trigger only discrete changes in the system.

We describe a detailed algorithm for synthesis of a minimally interventive legal controller (that prevents the system from ever entering a specified set of illegal configurations).

A major issue associated with hybrid systems, to which we devote attention in this paper, is the Zenoness phenomenon. Intuitively, a system is Zeno if it can undergo an unbounded number of discrete changes (transitions) in a bounded length of time. When a controlled CHM is Zeno, it cannot be guaranteed to satisfy the safety specification indefinitely, and hence may violate the basic *viability* requirement. The computational verification of non-Zenoness has been shown to be algorithmically a hard problem [6].

We show that when our controller synthesis algorithm terminates, then the synthesized controller is legal and minimally interventive if the resultant closed loop system is non-Zeno. To examine the non-Zenoness, we introduce two concepts: *instantaneous configuration clusters* (ICCs) and *hybrid attractors*. We then show that the system is non-Zeno if and only if it has no ICC that is a hybrid attractor.

1.2 Design Philosophy

Intuitively, a controller for legal behavior of a hybrid system is minimally interventive if it never takes action unless a (safety) constraint violation becomes imminent. When the latter happens, the controller is expected do no more than to prevent the system from becoming "illegal". This is a familiar setting in the discrete-event control literature since, there, the role of the controller has traditionally been viewed as that of a *supervisor* that can only intervene in the system's activity by event disablement [35] [24]. Thus, a minimally interventive supervisor of a discrete-event system is one that disables events only whenever their enablement would permit the system to violate the specification.

It is not difficult to see that a natural candidate for a "template" of a minimally interventive supervisor is a system whose range of possible behaviors coincides with the set of behaviors permitted by the specification. The concurrent execution of the controlled system and such a supervisor, in the sense that events are permitted to occur in the controlled system whenever they are possible in the controller template, would then constrain the system to satisfy the specification exactly. We shall then say that we have employed the specification as a candidate implementation. If all the events that are possible in the system but not permitted by the candidate supervisor can actually be disabled, we say that the specification is *implementable* or (when the specification is given as a legal language) *controllable* [35]. Generally, a specification may not be implementable because not all the events can be disabled.

The standard approach to supervisory controller synthesis can then be interpreted as an iterative procedure where, starting with the specification as a candidate implementation, at each stage of the iteration the specification is tightened so as to exclude behaviors that cannot be prevented from becoming illegal by instantaneous disablement of events [17] [18]. The *sub-specification* thus obtained, is then used as a new candidate implementation. When the procedure converges in a finite number of steps (a fact guaranteed in case the system is a finite automaton and the specification a regular language), the result is either an empty specification (meaning that a legal supervisor does not exist) or a minimally interventive implementable sub-specification.

In the present paper we shall employ the same design philosophy for the synthesis of minimally interventive controllers of hybrid systems. However, due to the addition of continuous dynamics and dynamic transitions caused by continuous dynamics, the synthesis problem for hybrid systems becomes much more complex. In particular, it is often necessary to "split" configurations into legal and illegal sub-configurations by considering some preemptive conditions that depend explicitly on continuous dynamics. Moreover, the existence of controllers for hybrid systems is, in general, an undecidable problem and the finite termination of the (fixed-point) synthesis algorithm is not guaranteed.

1.3 Comparison with Other Work

As stated before, the basic approach employed in our synthesis method is standard in the supervisory control theory of discrete-event systems, where a similar (least) fixed-point algorithm is usually employed (see, e.g., some of our own work on discrete-event systems [16] [17] [18] [19] [24] [25] [26] [12]). Needless to say, however, that there are significant differences between this work and that on supervisory control.

Our hybrid-machine formalism, while similar in spirit to the well-known hybrid automata model, (see, for example, [3]), differs from the latter in some subtle (but important) detail. Most importantly, we insist that vertices (and hence configurations) be always *completely guarded*, thereby insuring that CHMs are well-defined at every vertex (or configuration) in the sense that if an invariant becomes false, a dynamic transition is triggered. Furthermore, our model provides an explicit mechanism for interaction between EHMs by introducing input/output events and shared variables. Such an explicit mechanism is critical for controller specification and design as proposed in this paper.

Other works on control synthesis have recently been proposed for timed automata [37] [9], for hybrid automata [29] [38] [23] as well as for other formulations of hybrid systems [27]. While there are intersections of these works with ours, we present an explicit synthesis algorithm for minimally interventive legal controllers of hybrid machines. We also investigate the issue of viability of closed-loop systems, and we show that it essentially reduces to (non-)Zenoness of hybrid systems.

Another issue that has been investigated extensively in hybrid systems is that of the decidability of reachability [14] [15]. Since our goal is to design a legal controller that can prevent the system from reaching illegal states, we are more interested in the decidability of existence of such a controller (and hence the termination of our algorithm). Although this may be counter-intuitive, it can be shown that the class of hybrid systems for which reachability is decidable is incomparable with (neither containing nor contained in) the class of hybrid systems for which the existence of a legal controller is decidable. This latter issue will be discussed in detail elsewhere.

2 Hybrid Machines

We first introduce a modeling formalism for a class of hybrid systems which we call *hybrid machines*. We begin by an informal example.

2.1 Illustrative example

Figure 1 describes schematically a hybrid system that consists of a water-tank with water supplied by a pump and with outflow controlled by a two-position valve.

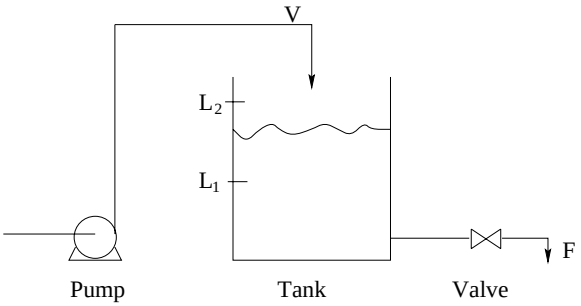

Figure 1: Water Tank System

The system is described graphically in Figure 2 as a *composite hybrid machine* (CHM) that consists of three *elementary hybrid machines* (EHMs) running in parallel:

$$CHM = Pump||Tank||Valve.$$

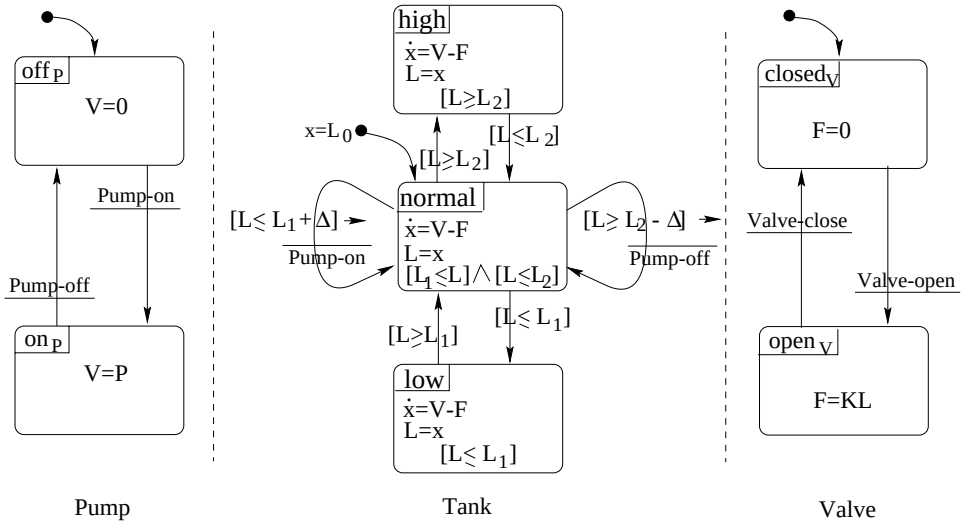

Figure 2: Water Tank System CHM

The EHM *Tank* has three *vertices* <high>, <normal> and <low>, representing the tank's "high" , "normal" and "low" levels , respectively. The dynamic behavior of the tank's water level L is described by the equations $\dot{x} = V - F, L = x$, where x is the (internal) state of the vertex, and V and F are the rates of water inflow and outflow, respectively. In this example, the (continuous) dynamic equations are the same at all three vertices. In general, however, they may be different. The quantity L is the *output-signal* of the EHM *Tank*. The quantities V and F constitute *input-signals* to the EHM *Tank* and output-signals of the EHMs *Pump* and *Valve*, respectively. *Tank* may reside at a given vertex provided the vertex invariant [.] is true. Thus, it may reside at the vertex <normal> so long as the invariant $[L_1 \leq L] \wedge [L \leq L_2]$ is satisfied, and similarly for the other vertex invariants. The transitions between the three vertices are *dynamic* in the sense that they are triggered, respectively, by the guards $[L \geq L_2]$, $[L \leq L_2]$, $[L \geq L_1]$ and $[L \leq L_1]$ becoming true. The self-loop dynamic transition of the vertex <normal> labeled by $[L \leq L_1 + \Delta] \rightarrow \overline{pump - on}$ is guarded by the predicate $[L \leq L_1 + \Delta]$ (where $\Delta > 0$ is some small constant), and upon occurrence triggers the *output-event* $\overline{pump - on}$. (Throughout, underlined event labels denote input-events and overlined

event labels denote output-events.) Similarly, the other self-loop transition of the vertex <normal> is guarded by $[L \geq L_2 - \Delta]$ and triggers the output-event $\overline{pump-off}$.The EHM *Tank* is initialized at the vertex <normal> with initial water level L_0 (that lies between the lower bound L_1 and the upper bound L_2).

The EHM *Pump* has two vertices: $< \text{off}_\text{P} >$ and $< \text{on}_\text{P} >$. At the vertex $< \text{off}_\text{P} >$, the pump is off, reflected by the vertex output $V = 0$. Similarly, at the vertex $< \text{on}_\text{P} >$, the pump is running and the vertex output V is the pump's (constant) flow rate P. The transitions between the two vertices are labeled by the *input-event* labels $\underline{pump-on}$ and $\underline{pump-off}$. These transitions are triggered by and take place concurrently and synchronously with the output-events $\overline{pump-on}$ and $\overline{pump-off}$, respectively.

Finally, the EHM *Valve* can be at either of the vertices $< \text{open}_\text{V} >$ or $< \text{closed}_\text{V} >$. Transition between the two vertices are labeled by input-events $\underline{valve-open}$ and $\underline{valve-closed}$, respectively. These transition labels do not appear as output-events in any of the other parallel EHMs but can be received from the (unmodeled) environment. When *Valve* is closed, the rate of outflow is $F = 0$ and when it is open, the rate is proportional to the water level in the tank $F = KL$.

Notice that in general there are two mechanisms for communication between parallel EHMs: (1) Input/output-event synchronization; by which transitions are synchronized. Transitions labeled by input-events can take place only in synchrony with a corresponding output-event that is being transmitted either by a parallel EHM or by the environment. (However, an output-event can be triggered without participation of any input-event, if no corresponding input-event is feasible.) (2) Signal sharing; by which outputs (output signals) of a vertex are available as vertex inputs to any other parallel EHM.

2.2 Elementary hybrid machines

With the above illustrative example in mind, we can now formally define hybrid machines as follows. An elementary hybrid machine is denoted by

$$EHM = (Q, \Sigma, D, I, E, (q_0, x_0)).$$

The elements of EHM are as follows.

- Q is a finite set of vertices.
- Σ is a finite set of event labels. An event is an input event, denoted by $\underline{\sigma}$ (underline), if it is received by the EHM from its environment; and an output event, denoted by $\overline{\sigma}$ (overline), if it is generated by the EHM and transmitted to the environment.
- $D = \{d_q = (x_q, y_q, u_q, f_q, h_q) : q \in Q\}$ is the dynamics of the EHM, where d_q, the dynamics at the vertex q, is given by:

  $$\dot{x}_q = f_q(x_q, u_q),$$
  $$y_q = h_q(x_q, u_q),$$

  with x_q, u_q, and y_q, respectively, the state, input, and output variables of appropriate dimensions. f_q is a Lipschitz continuous function and h_q a continuous function. (A vertex need not have dynamics associated with it, that is $d_q = \emptyset$, in which case we say that the vertex is *static*.) Note that the dynamics, and in particular the dimension of x_q, can change from vertex to vertex.

- $E = \{(q, G \wedge \underline{\sigma} \rightarrow \overline{\sigma'}, q', x^0_{q'}) : q, q' \in Q\}$ is a set of edges (transition-paths), where q is the vertex exited, q' is the vertex entered, $\underline{\sigma}$ is the input-event, $\overline{\sigma'}$ the output-event. G is the guard, formally given as a Boolean combination of inequalities (called atomic formulas) of the form $\Sigma_i a_i s_i \geq C_j$ or $\Sigma_i a_i s_i \leq C_j$, where the s_i are shared (signal) variables, to be defined in the next subsection, and the a_i and C_j are real constants. Finally, $x^0_{q'}$ is the initialization value for $x_{q'}$ upon entry to q'.

  $(q, G \wedge \underline{\sigma} \rightarrow \overline{\sigma'}, q', x^0_{q'})$ is interpreted as follows: If G is true and the event $\underline{\sigma}$ is received as an input, then the transition to q' takes place at the instant $\underline{\sigma}$ is received[1], with the assignment of the initial condition $x_{q'}(t_0) = x^0_{q'}$ (where t_0 denotes the time at which the vertex q' is entered and $x^0_{q'}$ is a constant (vector)). The output-event $\overline{\sigma'}$ is transmitted at the same time.

  If $\overline{\sigma'}$ is absent, then no output-event is transmitted. If $x^0_{q'}$ is absent (or partially absent), then the initial condition is inherited (or partially inherited) from x_q (assuming x_q and $x_{q'}$ represent the same physical object, and hence are of the same dimension). If $\underline{\sigma}$ is absent, then the transition takes place immediately upon G becoming true. Such a transition is called a *dynamic* transition and is sometimes abbreviated as (q, G, q') when $\overline{\sigma'}$ and $x^0_{q'}$ are either absent or understood. The guard associated with a dynamic transition is called a *dynamic* guard. If G is absent, the guard is always true and the transition will be triggered by the input-event $\underline{\sigma}$. Such a transition is called an *event transition* and is sometimes abbreviated as $(q, \underline{\sigma}, q')$ when $\overline{\sigma'}$ and $x^0_{q'}$ are either absent or understood. When both G and $\underline{\sigma}$ are present, the transition is called a *guarded event transition.*

- $I = \{I_q : q \in Q\}$ is a set of invariants. For each $q \in Q$, I_q is defined as $I_q = cl(\neg(G_1 \vee ... \vee G_k))$, where $G_1, ..., G_k$ are the dynamic guards at q, and where $cl(.)$ denotes set-closure[2].

- (q_0, x_0) denote the initialization condition: q_0 is the initial vertex and $x_{q_0}(t_0) = x_0$.

The invariant I_q of a configuration q expresses the condition under which the EHM is permitted to reside at q; that is, the condition under which none of the dynamic guards is true. In particular, from the definition of I_q as $I_q = cl(\neg(G_1 \vee ... \vee G_k))$, it follows that each of the vertices of the EHM is *completely guarded.* That is, every invariant violation implies that some dynamic guard becomes true, triggering a transition out of the current vertex[3]. (It is, in principle, permitted that more than one guard become true at the same instant. In such a case the transition that will actually take place is resolved nondeterministically.) It is further permitted that, upon entry into q', one (or more) of the dynamic guards at q' be already true. In such a case, the EHM will immediately exit q' and enter a vertex specified by (one of) the true guards. Such a transition is considered instantaneous.

The EHM runs as follows: At a vertex q, the continuous dynamics evolves according to d_q until either a dynamic transition is triggered by a dynamic guard becoming true, or an event transition is triggered by the environment (through an input event, while the associated guard is either absent or true).

[1] If $\underline{\sigma}$ is received as an input while G is false, then no transition is triggered.

[2] We shall always insist (especially during computations), that invariants and guards be derived as closed sets by taking their closure.

[3] Complete guardedness prevents the possibility that an invariant becomes false while no transition out of the current vertex is dynamically triggered. This possibility has not been precluded in the *hybrid automata* formalism as formulated e.g. in [6] [9], [38], leading to behaviors which have been termed there as *Zeno.* As will be discussed in more detail below, the Zeno behaviors as defined in the present paper consists only of a subset of the Zeno behaviors as defined there.

A *run* of the EHM is a sequence

$$q_0 \xrightarrow{e_1,t_1} q_1 \xrightarrow{e_2,t_2} q_2 \xrightarrow{e_3,t_3} ...$$

where e_i is the ith transition and $t_i (\geq t_{i-1})$ is the time when the ith transition takes place. For each run, we define its trajectory, path and trace as follows.

- The trajectory of the run is the sequence of the vector time functions of the (state) variables:

  $$x_{q_0}, x_{q_1}, x_{q_2}, ...$$

  where $x_{q_i} = \{x_{q_i}(t) : t \in [t_i, t_{i+1})\}$.
- The path of the run is the sequence of the vertices.
- The input trace of the run is the sequence of the input-events.
- The output trace of the run is the sequence of the output-events.

We say that a path is *irreducible* if for any two consecutive vertices q, q', in the sequence, either q and q' have different dynamics ($d_q \neq d_{q'}$), or upon entry into q', the state variable is (at least partially) re-initialized ($x^0_{q'} \neq \emptyset$). A run is irreducible if its associated path is irreducible.

To facilitate our ensuing exposition and, in particular, the synthesis in the next sections, we will standardize EHMs as follows. Recall that our model allows guarded event transitions of the form

$$q \xrightarrow{G \wedge \sigma} q'.$$

However, since for the transition to take place the guard must be true when the event is triggered, a guarded event transition can be decomposed into

$$q_1 \underset{\xleftarrow{\neg G}}{\xrightarrow{G}} q_2 \xrightarrow{\underline{\sigma}} q',$$

where q has been partitioned into q_1 and q_2, with $I_{q_1} = I_q \wedge \neg G$ and $I_{q_2} = I_q \wedge G$[4]. The dynamics of q_1 and q_2 and the transitions leaving and entering these vertices are the same as for q, except that the transition $(q_1, \underline{\sigma}, q')$ is now impossible. It follows that a guarded event transition can be treated as a combination of a dynamic and an event transition. Thus, in computations, we shall only need to consider two types of transitions: (1) dynamic transitions, that are labeled by guards only, and (2) event transitions, that are labeled by events only.

Remark 1 It is easily seen that discrete-event systems and continuous-variable systems are special cases of the hybrid systems as described above. Indeed, we notice that if there is no dynamics in an EHM (and hence no D and I), then

$$EHM = (Q, \Sigma, E, q_0)$$

where edges E are labeled only by events: a typical discrete-event system. Similarly, if there is no event and only one vertex in an EHM (and hence no need to introduce Q, Σ, I and E), then

$$EHM = (D, x_0) = (x, y, u, f, h, x_0),$$

which is a typical continuous-variable system.

[4]Since we use only closed invariants and guards, as described earlier, if I_{q_1}, I_{q_2} or $\neg G$ are not closed, we will take their closure.

In this paper we shall study a restricted class of hybrid machines called *bounded-rate* hybrid machines, characterized by the following assumption.

Assumption 1 The dynamics described by f_q and h_q has the following properties: (1) $h_q(x_q, u_q)$ is a linear function; and (2) $f_q(x_q, u_q)$ is bounded by a lower limit k_q^L and an upper limit k_q^U; that is, the only information given about $f_q(x_q, u_q)$ is that $f_q(x_q, u_q) \in [k_q^L, k_q^U]$.

2.3 Composite hybrid machines

A composite hybrid machine consists of several elementary hybrid machines running in parallel:

$$CHM = EHM^1||EHM^2||...||EHM^n.$$

Interaction between EHMs is achieved by means of signal transmission (shared variables) and input/output-event synchronization (message passing) as described below.

Shared variables consist of output signals from all EHMs as well as signals received from the environment. They are shared by all EHMs in the sense that they are accessible to all EHMs. A shared variable s_i can be the output of at most one EHM. The set of shared variables defines a signal space $S = [s_1, s_2, ..., s_m] \in \mathcal{R}^m$.

Transitions are synchronized by an input/output synchronization formalism. That is, if an output-event $\overline{\sigma}$ is either generated by one of the EHMs or received from the environment, then all EHMs for which $\underline{\sigma}$ is an active transition label (i.e., $\underline{\sigma}$ is defined at the current vertex with an absent guard or a true guard) will execute $\underline{\sigma}$ (and its associated transition) concurrently with the occurrence of $\overline{\sigma}$. A specific output-event can be generated by at most one EHM. Clearly, input-events do not synchronize among themselves[5].

To describe the behavior of

$$CHM = EHM^1||EHM^2||...||EHM^n,$$

we define a *configuration* of the CHM to be

$$q =< q_{i_1}^1, q_{i_2}^2, ..., q_{i_n}^n > \in Q^1 \times Q^2 \times ... \times Q^n,$$

where Q^j is the set of vertices of EHM^j (components of the EHMs are superscripted).

When all the elements of q are specified, we call q a *full* configuration. When only some of the elements of q are specified, we call q a *partial* configuration and we mean that an unspecified element can be any possible vertex of the respective EHM. For example, $< , q_{i_2}^2, ..., q_{i_n}^n >$ is interpreted as the set

$$<, q_{i_2}^2, ..., q_{i_n}^n >= \{< q_{i_1}^1, q_{i_2}^2, ..., q_{i_n}^n > : q_{i_1}^1 \in Q^1\}$$

of full configurations. Thus, a partial configuration is a compact description of a set of (full) configurations.

A transition

$$< q_{i_1}^1, q_{i_2}^2, ..., q_{i_n}^n > \xrightarrow{l} < q_{i'_1}^1, q_{i'_2}^2, ..., q_{i'_n}^n >$$

of a CHM is a triple, where $q =< q_{i_1}^1, q_{i_2}^2, ..., q_{i_n}^n >$ is the source configuration, $q' =< q_{i'_1}^1, q_{i'_2}^2, ..., q_{i'_n}^n >$ the target configuration, and l the label that triggers the transition. l can be either an event, or

[5]Notice that this formalism is a special case of the prioritized synchronous composition formalism [16], where each event is in the priority set of at most one parallel component.

a guard becoming true[6]. Thus, if $l = \underline{\sigma}$ is an event (generated by the environment), then either $q^j_{i'_j} = q^j_{i_j}$ if $\underline{\sigma}$ is not active at $q^j_{i_j}$, or $q^j_{i'_j}$ is such that $(q^j_{i_j}, \underline{\sigma} \rightarrow \overline{\sigma'}, q^j_{i'_j}, x^0_{q^j_{i'_j}})$ is a transition (edge) in E^j. On the other hand, if $l = G$ is a guard, then there must exist a transition $(q^m_{i_m}, G \rightarrow \overline{\sigma'}, q^m_{i'_m}, x^0_{q^m_{i'_m}})$ in some EHM^m, and for $j \neq m$, either $q^j_{i'_j} = q^j_{i_j}$ if $\underline{\sigma'}$ is not active at $q^j_{i_j}$, or $q^j_{i'_j}$ is such that $(q^j_{i_j}, \underline{\sigma'} \rightarrow \overline{\sigma''}, q^j_{i'_j}, x^0_{q^j_{i'_j}})$ is a transition in E^j. For brevity we shall sometimes denote the transition simply by (q, l, q'). Note that for simplicity, we do not specify the output events and initial conditions, since they are defined in the EHMs.

The transitions are assumed to occur instantaneously, and concurrent vertex changes in parallel components are assumed to occur exactly at the same instant (even when constituting a logically triggered finite chain of transitions).

Remark 2 Based on the above definition, a CHM can be viewed as the same object as an EHM:

$$CHM = (Q, \Sigma, D, I, E, (q_0, x_0))$$

where

$$\begin{array}{lcl}
Q & = & Q^1 \times Q^2 \times ... \times Q^n, \\
\Sigma & = & \Sigma^1 \cup \Sigma^2 \cup ... \cup \Sigma^n, \\
D & = & \{(x_q, y_q, u_q, f_q, h_q) : q =< q^1_{i_1}, q^2_{i_2}, ..., q^n_{i_n} > \in Q^1 \times Q^2 \times ... \times Q^n\} \\
 & & \text{combines all the dynamics of } q^j_{i_j}, j = 1, 2, ..., n, \\
I & = & \{I_{q^1_{i_1}} \wedge I_{q^2_{i_2}} \wedge ... \wedge I_{q^n_{i_n}} :< q^1_{i_1}, q^2_{i_2}, ..., q^n_{i_n} > \in Q^1 \times Q^2 \times ... \times Q^n\}, \\
E & & \text{is defined as above, } and \\
(q_0, x_0) & = & (< q^1_0, q^2_0, ..., q^n_0 >, (x^1_0, x^2_0, ..., x^n_0)).
\end{array}$$

Therefore, we can define a run of a CHM in the same way as that of an EHM. It can also be easily verified that in view of the fact that the component EHMs are completely guarded, so is the composite CHM.

3 Control

3.1 Specifications

As stated in the previous section, a CHM can interact with its environment in two ways: (1) by signal transmission (shared variables), and (2) by input/output-event synchronization. Formally, a *Controller* of a CHM is a hybrid machine C that runs in parallel with the CHM. The resultant system

$$CHM||C$$

is called the *controlled* or *closed-loop* system. The objective of control is to force the controlled system to satisfy a prescribed set of behavioral specifications.

For conventional (continuous) dynamical systems, control specification might consist of the requirement of stability, robustness, disturbance rejection, optimality and the like. For discrete-event systems, specifications of required behavior are typically given as *safety* specifications, where

[6]This follows from the decomposition of guarded event transitions into dynamic and event transitions as described in the preceding subsection.

a prescribed set of unwanted behaviors or configurations is to be avoided, or *liveness* specifications, where a prescribed set of termination conditions is to be met, or both.

For general hybrid systems, specifications can, in principle, be of a very complex nature incorporating both dynamic requirements and the logical (discrete) aspects.

In the present paper we consider only safety specifications given as a set of *illegal* configurations

$$Q_b = \{q =< q^1_{i_1}, q^2_{i_2}, ..., q^n_{i_n} >\in Q^1 \times Q^2 \times ... \times Q^n : q \text{ is illegal}\}$$

that the system is not permitted to visit.

As we stated previously, it is possible that guards of several dynamic transitions are true at the same time. When competing guards become true simultaneously, we shall give precedence to a legal guard (i.e., one that leads to a legal configuration) over an illegal one[7], and we shall resolve nondeterministically between competing legal guards.

Our goal is to synthesize a controller that guarantees satisfaction of the above stated configuration-based safety requirement. A controller that achieves the specification is then said to be *legal*.

In this paper, we shall consider only restricted interaction between the controller and the CHM by permitting the controller to communicate with the CHM only through input/output-event synchronization. Thus, we make the following assumption.

Assumption 2 C can only control the CHM by means of input/output-event synchronization. That is, C can only control event transitions in the CHM.

Thus, we assume that the CHM is "closed" in terms of its shared variables: Each shared variable is an output of one of its EHM. The controller does not generate any (dynamic) output signals that may affect the CHM.

We shall assume further that C can control all the event transitions in the CHM. That is, all the (externally triggered) event transitions are available to the controller. This leads to no essential loss of generality because, when some of the events are *uncontrollable*, we can use the methods developed in supervisory control of discrete-event systems [35] [36] to deal with uncontrollable event transitions.

A legal controller C is said to be *less interventive* (or restrictive) than another legal controller C' if every run permitted by C' is also permitted by C (a formal definition will be given in Section 5). A legal controller is said to be *minimally interventive* if it is less interventive than any legal controller.

With a slight modification of the formalism that we shall present in Section 5, two or more controllers can be combined by parallel composition to form a composite controller. An important characteristic of a minimally interventive controller is the fact that when it is combined with any other controller (legal or not), which is possibly designed for satisfying some other specifications, such as liveness, stability, or optimality, the combined controller is guaranteed to be safe (i.e., legal). Hence, no further verification of safety will be needed. Furthermore, the minimally interventive controller will intervene with the action of the other controller only minimally; that is, when it is absolutely necessary to do so in order to guarantee the safety of the system.

In the following synthesis, we shall assume that:

Assumption 3 The invariants of legal configurations are convex sets.

[7] That is, we shall assume that a transition to a legal configuration is triggered "just in time" before the system becomes illegal.

We shall also make the guards independent by considering only their (active) boundary with the invariant. Thus, since all the guards and invariants are closed, each guard G will be represented (i.e., replaced) by its active boundary

$$G_{new} = G \wedge I.$$

It will then follow that along any trajectory at most one (new) guard will ever become true. This will considerably facilitate the ensuing computations.

3.2 Control synthesis

As stated, our control objective is to ensure that the system CHM never enter the set of illegal configurations $Q_\flat$. Such entry can occur either via an event transition or via a dynamic transition. Since all event transitions are at the disposal of the controller, prevention of entry to the illegal set via event transitions is a trivial matter (they simply must not be triggered). Therefore, in our control synthesis we shall focus our attention on dynamic transitions. Intuitively, the minimally interventive legal controller must take action, by forcing the CHM from the current configuration to some other legal configuration, just in time (but as late as possible) to prevent a dynamic transition from leading the system to an illegal configuration. Clearly, entry to a configuration which is legal but at which an inescapable (unpreventable) dynamic transition to an illegal configuration is possible, must itself be deemed technically illegal and avoided by the controller. Thus the controller synthesis algorithm that we present below, will iterate through the (still) legal configurations and examine whether it is possible to prevent a dynamic transition from leading to an illegal configuration. In doing so, it will frequently be necessary to "split" configurations by partitioning their invariants into their *legal* and *illegal* parts.

If q_1' is the legal subconfiguration of a configuration q' whose invariant has been split into its legal part q_1' and its illegal part q_2', transition from a configuration q into q_1' (rather than into q_2') depends on satisfaction, upon entry into q', of the invariant $I_{q_1'}$ (rather than $I_{q_2'}$). Thus, let us define $wp(q,l,q')$ to be the weakest precondition under which the transition (q,l,q') will not violate the invariant $I_{q_1'}$ upon entry into q'. Since some of the shared variables that appear in $I_{q'}$ are possibly (re-)initialized upon entering q' because x_i is (re-)initialized, the condition $wp(q,l,q')$ can be computed from $I_{q_1'}$ by substituting into $I_{q_1'}$ the appropriate initial (entry) values of all the shared variables that are also output variables of q'. That is, if y_j is the jth output variable of q' and $s_i = y_j$ is a shared variable that appears in $I_{q'}$, then the value of s_i must be set to $s_i = h_j(x^0_{q'}, u_{q'})$.

Using this weakest precondition, we can replace each transition (q,l,q') by its equivalence $(q, wp(q,l,q') \wedge l, q')$. That is, a dynamic transition with guard G will be replaced by a dynamic transition with guard $G \wedge wp(q,G,q')$. Similarly, an event transition triggered by $\underline{\sigma}$ will be replaced by a guarded event transition $wp(q,\underline{\sigma},q') \wedge \underline{\sigma}$, which in turn will be decomposed into event and dynamic transitions.

Therefore, at the beginning of each iteration, we will normalize the CHM by performing the following steps:

1. Replace each transition $q \xrightarrow{l} q'$ by $q \xrightarrow{wp(q,l,q') \wedge l} q'$;
2. Decompose each guarded event transition $q \xrightarrow{G \wedge \sigma} q'$ into $q_1 \underset{\xleftarrow{\neg G}}{\xrightarrow{G}} q_2 \xrightarrow{\sigma} q'$;
3. Replace each invariant I by the closure of the negation of the disjunction of all the dynamic guards $cl(\neg(G_1 \vee ... \vee G_k))$;
4. Replace each guard G by its closure $cl(G)$;
5. Replace each guard G by its active boundary $G \wedge I$.

After this normalization, we can proceed to split invariants if necessary. To do this efficiently, we shall first consider the time at which a predicate will become true. Thus, let $T(P(x(t)))$ $(=T(true(P(x(t)))))$ be the time at which P becomes true along the trajectory $x(t)$.

Since our goal is to guarantee that the safety specification will not be violated under any condition, we must consider the maximum and minimum values of $T(P(x(t)))$ when evaluated over all possible trajectories of all possible runs. Thus, let us define

$$T_{max}(true(P)) = \max_{x(t)} T(P(x(t)))$$
$$T_{min}(true(P)) = \min_{x(t)} T(P(x(t))).$$

These maximum and minimum values can be calculated from the expression of P and the associated dynamics. For example, if P is atomic formula of the form

$$P{=}(s_i \geq C_i),$$

where C_i is some constant and s_i is a signal variable whose rate of change is bounded by $\dot{s}_i{\in}[{r_i}^L, {r_i}^U]$, then

$$T_{min}(true(P)) = \min_{s_i} T(s_i \geq C_i) = \begin{cases} 0 & \text{if } \overline{s_i} \geq C_i \\ (C_i - \overline{s_i})/{r_i}^U & \text{if } \overline{s_i} \leq C_i \wedge {r_i}^U > 0 \\ \infty & \text{otherwise,} \end{cases}$$

where $\overline{s_i}$ is the current value of s_i.

Let us now consider a legal configuration q. Since transitions leaving q are either dynamic transitions or event transitions, and can lead to either legal or illegal configurations, we classify them into four types:

1. Legal event transitions that lead to legal configurations:

$$ET_g(q, Q_b) = \{(q, \underline{\sigma}, q') : q \xrightarrow{\sigma} q' \wedge q' \notin Q_b\}.$$

2. Illegal event transitions that lead to illegal configurations:

$$ET_b(q, Q_b) = \{(q, \underline{\sigma}, q') : q \xrightarrow{\sigma} q' \wedge q' \in Q_b\}.$$

3. Legal dynamic transitions that lead to legal configurations:

$$DT_g(q, Q_b) = \{(q, G, q') : q \xrightarrow{G} q' \wedge q' \notin Q_b\}.$$

4. Illegal dynamic transitions that lead to illegal configurations:

$$DT_b(q, Q_b) = \{(q, G, q') : q \xrightarrow{G} q' \wedge q' \in Q_b\}.$$

If $ET_g(q, Q_b) \neq \emptyset$, then we can always safely exit q by forcing a transition $(q, \underline{\sigma}, q') \in ET_g(q, Q_b)$. Therefore, there is no need to split the invariant I_q.

If $ET_g(q, Q_b) = \emptyset$, then the transitions in $DT_b(q, Q_b)$ will be prevented from taking place, only if they are either preempted by some dynamic transitions in $DT_g(b, Q_b)$ or will never take place due to the dynamics at q.

Clearly, the dynamic transition $(q, G, q') \in DT_b(q, Q_b)$ will be preempted by another dynamic transition, provided I_q, the invariant of q, becomes false before G becomes true (recall our assumption that the invariant is violated if and only if one or more of the guards is true). The earliest time G will become true is $T_{min}(true(G))$ and the latest time I_q will become false is given by $T_{max}(false(I_q)) = T_{max}(true(\neg I_q))$. Therefore, to ensure that the transition (q, G, q') will not take place, it must be required that the following *preemptive condition*

$$pc(q, G, q') = ((T_{min}(true(G)) > T_{max}(false(I_q)))) \vee T_{min}(true(G)) = \infty$$

be satisfied.

To show that this preemptive condition is indeed the right condition, we prove the following proposition.

Proposition 1 The preemptive condition $pc(q, G, q')$ is true if and only if for any trajectory $x(t)$ in any run,

$$T(G(x(t))) > T(\neg I_q(x(t))) \vee T(G(x(t))) = \infty.$$

Proof

If $pc(q, G, q')$ is true, that is,

$$\min_{x(t)} T(G(x(t))) > \max_{x(t)} T(\neg I_q(x(t))),$$

or

$$\min_{x(t)} T(G(x(t))) = \infty,$$

then, clearly,

$$(\forall x(t)) T(G(x(t))) > T(\neg I_q(x(t))),$$

or

$$(\forall x(t)) T(G(x(t))) = \infty.$$

Therefore,

$$(\forall x(t)) T(G(x(t))) > T(\neg I_q(x(t))) \vee T(G(x(t))) = \infty.$$

On the other hand, if $pc(q, G, q')$ is not true, then

$$\min_{x(t)} T(G(x(t))) < \infty.$$

Therefore, there exists $x(t)$ such that $T(G(x(t))) < \infty$. Since by our assumption, along any trajectory at most one guard will ever become true, for the trajectory $x(t)$, we have

$$T(\neg I_q(x(t))) = T(G(x(t))).$$

Thus,

$$(\exists x(t)) T(G(x(t))) \leq T(\neg I_q(x(t))) \wedge T(G(x(t))) < \infty,$$

a contradiction.

■

By the above proposition, we will split the configuration q into two sub-configurations q_1 and q_2, by partitioning the invariant I_q as

$$\begin{aligned} I_{q_1} &= I_q \wedge pc(q, G, q') \\ I_{q_2} &= I_q \wedge \neg pc(q, G, q'). \end{aligned}$$

Clearly, the dynamics of q_1 and q_2 and the transitions leaving and entering these configurations are the same as for q, except that the transition (q_1, G, q') is now impossible. Also the transition from q_1 to q_2 is dynamic with the guard $\neg pc(q, G, q')$ and from q_2 to q_1 with guard $pc(q, G, q')$[8].

If there are more than one illegal dynamic transitions at q, then we will split q into q_1 and q_2 as follows.

$$\begin{aligned} I_{q_1} &= I_q \wedge (\wedge_{(q,G,q') \in DT_b(q,Q_b)} pc(q, G, q')) \\ I_{q_2} &= I_q \wedge \neg(\wedge_{(q,G,q') \in DT_b(q,Q_b)} pc(q, G, q')). \end{aligned}$$

Such splitting will be repeated until no further splitting is needed. After this procedure of splitting terminates, the surviving legal configuration (and its invariant) have the following property: Either all the possible dynamic transitions are legal or there exists at least one legal event transition that can be forced. Although the forcing can be done at any time when the CHM is in the corresponding configuration, the minimally interventive legal controller will not force the legal event transition unless it is absolutely necessary. Therefore, the transition will be forced at the boundary specified by the illegal dynamic guards.

$$\vee_{(q,G,q') \in DT_b(q,Q_b)} G.$$

By our assumption that legal guards have precedence over illegal guards, the forcing will have priority and take the CHM to safety.

From the above discussions, we can now formally describe our synthesis algorithm.

Algorithm 1 *(***Control Synthesis***)*
Input

- The model of the system

$$CHM = (Q, \Sigma, D, I, E, (q_0, x_0)).$$

- The set of illegal configurations $Q_b \subseteq Q$.

Output

- The controller

$$C = (Q^c, \Sigma^c, D^c, I^c, E^c, (q_0^c, x_0^c)).$$

Initialization

1. Set of bad configurations

$$BC := Q_b;$$

[8]Note that at the time of transition, the guards $\neg pc(q, G, q')$ and $pc(q, G, q')$, or more precisely, their closures, are both true. We do not need to worry about this because among other reasons, by employing control, we do not permit transitions to illegal configurations to actually take place.

2. Set of pending configurations

$$PC := Q - Q_b;$$

3. New set of pending configurations

$$NPC := \emptyset;$$

Iteration

4. For all $q \in PC$, $e = (q, l, q') \in E$ do

$$E := (E - \{e\}) \cup \{(q, wp(q, l, q') \wedge l, q')\};$$

5. Let

$$Repeat := false;$$

6. For all $q \in PC, e = (q, G \wedge \underline{\sigma}, q') \in E$ do
   begin

$$I_{q_1} := I_q \wedge \neg G;$$
$$I_{q_2} := I_q \wedge G;$$

   If $I_{q_1} = false$, then

$$E := (E - \{e\}) \cup \{(q, \underline{\sigma}, q')\};$$

   If $I_{q_2} = false$, then

$$E := E - \{e\};$$

   Else do
   begin

$$Repeat := true;$$
$$PC := (PC - \{q\}) \cup \{q_1, q_2\};$$
$$E := (E - \{e\}) \cup \{(q_1, G, q_2), (q_2, \neg G, q_1), (q_2, \underline{\sigma}, q')\};$$

   For all $e' = (q, l, q'') \in E - \{e\}$ do

$$E := (E - \{e'\}) \cup \{(q_1, l, q''), (q_2, l, q'')\};$$

   For all $e' = (q'', l, q) \in E$ do

$$E := (E - \{e'\}) \cup \{(q'', l, q_1), (q'', l, q_2)\};$$

   end;
   end;

7. If $Repeat = true$, go to 4;

8. For all $q \in PC$ do

$$I_q = cl(\neg \vee_{(q,G,q') \in E} G);$$

9. For all $(q, G, q') \in E$ do

$$G := cl(G) \wedge I_q;$$

10. For all $q \in PC$ do

begin

If $(\exists q' \in PC)(\exists (q, \underline{\sigma}, q') \in E)$, then

$$NPC := NPC \cup \{q\};$$

Else do

begin

$$H := \wedge_{(q,G,q') \in DT_b(q,BC)} pc(q, G, q');$$
$$I_{q_1} := I_q \wedge H;$$
$$I_{q_2} := I_q \wedge \neg H;$$

If $I_{q_1} = false$, then

$$BC := BC \cup \{q\};$$

If $I_{q_2} = false$, then

$$NPC := NPC \cup \{q\};$$

Else do

begin

$$BC := BC \cup \{q_2\};$$
$$NPC := NPC \cup \{q_1\};$$
$$E := E \cup \{(q_1, \neg H, q_2), (q_2, H, q_1)\};$$

For all $e = (q, l, q') \in E - DT_b(q, BC)$ do

$$E := (E - \{e\}) \cup \{(q_1, l, q'), (q_2, l, q')\};$$

For all $e = (q, l, q') \in DT_b(q, BC)$ do

$$E := (E - \{e\}) \cup \{(q_2, l, q')\};$$

For all $e = (q', l, q) \in E$ do

$$E := (E - \{e\}) \cup \{(q', l, q_1), (q', l, q_2)\};$$

11. If $PC \neq NPC$, then

$$PC := NPC;$$
$$NPC := \emptyset;$$

Go to 4;

Construction of C

12. Define vertices, events, dynamics and invariants:

$$Q^c := PC;$$
$$\Sigma^c := \Sigma \cup \{\underline{\tilde{\sigma}} : \sigma \in \Sigma\};$$
$$D^c := \emptyset;$$
$$I^c := I|_{Q_c};$$

13. Define transitions:

$$E^c := \{(q, \underline{\tilde{\sigma}} \rightarrow \overline{\sigma}, q') : q, q' \in Q^c \wedge (q, \underline{\sigma}, q') \in E\};$$
$$E^c := E_c \cup \{(q, \vee_{(q,G,q') \in DT_b(q,BC)} G \rightarrow \overline{\sigma}, q') : q, q' \in Q^c \wedge (q, \underline{\sigma}, q') \in E\};$$

14. End.

It is readily seen that the configurations of the controller C consist of the set of all "good" configurations with their invariants as calculated during the iteration phase of the algorithm. The controller C has no continuous dynamics, so it is "driven" by the dynamics of the CHM. The transitions of C are then triggered when the boundary of some illegal dynamic transitions is reached. The controller thus synthesized is minimally interventive. Its interaction with the system is restricted to the exclusive objective of preventing safety violation. The controller is augmented to allow "environment-triggered" transitions labeled by $\underline{\tilde{\sigma}}$, which are allowed to be generated by the environment (possibly by an additional controller) and trigger transitions in C and hence in the CHM whenever such transitions are not preempted by C. A preemption will occur only if otherwise a safety constraint cound be violated. We will illustrate the algorithm by the following train-gate example.

3.3 A Train-Gate Example

Let us consider an example of trains passing a railway gate. The uncontrolled system can be described as a CHM that consists of two EHMs running in parallel:

$$CHM = Gate||Train.$$

The EHM *Gate* is described graphically in Figure 3.

It has four vertices <open>, <lowering>, <closed>, and <raising>, representing the gate open, being lowered, closed, and being raised, respectively. The dynamic behavior of the gate at <lowering> and <raising> is described by $\dot{x} = -9, g = x$ and $\dot{x} = 9, g = x$ respectively, where x is the (internal) state variable of the vertex, and g is the output-signal, representing the angle of the gate (in degrees). From this dynamics, we know that it takes 10 seconds for the gate to open or close. Similarly, the dynamics at <open> and <closed> are described by $\dot{x} = 0, g = x$. The input event $\underline{\textit{lower}}$, for instance, takes *Gate* from <open> to <lowering>. *Gate* can stay in

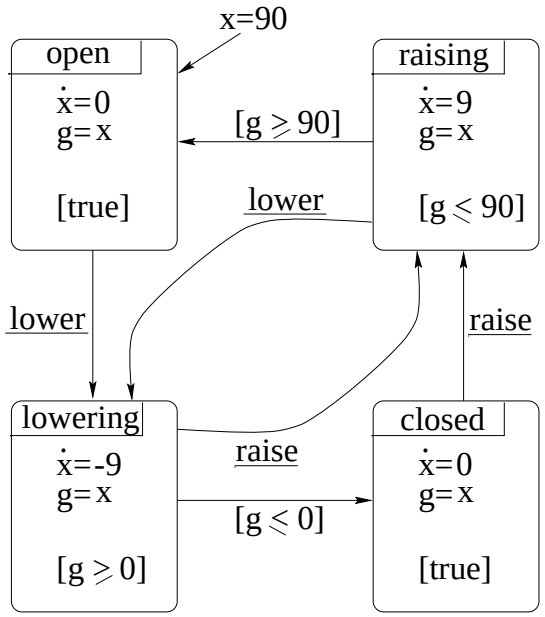

Figure 3: Gate

<lowering> as long as the invariant at <lowering>, $[g \geq 0]$, is satisfied. When the guard $[g \leq 0]$ becomes true, *Gate* moves from <lowering> to <closed>. The invariant at <open> and <closed> is $[true]$, meaning that *Gate* can stay in <open> and <closed> for ever, if no event is triggered. Initially, *Gate* is at <open> with $x(0) = 90$.

Similarly, the EHM $Train$ is shown in Figure 4.

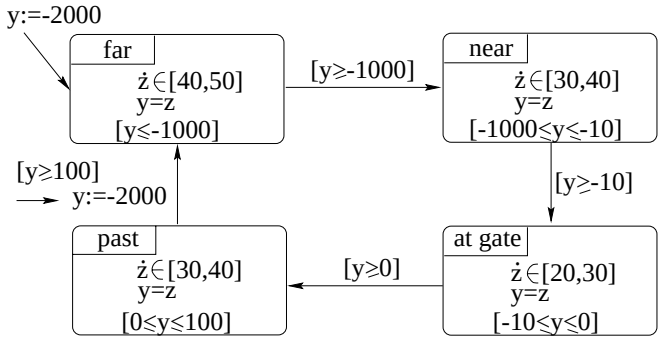

Figure 4: Train

It has four vertices <far>, <near>, <at gate>, and <past>, representing a train being far from, near to, at, and past the gate, respectively. The dynamic behavior of $Train$ is specified by $\dot{z} \in [r^L, r^U]; y = z$, representing the fact that the speed of a train is uncertain and we only know its lower bound r^L and upper bound r^U. In our simple model, when a train passes the gate, the EHM $Train$ is reset to <far> for the next approaching train. This is described by the transition $[y > 100] \rightarrow z = -2000$ from <past> to <far>.

Let us now apply the synthesis algorithm to the train-gate example. The CHM consisting of

Gate and $Train$ has 16 configurations. They are:

$q_1 =< open, far >$ $q_7 =< lowering, at >$ $q_{12} =< closed, past >$
$q_2 =< open, near >$ $q_8 =< lowering, past >$ $q_{13} =< raising, far >$
$q_3 =< open, at >$ $q_9 =< closed, far >$ $q_{14} =< raising, near >$
$q_4 =< open, past >$ $q_{10} =< closed, near >$ $q_{15} =< raising, at >$
$q_5 =< lowering, far >$ $q_{11} =< closed, at >$ $q_{16} =< raising, past >$
$q_6 =< lowering, near >$

Among these configurations, q_3, q_7, and q_{15} are illegal, because they represent the situation that the train passes the gate while the gate is not closed. Since there are dynamic transitions leading to these illegal configurations, the synthesis algorithm will split some remaining configurations by iteration.

Let us consider the first iteration. Since in the original CHM, The weakest preconditions are all satisfied and there are no guarded event transitions, Steps 4-7 will do nothing. Step 8 will also do nothing because all invariants are closed originally. Step 9 will replace guards by their active boundary. For example, $(q_6, [y \geq -10], q_7)$ will be replaced by $(q_6, [y = -10] \wedge [g \geq 0], q_7)$. Step 10 will split two configurations q_6 and q_{14}. For q_6, since there is one bad dynamic transition $(q_6, [y = -10] \wedge [g \geq 0], q_7) \in DT_b(q_6, BC)$ leaving q_6, we can calculate H as

$$\begin{aligned} H \quad & \Leftrightarrow pc(q_6, [y = -10] \wedge [g \geq 0], q_7) \\ & \Leftrightarrow T_{min}(true([y = -10] \wedge [g \geq 0])) > T_{max}(false([-1000 \leq y \leq -10] \wedge [g \geq 0]) \\ & \Leftrightarrow -\tfrac{y+10}{40} > \tfrac{g}{9} \\ & \Leftrightarrow y < -10 - \tfrac{40g}{9}. \end{aligned}$$

Therefore, q_6 is split into

$$I_{q_{61}} = I_{q_6} \wedge H = [-1000 \leq y < -10 - \tfrac{40g}{9}] \wedge [g \geq 0]$$
$$I_{q_{62}} = I_{q_6} \wedge \neg H = [-10 - \tfrac{40g}{9} \leq y \leq -10] \wedge [g \geq 0].$$

Similarly,

$$I_{q_{141}} = I_{q_6} \wedge H = [-1000 \leq y < -410 + \tfrac{40g}{9}] \wedge [g \leq 90]$$
$$I_{q_{142}} = I_{q_6} \wedge \neg H = [-410 + \tfrac{40g}{9} \leq y \leq -10] \wedge [g \leq 90].$$

The closures of the above invariants will be taken in the second iteration. Then q_2 will be split in the second iteration as

$$I_{q_{21}} = [-1000 \leq y < -410]$$
$$I_{q_{22}} = [-410 \leq y < -10].$$

The full computation of the first two iterations is summarized in Table 1. The minimally interventive legal controller is shown in Figure 5, where all the transitions are copies of transitions in the CHM, except the transition from q_2 to q_6 labeled by $[y \geq -410] \rightarrow \overline{lower}$, indicating that the latest legal closing of the gate must commence when the train is 410 (meters) away. (Since our controller is concerned only with safety, there is no controller command to raise the gate.)

	initialization	1st iteration	2nd iteration
q_1	[y≤-1000]	[y≤-1000]	[y≤-1000]
q_2	[-1000≤y≤-10]	[-1000≤y≤-10]	[-1000≤y<-410]
q_4	[0≤y≤100]	[0≤y≤100]	[0≤y≤100]
q_5	[y≤-1000]∧[g≥0]	[y≤-1000]∧[g≥0]	[y≤-1000]∧[g≥0]
q_6	[-1000≤y≤-10]∧[g≥0]	[-1000≤y<-10-40g/9]∧[g≥0]	[-1000≤y≤-10-40g/9]∧[g≥0]
q_8	[0≤y≤100]∧[g≥0]	[0≤y≤100]∧[g≥0]	[0≤y≤100]∧[g≥0]
q_9	[y≤-1000]	[y≤-1000]	[y≤-1000]
q_{10}	[-1000≤y≤-10]	[-1000≤y≤-10]	[-1000≤y<-10]
q_{11}	[-10≤y≤0]	[-10≤y≤0]	[-10≤y≤0]
q_{12}	[0≤y≤100]	[0≤y≤100]	[0≤y≤100]
q_{13}	[y≤-1000]∧[g≤90]	[y≤-1000]∧[g≤90]	[y≤-1000]∧[g≤90]
q_{14}	[-1000≤y≤-10]∧[g≤90]	[-1000≤y<-410+40g/9]∧[g≤90]	[-1000≤y≤-410+40g/9]∧[g≤90]
q_{16}	[0≤y≤100]∧[g≤90]	[0≤y≤100]∧[g≤90]	[0≤y≤100]∧[g≤90]

Table 1: Controller synthesis

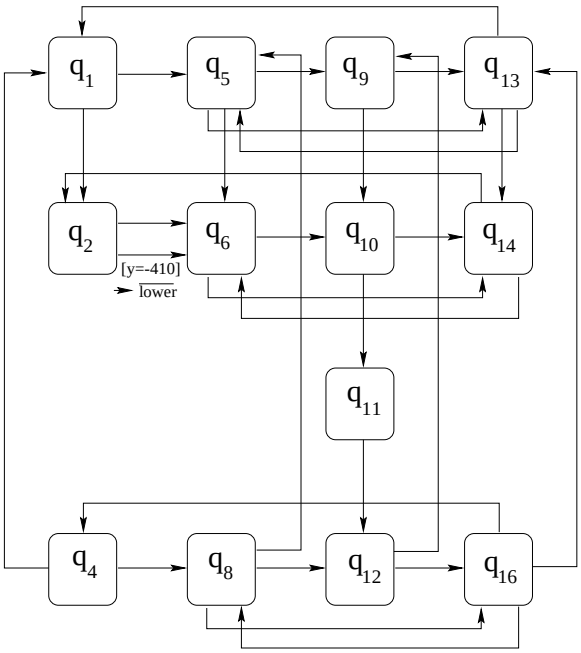

Figure 5: Controller

4 Zenoness

We shall call a run of a CHM *dynamic* if all its transitions are dynamic transitions. If a dynamic run is reducible, i.e., if its associated path has consecutive configurations q and q' with identical dynamics and no re-initialization upon transition from q to q', the run can be reduced by combining q and q' into a single configuration. Thus, every dynamic run can be reduced to an irreducible one. An unbounded irreducible dynamic run

$$q_0 \xrightarrow{e_1,t_1} q_1 \xrightarrow{e_2,t_2} q_2 \xrightarrow{e_3,t_3} ...$$

is called a *Zeno* run if

$$lim_{i\to\infty} t_i = T < \infty$$

A CHM is called *Zeno* if it possesses Zeno runs. Otherwise it is called *viable* or *non-Zeno*.

Clearly Zeno CHMs are ill defined, in that they may uncontrollably execute an unbounded number of transitions in a finite (and bounded) time interval and thus describe systems whose lifetime is limited, contrary to our intention of modeling ongoing behaviors (that never terminate). To illustrate the Zeno phenomenon, let us examine the following example.

Example 1 Consider the hybrid system modeled by the CHM shown in Figure 6, where k and l are constant real numbers.

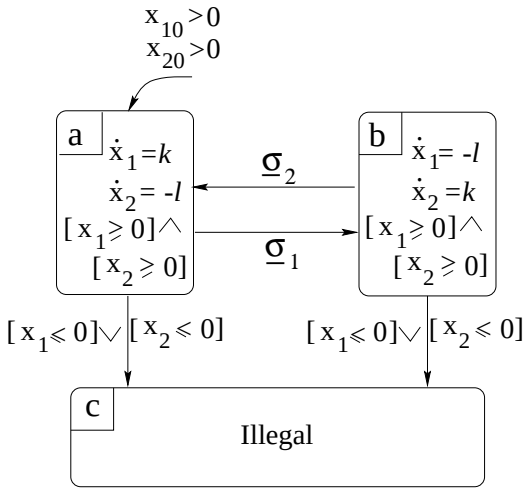

Figure 6: CHM of Example 1

It models a two-tank water system, where both tanks are leaking with rate l. A pump fills one of the tanks at rate $k+l$, and it can be switched between the two tanks (events $\underline{\sigma_1}$ and $\underline{\sigma_2}$). The system starts with both tanks non-empty ($x_1(0) = x_{10} > 0, x_2(0) = x_{20} > 0$). The system becomes illegal if and when the level in one of the tanks gets to be below zero. This is represented by a transition to the illegal configuration c that has no dynamics and invariant "true". It is readily seen that the rate of change of the total volume of water in the system is independent of the pump-switching policy and is given by k-l. Thus, if k-$l < 0$, then at time no later that $T = (x_{10} + x_{20})/(k\text{-}l)$ the total volume of water will become zero, so that the system must have become illegal no later than T. On the other hand, if k-$l \geq 0$, the volume of water does not diminish and, as easily see, a viable controller actually exist.

	a	b
initial	$[x_1 \geq 0] \wedge [x_2 \geq 0]$	$[x_1 \geq 0] \wedge [x_2 \geq 0]$
1st	$[x_1 \geq 0] \wedge [x_2 \geq 0]$	$[x_1 \geq 0] \wedge [x_2 \geq 0]$

Table 2: Controller synthesis of Example 1

The controller synthesis algorithm can be carried out for arbitrary k and l, and as shown in Table 2 it terminates after just one step.

The (augmented) controller C generated by the algorithm is shown in Figure 7. The guards are given as follows

$$G_1 = G_2 = [x_1 = 0] \vee [x_2 = 0].$$

Since at configuration a, $x_1 > 0$, the switching to configuration b actually occur at $x_2 = 0$. In other words, the controller switches the pump to a tank whose level reaches 0. If $k - l < 0$, then the switching becomes faster and faster, with infinite switching rate occurring after finite time. In other words, the closed-loop system $CHM||C$ is Zeno.

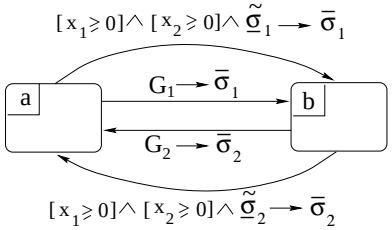

Figure 7: Controller of Example 1

To examine the Zenoness phenomenon and its relation to control synthesis, we begin by introducing the concept of an *instantaneous configuration cluster* (ICC). Let $v = [s_1, ..., s_m] \in S$ be a valuation of the signal vector and let q be a configuration. Suppose that q is entered by a dynamic transition G whose value is true at v. Assume further that q has an outgoing guarded transition G' which becomes (or is) true at the entry value of the signal vector q. (This value will be v if the signal vector is not re-initialized via a re-initialization of state variables, or it will be $v' \in S$ if it is re-initialized to this value upon entry to q.) Since G' follows G instantaneously, we say that the transition associated with G' is triggered by that associated with G. We say that a sequence of transitions $G_1, G_2, G_3, ...$ is triggered by v if G_1 is true at v and G_{i+1} is triggered by G_i for all $i \geq 1$. For a signal value v, consider all transition sequences in the CHM triggered by v. Let CHM(v) denote the CHM obtained by deleting all transitions that are not elements of transition sequences triggered by v. A strongly connected component[9] of CHM(v) that consists of two or more configurations is called an ICC. The triggering value v of the signal vector will be called a *Zeno point* of the CHM. We emphasize that an ICC consists of a set of configurations, the transitions and the associated triggering value v of the signal vector. In the two leaking tank example, $v = [0, 0]$ is a

[9] A strongly connected component is a set of configurations for which there is a directed path from any configuration to any other.

Zeno point associated with an ICC which includes the configurations $(a,a),(b,b)$ of the controlled system $CHM||C$.

The following theorem gives a necessary condition for Zenoness of a CHM.

Theorem 1 If a CHM is Zeno, then there exists an instantaneous configuration-cluster.

Proof

Assume that we have a Zeno CHM. Then there exists a Zeno run

$$q_0 \xrightarrow{e_1,t_1} q_1 \xrightarrow{e_2,t_2} q_2 \xrightarrow{e_3,t_3} \ldots$$

such that $t_i \to T$ as $i \to \infty$, where T is a finite constant. Since there are infinitely many transitions in finite time, the interval between two successive transitions approaches zero. Consider i sufficiently large, let G_i be the guard of e_i and let v_i be the value of the signal vector upon entering q_i. Since $t_{i+1} - t_i \to 0$, and since the rates of changes of all signals are finitely bounded, either $v_{i+1} - v_i \to 0$, or v_i is re-initialized to v_{i+1} (which is a constant for a given transition). In the prior case, since the invariants are closed, there must exists a limit point $\overline{v}$ in the invariant of q for which G_i and G_{i+1} are both true. In the latter case, G_{i+1} must be true at v_{i+1} for otherwise a bounded (below) amount of time must elapse before G_{i+1} becomes true after re-initialization to v_{i+1}, contradicting the time convergence of the Zeno run. Therefore, in both cases, G_{i+1} can be (instantaneously) triggered by G_i, proving the existence of some ICC.

■

The existence of an ICC does not in itself imply Zenoness. In the two leaking tanks of Example 1, for instance, if $k - l > 0$ the closed-loop system is non-Zeno, although $v = [0,0]$ is a Zeno point associated with an ICC.

To obtain a necessary and sufficient condition for Zenoness, we shall now refine our examination of ICCs. Clearly, once at an ICC, the behavior of the CHM is necessarily Zeno. Thus, the question that must be examined is, whether if initialized outside (or away from) an ICC, a possible run will enter the ICC after a bounded length of time. We shall say that an ICC is a *hybrid attractor* whenever there exist initializations of the CHM outside the ICC such that for some run, the ICC will be reached in bounded time.

Theorem 2 A CHM is non-Zeno if and only if its initialization set does not intersect an ICC and it has no hybrid attractor.

Proof

Elementary.

■.

Thus, the problem of checking Zenoness of a CHM (or, in particular, a closed-loop system) consists of identifying its ICCs, if any, and checking whether they include hybrid attractors. The detailed investigation of these issues will be presented elsewhere.

5 Correctness of the Algorithm

It is clear from Algorithm 1 that whenever a controlled systems $CHM||C$ undergoes a transition, it moves from a legal configuration to another. Thus, there remain two questions with respect to the correctness of the algorithm. The first is whether the controlled system is safe and viable. Since the safety is clear, this boils down to the question of non-Zenoness, which was discussed in the previous section. The second question is whether the synthesized controller is minimally

interventive. To this end, we define the conjunction of C with another controller D as follows. First, all the output-events $\overline{\sigma}$ in D are replaced by $\overline{\tilde{\sigma}}$ to obtain $\tilde{D}$. Then the composite controlled system is given by

$$CHM||C||\tilde{D}.$$

Theorem 3 If Algorithm 1 terminates in a finite number of steps and the closed-loop system $CHM||C$ is non-Zeno, then the controller synthesized is a minimally interventive legal controller in the following sense.

1. For any controller D (legal or not), a run of $CHM||C||\tilde{D}$ never visits illegal configurations in Q_b.
2. For any legal controller D, every run of $CHM||D$ has a corresponding run in $CHM||C||\tilde{D}$.

Proof

Since Algorithm 1 terminates in a finite number of steps and the closed-loop system $CHM||C$ is non-Zeno, it is safe and viable.

Notice that it is possible that the closed-loop system $CHM||C||\tilde{D}$ may generate a Zeno run due to $\tilde{D}$. Although such an ill-defined controller D should be avoided in practice, the correctness of C will not be affected.

To prove part 1, it is sufficient to show that a run in $CHM||C||\tilde{D}$ will only visit configurations in

$$Q^c \subseteq Q - Q_b.$$

If this is not the case, then there exists a run

$$q_0 \stackrel{e_1,t_1}{\longrightarrow} q_1 \longrightarrow ... \longrightarrow q_{n-1} \stackrel{e_n,t_n}{\longrightarrow} q_n$$

such that $q_0, q_1, ..., q_{n-1} \in Q^c$ but $q_n \notin Q^c$.

Let us consider the transition from q_{n-1} to q_n. It cannot be an event transition because such illegal event transitions are not permitted by C. If it is a dynamic transition, then since it is not preempted at q_{n-1}, it implies that $q_{n-1} \notin Q^c$, a contradiction.

To prove part 2, observe first that in view of the fact that Algorithm 1 progressively removes illegal behaviors, a controller will be legal only if it does not exceed the configurations and invariants of C. Assume that

$$q_0 \stackrel{e_1,t_1}{\longrightarrow} q_1 \longrightarrow ... \longrightarrow q_{n-1} \stackrel{e_n,t_n}{\longrightarrow} q_n$$

is a possible run of $CHM||D$ and the first $n-1$ transitions are also possible in $CHM||C||\tilde{D}$ but the last transition from q_{n-1} to q_n is impossible in $CHM||C||\tilde{D}$, that is, it is preempted by C. Since C only takes action at the boundary of some illegal dynamic transitions, the inaction of D at that point implies that for some trajectory associated with some continuation of this run, the invariant of C will be violated, contradicting the hypothesis that D is legal.

■

An important consequence of Theorem 3 is the fact that if the CHM can only execute transitions at a bounded finite rate (that is, not faster than at a given finite rate) then the termination of the algorithm implies that the synthesized controller is legal and minimally interventive. This is clear because in that case, there can be no ICCs in the controlled CHM. This will be the case if, for example, the controller can interact with the CHM only at discrete times (finite sampling rate) or if the CHM responds to the controller's actions with a finite delay. Thus we have the following immediate corollary to Theorem 3:

Corollary 1 If the CHM can execute transitions only at a finitely bounded rate and if Algorithm 1 terminates in a finite number of steps, then the controller synthesized is a minimally interventive legal controller.

6 Conclusion

In this paper we introduced composite hybrid machines (CHMs) as a formalism for modeling a class of hybrid systems that can interact by signal transmission and event synchronization. We presented an algorithm for synthesis of a minimally interventive legal controller for a CHM that interacts with the CHM discretely (i.e., triggers transitions in the CHM but does not interact with the signals). While the existence of a controller is in general undecidable, we show that when our algorithm terminates finitely, the existence of minimally interventive controller (and hence the validity of the controller synthesized) depends on whether the controlled (closed loop) system is non-Zeno. The latter is guaranteed if there are no instantaneous configuration clusters that are hybrid attractors. The synthesis of minimally interventive controllers that guarantee both safety as well as weak and strong liveness will be discussed elsewhere.

Acknowledgement. The authors express their thanks to Howard Wong-Toi for pointing out some subtle errors in an earlier formulation of CHMs and for very interesting discussions about control of hybrid systems.

References

[1] J.-R. Abrial, 1995. Steam-boiler control specification problem. *Dagstuhl Meeting: Method for Semantics and Specification.*

[2] R. Alur and D. Dill, 1990. Automata for modeling real-time systems. *Proc. of the 17th International Colloquium on Automata, Languages and Programming*, pp. 322-336.

[3] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, 1993. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. *Hybrid Systems, Lecture Notes in Computer Science, 736*, Springer-Verlag, pp. 209-229.

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

[5] R. Alur, T. A. Henzinger, E. Sontag, (Eds.) 1996. *Hybrid Systems III, Verification and Control, Lecture Notes in Computer Science, 1066*, Springer Verlag.

[6] R. Alur and T.A. Henzinger, 1997. Modularity of timed and hybrid systems. *Preprint.*

[7] P. Antsaklis, W. Kohn, A. Nerode, S. Sastry, (Eds.) 1995. *Hybrid Systems II, Lecture Notes in Computer Science, 999*, Springer Verlag.

[8] P. J. Antsaklis, J. A. Stiver, and M. Lemmon, 1993. Hybrid system modeling and autonomous control systems. *Hybrid Systems, Lecture Notes in Computer Science, 736*, Springer-Verlag, pp. 366-392.

[9] E. Azarin, O. Maler, and A. Pnueli, 1995. Symbolic controller synthesis for discrete and timed systems, *Hybrid Systems II, Lecture Notes in Computer Science, 999*, Springer Verlag, pp. 1-20.

[10] M. S. Branicky, 1995. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science, 138*, pp. 67-100.

[11] R. W. Brockett, 1993. Hybrid models for motion control systems. *Essays in Control: Perspectives in the theory and its applications*, pp. 29-53, Birkhauser, Boston.

[12] S. L. Chung, S. Lafortune and F. Lin, 1992. Limited lookahead policies in supervisory control of discrete event systems. *IEEE Transactions on Automatic Control, 37(12),* pp. 1921-1935.

[13] R. L. Grossman, A. Nerode, Rischel, Raven, (Eds.) 1993. *Hybrid Systems, Lecture Notes in Computer Science, 736*, Springer Verlag.

[14] T. Henzinger, P. Kopke, A. Puri and P. Varaiya, 1995. What's decidable about hybrid automata. *Proc. of the 27th Annual ACM Symposium on the Theory of Computing.*

[15] T. A. Henzinger and P. W. Kopke, 1997. Discrete time control for rectangular hybrid automata. *Proceedings , 24th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science*, Springer Verlag.

[16] M. Heymann 1990. Concurrency and discrete event control. *IEEE Control Systems Magazine, Vol. 10, No. 4,* pp 103-112.

[17] M. Heymann and F. Lin, 1994. On-line control of partially observed discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications, 4(3)*, pp. 221-236.

[18] M. Heymann and F. Lin, 1996. Discrete event control of nondeterministic systems. control of nondeterministic systems. *CIS Report 9601*, Technion, Israel.

[19] M. Heymann and F. Lin, 1996. Nonblocking supervisory control of nondeterministic systems. *CIS Report 9620*, Technion, Israel.

[20] M. Heymann, F. Lin and G. Meyer, 1997. Control synthesis for a class of hybrid systems subject to configuration based safety constraints. in O. Maler, Ed., *Hybrid and Real Time Systems, HART'97, Lecture Notes in Computer Science 1201*, pp. 376-390, Springer Verlag.

[21] M. Heymann, F. Lin and G. Meyer, 1997. Synthesis of minimally restrictive controllers for a class of hybrid systems. In P. Antsaklis, W. Kohn, A.Nerode and S. Sastry, Eds. *Hybrid Systems IV, Lecture Notes in Computer Science 1273*, pp. 134-159, Springer Verlag.

[22] M. Heymann, F. Lin and G. Meyer, 1997. Control synthesis for a class of hybrid systems subject to configuration based safety constraints. *NASA Technical Memorandum 112196.*

[23] D. Kapur and R. K. Shyamasundar, 1997. Synthesizing Controllers for Hybrid Systems. in O. Maler, Ed., *Hybrid and Real Time Systems, HART'97, Lecture Notes in Computer Science 1201*, pp. 361-375, Springer Verlag.

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

[25] F. Lin and W. M. Wonham, 1990. Decentralized control and coordination of discrete event systems with partial observation. *IEEE Transactions on Automatic Control, 35(12)*, pp. 1330-1337.

[26] F. Lin and W. M. Wonham, 1994. Supervisory control of timed discrete event systems under partial observation. *IEEE Transactions on Automatic Control, 40(3),* pp. 558-562.

[27] J. Lygeros, D.Godbole and S. Sastry, 1996. Multiagent Hybrid System Design using Game Theory and Optimal Control. *Proceedings 1996 Conference on Decision and Control*, pp. 1190-1195, Kobe, Japan, Dec. 11-13.

[28] O. Maler, Z. Manna and A. Pnueli, 1991. From timed to hybrid systems. In *Real Time: Theory in Practice, Lecture Notes in Computer Science 600,* pp. 447-484. Springer-Verlag.

[29] O. Maler, A. Pnueli and J. Sifakis, 1995. On the synthesis of discrete controllers for timed systems. *Lecture Notes in Computer Science 900,* pp. 229-242. Springer-Verlag.

[30] O. Maler (Ed.) 1997. *Hybrid and Real-Time Systems, HART'97, Lecture Notes in Computer Science, 1201*, Springer Verlag.

[31] Z. Manna and A. Pnueli, 1993. Verifying hybrid systems. *Hybrid Systems, Lecture Notes in Computer Science, 736*, Springer-Verlag, pp. 4-35.

[32] A. Nerode and W. Kohn, 1993. Models for hybrid systems: automata, topologies, controllability, observability. *Hybrid Systems, Lecture Notes in Computer Science, 736*, Springer-Verlag, pp. 317-356.

[33] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, 1993. An approach to the description and analysis of hybrid systems. *Hybrid Systems, Lecture Notes in Computer Science, 736*, Springer-Verlag, pp. 149-178.

[34] X. Nicollin, J. Sifakis, and S. Yovine, 1991. From ATP to timed graphs and hybrid systems. *Real Time: Theory in Practice, Lecture Notes in Computer Science 600*, Springer-Verlag, pp. 549-572.

[35] R. J. Ramadge and W. M. Wonham, 1987. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization, 25(1)*, pp. 206-230.

[36] P. J. Ramadge and W. M. Wonham, 1989. The control of discrete event systems. *Proceedings of IEEE, 77(1)*, pp. 81-98.

[37] H. Wong-Toi and G. Hoffmann, 1991. The Control of Dense Real-Time Systems. *Proceedings 1991 Conference on Decision and Control*, pp.1527-1528, Brighton, England.

[38] H. Wong-Toi, 1997. Synthesis of controllers for linear hybrid automata. *Proceedings 1997 Conference on Decision and Control*, San Diego, December 10-12.