Synthesis of Minimally Restrictive Legal Controllers for a Class of Hybrid Systems[1]

Michael Heymann[2], Feng Lin[3] and George Meyer[4]

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 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-restrictive (or minimally-interventive) one, which interferes in the system's operation only when constraint violation is otherwise inevitable. Thus, when composed to operate concurrently with another legal controller, our controller will never interfere with the operation of the other. Therefore, a minimally-restrictive 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. We describe in detail an algorithm for controller synthesis and examine through several examples questions associated with algorithm termination and controller existence.

1 Introduction

Various definitions 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] [7] [8] [19] [22]. Broadly speaking, they 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.

[1] 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.

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

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

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

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] [24]). This interest evolved progressively from logical systems, through "logically-timed" temporal systems [2] to real-time systems modeled as timed automata [20] and, most recently, to a restricted class of hybrid systems called *hybrid automata* [3] [19]. 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, where traditionally systems have been viewed as "purely" dynamic systems that are modeled by differential or difference equations [5] [7] [8]. More recently, control of purely discrete systems, modeled as discrete-event systems, also received attention in the literature [25] [26] [16]. 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 [21] [23], 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 [10].

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 restrictive ones.

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

We confine our attention to a special class of CHMs where system dynamics is *rate-limited* and legal guards are conjunctions or disjunctions of atomic formulas in the dynamic variables (of the type $S < C$, $S > C$, $S \leq C$, or $S \geq C$). We present an algorithm for synthesis of the minimally restrictive legal controller.

2 Design Philosophy

Intuitively, a controller for legal behavior of a hybrid system is minimally restrictive if it never takes action unless constraint violation becomes imminent. When the latter happens, the controller is expected do no more than 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 [25] [26]. Thus, a minimally restrictive 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 restrictive 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* [25]. 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 [12] [13]. 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 restrictive implementable sub-specification.

In the present paper we shall employ the same design philosophy for the synthesis of minimally restrictive 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 weakest preconditions, safe-exit conditions, and preemptive conditions that depend explicitly on continuous dynamics.

3 Comparison with Other Work

As state 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 [11] [12] [13] [14] [16] [17] [18] [9]). Needless to say, however, that there are significant differences between this work and that of 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 always well-defined (and every run is physically realizable). This prevents the possibility of ill-defined behaviors (that are possible in hybrid automata and are frequently referred to as the "prevention of time from progressing"). 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 to controller specification and design as proposed in this paper.

Finally, there are other recent works on control synthesis; in particular, the works reported in [6] [20] where attention is confined to timed automata, and where a similar fixed-point approach to control-synthesis is proposed. There are, however, significant differences between our work and the latter. First, we extend our attention to hybrid machines rather than confine it to timed automata. This allows, for example, dynamics of bounded-rate without resetting rather than constant rate with resetting. Our model also allows dynamic transitions in addition to event transitions. Secondly, contrary to that of [6] [20], our plant is autonomous in the sense that it can run by itself without the intervention of a controller. Because of this property, our control is "supervisory". It gives the plant freedom to do what it wants as long as there is no safety violation. Finally, and most importantly, we develop an explicit synthesis algorithm for design of a minimally restrictive controller, while in [6] [20] the fixed-point algorithm is only abstractly outlined (in the discrete-event control spirit) but no explicit algorithm is given.

Another noteworthy difference between the control problem for timed automata and hybrid automata is the decidability issue. While in the timed automata case the control synthesis problem is always decidable (a fact proved in [20]), this is not the case in hybrid automata (see [10]) and in fact our synthesis algorithm may not terminate as is demonstrated in a simple example in Section 6.

4 Hybrid Machines

We first introduce a modeling formalism for a class of hybrid systems which we call *hybrid machines* and which are a special case of *hierarchical hybrid machines* to be discussed elsewhere. We begin by an informal example.

4.1 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.

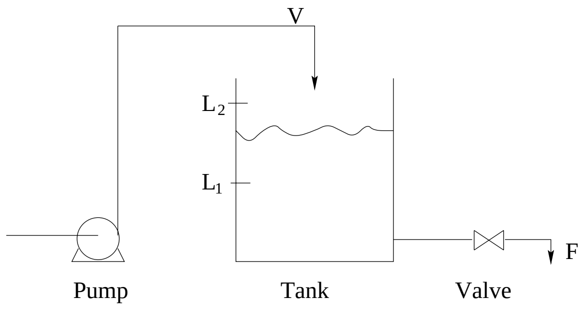

Fig. 1. Water Tank System

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:

$$PUMP||TANK||VALVE.$$

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 same at all three vertices. In general, however, they may be different. 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 $[L > L_2]$, $[L \leq L_2]$, $[L \geq L_1]$ and $[L < L_1]$ becoming true. The self-loop dynamic transition of the vertex <normal> labeled

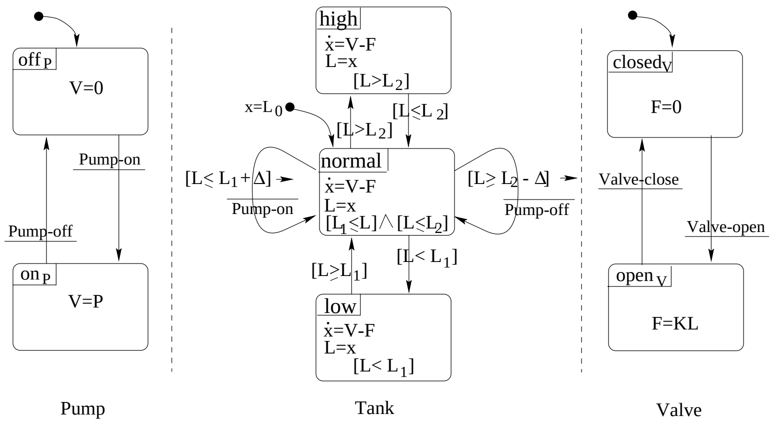

Fig. 2. Water Tank System CHM

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: $<$ off$_\text{P}$ $>$ and $<$ on$_\text{P}$ $>$. At the vertex $<$ off$_\text{P}$ $>$, the pump is off, reflected by the vertex output $V = 0$. Similarly, at the vertex $<$ 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 $<$ open$_\text{V}$ $>$ or $<$ 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.

4.2 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:

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

  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*.)
- $I = \{I_q : q \in Q\}$ is a set of invariants. I_q represents conditions under which the EHM is permitted to reside at q. A formal definition of I_q will be given in the next subsection.
- $E = \{(q, G \wedge \underline{\sigma} \rightarrow \overline{\sigma'}, q', x^0_{q'}) : q, q' \in Q\}$ is a set of edges (transition-paths), where q is the exiting vertex, q' the entering vertex, $\underline{\sigma}$ the input-event, $\overline{\sigma'}$ the output-event, G the guard to be formally defined in the next subsection, and $x^0_{q'}$ 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 with the assignment of the initial condition $x_{q'}(t_0) = x^0_{q'}$ (here t_0 denotes the time at which the vertex q' is entered). 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, then the initial condition is 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 become 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 absent or understood.) 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 sometimes abbreviated as $(q, \underline{\sigma}, q')$ when $\overline{\sigma'}$ and $x^0_{q'}$ are absent or understood.)
- (q_0, x_0) denote the initialization condition: q_0 is the initial vertex and $x_{q_0}(t_0) = x_0$.

For the EHM to be well-defined, we require that all vertices be *completely guarded*. That is, every invariant violation (possible under the dynamics) implies that some guard associated with a dynamic transition becomes true. (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.) Note that we do not require the converse to be true. That is, a transition can be triggered even if the invariant is not violated. We further require that, upon entry to a vertex q', the invariant $I_{q'}$ be true. It is however possible that, upon entry to q', one of the guards at q' is already true. In such a case, the EHM will immediately exit q' and go to a vertex specified by (one of) the true guards. Such a transition is considered instantaneous. Naturally, we only allow finite chains of such instantaneous transitions to be possible, otherwise we say that the EHM is *divergent*. That is, for the EHM to be *nondivergent*, the guards must be such that no sequence of instantaneous transitions can form a loop.

In this paper we will 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, $f_q(x_q, u_q) \in [k_q^L, k_q^U]$.

An execution of the EHM is a sequence

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

where e_i is the ith transition and t_i is the time when the ith transition takes place. For each execution, we define its trajectory, path and trace as follows.

– The trajectory of the execution is the sequence of the vector time functions of the state variables:

$$x_{q_0}, x_{q_1}, x_{q_2}, \dots$$

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

Remark. 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.

4.3 Composite hybrid machine

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. If the EHM of the output variable does not update the variable, its value will remain unchanged. The set of shared variables defines a signal space $S = [S_1, S_2, ..., S_m]$.

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 a true guard) will execute $\underline{\sigma}$ (and its associated transition) concurrently with the occurrence of $\overline{\sigma}$. An output-event can be generated by at most one EHM. Notice that input-events do not synchronize among themselves. Notice further that this formalism is a special case of the prioritized synchronous composition formalism [11], where each event is in the priority set of at most one parallel component.

By introducing the shared variables S, we can now define invariants and guards formally as boolean combinations of inequalities of the form (called *atomic formulas*)

$$S_i \geq C_i \quad \text{or} \quad S_i \leq C_i,$$

where S_i is a shared variable and C_i is a real constant.

Remark. For consistency of the computations, we should mainly deal with closed invariants as well as closed guards (that is, the sets in which the invariants or guards are true are closed). Since the complement of a closed set is not closed, we should distinguish, for $S_i \geq C_i$, its strict negation $S_i < C_i$ and its negation $S_i \leq C_i$. To ensure the closedness, we will maily consider negation (unless otherwise stated) and, with some abuse of notation, write $\neg(S_i \geq C_i) = (S_i \leq C_i)$. Thus, it is possible that a boolean expression and its negation are both true at a point or on a hyperplane. If this matters, as in the case that several guards become true simultaneously, we will introduce suitable prioritization, as will be discussed below.

To describe the behavior of

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

we define a *configuration* of the CHM to be

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

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, $< \ , q^2_{i_2}, ..., q^n_{i_n} >$ is interpreted as the set

$$<, q^2_{i_2}, ..., q^n_{i_n} > = \{< q^1_{i_1}, q^2_{i_2}, ..., q^n_{i_n} >: q^1_{i_1} \in Q^1\}$$

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

A transition

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

of a CHM is a triple, where $q = < q^1_{i_1}, q^2_{i_2}, ..., q^n_{i_n} >$ is the source configuration, $q' = < q^1_{i'_1}, q^2_{i'_2}, ..., q^n_{i'_n} >$ the target configuration, and l the label that triggers the transition. l can be either an event or a guard (becoming true). 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 in E^j. On the other hand, if $l = G$ is a guard, then there must exists 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 defined 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. 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

$Q = Q^1 \times Q^2 \times ... \times Q^n$,
$\Sigma = \Sigma^1 \cup \Sigma^2 \cup ... \cup \Sigma^n$,
$D = \{(x_q, y_q, u_q, f_q, h_q) : q = < q^1_{i_1}, q^2_{i_2}, ..., q^n_{i_n} > \in Q^1 \times Q^2 \times ... \times Q^n\}$
combines all the dynamics of $q^j_{i_j}, j = 1, 2, ..., n$,

$$\begin{aligned} I &= \{I_{q^1_{i_1}} \wedge I_{q^2_{i_2}} \wedge ... \wedge I_{q^n_{i_n}} :< q^1_{i_1}, q^2_{i_2}, ..., q^n_{i_n} > \in Q^1 \times Q^2 \times ... \times Q^n\}, \\ E & \quad \text{is defined as above, } and \\ (q_0, x_0) &= (< q^1_0, q^2_0, ..., q^n_0 >, (x^1_0, x^2_0, ..., x^n_0)). \end{aligned}$$

Therefore, we can define an execution of a CHM in the same way as that of an EHM.

Recall that our model also 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

$$\begin{tikzcd} q_1 \arrow[r, shift left, "G"] & q_2 \arrow[l, shift left, "\neg G"] \arrow[r, "\sigma"] & q', \end{tikzcd}$$

where q has been partitioned into q_1 and q_2, with $I_{q_1} = I_q \wedge \neg G$ and $I_{q_2} = I_q \wedge G$. It follows that a guarded event transition can be treated as a combination of a dynamic and an event transition.

Thus, transitions in CHMs can be classified into two types: (1) dynamic transitions, that are labeled by guards only, and (2) event transitions, that are labeled by events.

5 Control

5.1 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 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.

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, the controller is assumed not to 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 [25] [26] to deal with uncontrollable event transitions. We shall elaborate on this issue elsewhere.

A legal controller C is said to be *less restrictive* than another legal controller C' if every execution permitted by C' is also permitted by C (a formal definition will be given in the next subsection). A legal controller is said to be *minimally restrictive* if it is less restrictive than any legal controller.

With a slight modification of the formalism that we shall present in the next subsection, two or more controllers can be combined by parallel composition to form a composite controller. An important characteristic of a minimally restrictive controller is the fact that when it is combined with any other controller (legal or not), that is possibly designed for satisfying some other specifications, such as liveness 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 restrictive 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.

5.2 Control synthesis

As stated, our control objective is to ensure that the system CHM never enter the set of illegal configurations Q_b. 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 restrictive 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.

In order to do this, we will need to consider first the time at which a predicate will become true. We begin by considering an atomic formula

$$P{=}(S_i \geq C_i).$$

Suppose that at a given instant t at which $S_i(t) = S_i$, P is false; that is, $S_i{\leq}C_i$ (or actually $S_i < C_i$). Then the interval of time that will elapse before P can become true is bounded by the minimum value

$$T_{min}(true(P)) = \begin{cases} (C_i - S_i)/{r_i}^U & \text{if } {r_i}^U > 0 \\ \infty & \text{otherwise,} \end{cases}$$

and the maximum value

$$T_{max}(true(P)) = \begin{cases} (C_i - S_i)/{r_i}^L & \text{if } {r_i}^L > 0 \\ \infty & \text{otherwise,} \end{cases}$$

where, ${r_i}^L$ and ${r_i}^U$ are the lower and upper bounds of $\dot{S}$, respectively (recall that, by our assumption, the shared variables S_i are rate-bounded; that is, $\dot{S}_i{\in}[{r_i}^L, {r_i}^U]$).

If, at the instant t, P is true, then clearly $T_{min}(true(P)) = T_{max}(true(P)) = 0$. Similarly, if P is given by

$$P{=}(S_i \leq C_i),$$

then if, at the instant t, P is true, $T_{min}(true(P)) = T_{max}(true(P)) = 0$, and otherwise, the minimum interval is

$$T_{min}(true(P)) = \begin{cases} (C_i - S_i)/{r_i}^L & \text{if } {r_i}^L < 0 \\ \infty & \text{otherwise,} \end{cases}$$

and the maximum interval is

$$T_{max}(true(P)) = \begin{cases} (C_i - S_i)/{r_i}^U & \text{if } {r_i}^U < 0 \\ \infty & \text{otherwise.} \end{cases}$$

For conjunction of two predicates, $P = P_1{\wedge}P_2$, it is clear that

$$T_{min}(true(P)) = max\{T_{min}(true(P_1)), T_{min}(true(P_2))\}$$

$$T_{max}(true(P)) = max\{T_{max}(true(P_1)), T_{max}(true(P_2))\},$$

and for disjunction of two predicates, $P = P_1 \vee P_2$

$$T_{min}(true(P)) = min\{T_{min}(true(P_1)), T_{min}(true(P_2))\}$$

$$T_{max}(true(P)) = min\{T_{max}(true(P_1)), T_{max}(true(P_2))\}.$$

Also, if a predicate is always false: $P = false$, then

$$T_{min}(true(P)) = T_{max}(true(P)) = \infty.$$

To streamline the ensuing analysis, we shall assume that the invariants of all legal configurations are expressed in conjunctive normal form

$$I = (I_{11} \vee ... \vee I_{1l_1}) \wedge ... \wedge (I_{m1} \vee ... \vee I_{ml_m}),$$

where I_{ij}=$(S_{ij} \geq C_{ij})$, I_{ij}=$(S_{ij} \leq C_{ij})$. Similarly, all the guards are in conjunctive normal form

$$G = (G_{11} \vee ... \vee G_{1l_1}) \wedge ... \wedge (G_{m1} \vee ... \vee G_{ml_m}).$$

When competing guards become true simultaneously, we shall give priority to a legal guard (i.e., one that leads to a legal configuration) over an illegal one, and we shall resolve nondeterministically between competing legal guards.

Without loss of generality, we shall assume that the invariant is violated if and only if one or more of the guards is true - recall the difference between negation and strict negation as discussed in the previous remark. (Otherwise, we can conjoin with the invariant the negation of the guards.)

The role of the least restrictive controller is to force event transitions (to other legal configurations) at "the boundary of the legal region". To specify the forcing condition formally, we need to introduce, for a predicate P, *critical*(P) that captures the fact that P is about to be violated. Thus, for $P = (S_i \leq C_i)$, we define

$$critical(P) = \begin{cases} (S_i \geq C_i) & \text{if } {r_i}^U > 0 \\ \text{false} & \text{otherwise,} \end{cases}$$

Similarly, for $P = (S_i \geq C_i)$,

$$critical(P) = \begin{cases} (S_i \leq C_i) & \text{if } {r_i}^L < 0 \\ \text{false} & \text{otherwise.} \end{cases}$$

For conjunction of two predicates $P = P_1 \wedge P_2$,

$$critical(P) = critical(P_1) \vee critical(P_2).$$

and for disjunction of two predicates $P = P_1 \vee P_2$,

$$critical(P) = critical(P_1) \wedge critical(P_2).$$

For the CHM to move from one configuration q to another configuration q', the invariant $I_{q'}$ must be satisfied upon entry to q'. (Notice that if q' is the legal subconfiguration of a configuration whose invariant has been split to a legal part and an illegal part, satisfaction of the invariant $I_{q'}$ is not automatically satisfied.) 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 $I_{q'}$ upon entry to q'. Since some of the shared variables that appear in $I_{q'}$ are possibly (re-)initialized upon entering q', the condition $wp(q, \underline{\sigma}, q')$ can be computed from $I_{q'}$ by substituting into $I_{q'}$ 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

$$S_i = h_j(x^0_{q'}, u_{q'}).$$

With these preliminaries, we can now discuss our synthesis algorithm. Let us consider a legal configuration q. As discussed earlier, we assume that transitions leaving q are either dynamic transitions or event transitions, and can lead to either legal or illegal configurations. Therefore, we classify the transitions 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:

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

Since transitions in $ET_b(q, Q_b)$ can be prevented by simply not being triggered, we need not discuss them further. If $DT_b(q, Q_b) = \emptyset$, then no dynamic transition from q leads to an illegal configuration and hence there is no need to split q. Otherwise, if $DT_b(q, Q_b) \neq \emptyset$, we may need to split q as discussed below. Let us consider the different cases.

Case 1. $DT_g(q, Q_b) = \emptyset$

Since $DT_g(q, Q_b) = \emptyset$, the only way to prevent transitions in $DT_b(q, Q_b)$ from taking place, is for the controller to trigger an event transition $(q, \underline{\sigma}, q') \in ET_g(q, Q_b)$, provided this set is nonempty, thereby forcing the CHM from q to q'.

To find under what condition we can count on such a $(q, \underline{\sigma}, q')$ to take the CHM to another legal state q', we define the following *safe-exit condition*

$$sc(q, \underline{\sigma}, q') = (T_{max}(true(wp(q, \underline{\sigma}, q'))) \leq T_{min}(false(I_q))),$$

where

$T_{max}(true(wp(q,\underline{\sigma},q')))$ is the latest time $wp(q,\underline{\sigma},q')$ will be true, and

$T_{min}(false(I_q))$ is the earliest time I_q will become false.

Therefore, $sc(q,\underline{\sigma},q')$ is true if $wp(q,\underline{\sigma},q')$ is guaranteed to be satisfied before I_q is violated. Under this condition, the CHM can always be forced to safely exit q. Notice that $wp(q,\underline{\sigma},q') \Rightarrow sc(q,\underline{\sigma},q')$, that is, we can always safely exit to q' when $wp(q,\underline{\sigma},q')$ is satisfied.

If $I_q \not\Rightarrow sc(q,\underline{\sigma},q')$, then we will split the configuration q into two sub-configurations q_1 and q_2 by partitioning the invariant I_q (and associating with each of the sub-configurations the corresponding invariant) as

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

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 $(q_2,\underline{\sigma},q')$ is not permitted or is impossible (because of the invariant violation). Also the transition from q_1 to q_2 is dynamic with the guard $\neg sc(q,\underline{\sigma},q')$ (strict negation), and from q_2 to q_1 with guard $sc(q,\underline{\sigma},q')$.

Clearly, q_1 is legal in the sense that from it the transition to the legal configuration q' can be forced, while q_2 is not legal. From q_1, the dynamic transitions in $DT_b(q_1,Q_b)$ and the dynamic transition $(q_1,\neg sc(q,\underline{\sigma},q'),q_2)$ are illegal and must not be permitted. To prevent these transitions from taking place in a minimally restrictive manner, $\underline{\sigma}$ must be forced just before any one of them can actually take place. In other words, $\underline{\sigma}$ must be forced just before I_{q_1} becomes false.

The condition under which the transition $(q,\underline{\sigma},q')$ will be forced is then

$$critical(I_{q_1}) = critical(I_q \wedge sc(q,\underline{\sigma},q')).$$

If there are more than one legal event transition in $ET_g(q,Q_b)$, then we will split q into q_1 and q_2 as follows.

$$\begin{aligned} I_{q_1} &= I_q \wedge (\vee_{(q,\underline{\sigma},q')\in ET_g(q,Q_b)} sc(q,\underline{\sigma},q')) \\ I_{q_2} &= I_q \wedge \neg(\vee_{(q,\underline{\sigma},q')\in ET_g(q,Q_b)} sc(q,\underline{\sigma},q')). \end{aligned}$$

The condition under which a legal event transition $(q,\underline{\sigma},q')$ needs to be forced is given by

$$critical(I_{q_1}) \wedge wp(q,\underline{\sigma},q').$$

Case 2. $ET_g(q,Q_b) = \emptyset$

Since $ET_g(q,Q_b) = \emptyset$, 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.

Note that because of configuration splitting, the target configuration of a dynamic transition guarded by a guard G, may depend on the dynamic condition at the source configuration at the instant when G becomes true. Thus, if the configuration q' is split into q'_1 and q'_2, then we may have either $(q,G,q'_1) \in$

$DT_g(q, Q_b)$ or $(q, G, q_2') \in DT_b(q, Q_b)$ depending on the dynamic conditions. To deal with such cases effectively, it will be convenient to modify (q, G, q') by the following equivalent dynamic transition

$$(q, G \wedge wp(q, G, q'), q').$$

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 $G \wedge wp(q, G, q')$ becomes true. The earliest time $G \wedge wp(q, G, q')$ will become true is $T_{min}(G \wedge wp(q, G, q'))$ 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*

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

be satisfied[5]. Therefore, we will split the configuration q into two sub-configurations q_1 and q_2, by partitioning the invariant I_q as

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

As in Case 1, 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.

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

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

General case.

That is, we require neither $ET_g(q, Q_b) = \emptyset$ nor $DT_g(q, Q_b) = \emptyset$. In this general case, we can either rely on legal dynamic transitions to preempt the illegal dynamic transitions, or if this does not happen, force some legal event transitions. Therefore, we shall split q into q_1 and q_2 as follows.[6]

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

[5] We take the convention that if $T_{min}(true(G \wedge wp(q, G, q'))) = \infty$, then $pc(q, G, q') = true$ even if $T_{max}(false(I_q)) = \infty$.

[6] If $(q, G, q') \in DT_b(q, Q_b)$ cannot be prevented from occurring, then we must consider q as illegal. In that case $I_{q_1} = false$ and $I_{q_2} = I_q$.

The condition under which a legal event transition $(q, \underline{\sigma}, q')$ needs to be forced is now given by[7]

$$critical(I_{q_1}) \wedge wp(q, \underline{\sigma}, q') \wedge (\neg(\wedge_{(q,G,q') \in DT_b(q,Q_b)} pc(q, G, q'))).$$

Notice that if we adopt the convention that

$$\begin{aligned} \wedge_{(q,G,q') \in DT_b(q,Q_b)} pc(q, G, q') &= true \ \ if \ DT_b(q, Q_b) = \emptyset \\ \vee_{(q,\underline{\sigma},q') \in ET_g(q,Q_b)} sc(q, \underline{\sigma}, q') &= false \ \ if \ ET_g(q, Q_b) = \emptyset, \end{aligned}$$

then this general case covers both Case 1 and Case 2.

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

$$BC := Q_b;$$

2. Set of pending configurations

$$PC := Q - Q_b;$$

3. New set of pending configurations

$$NPC := \emptyset;$$

4. For each $q \in PC$ set its *configuration origin* as

$$CO(q) = q;$$

Iteration

[7] There is a possible complication if the newly defined guards form an instantaneous configuration-cluster (formed by simultaneously true guards) that may force an unbounded instantaneous sequence of consecutive transitions. If this occurs, further analysis will be required.

5. For all $q \in PC$ do

$$\begin{aligned} I_{q_1} &:= I_q \wedge ((\wedge_{(q,G,q')\in DT_b(q,BC)} pc(q,G,q')) \\ &\quad \vee(\vee_{(q,\underline{\sigma},q')\in ET_g(q,BC)} sc(q,\underline{\sigma},q'))); \\ I_{q_2} &:= I_q \wedge (\neg(\wedge_{(q,G,q')\in DT_b(q,BC)} pc(q,G,q')) \\ &\quad \wedge\neg(\vee_{(q,\underline{\sigma},q')\in ET_g(q,BC)} sc(q,\underline{\sigma},q')));. \end{aligned}$$

If $I_{q_1} \neq false$, then

$$\begin{aligned} NPC &:= NPC \cup \{q_1\}; \\ CO(q_1) &:= CO(q); \end{aligned}$$

If $I_{q_2} \neq false$, then

$$BC := BC \cup \{q_2\};$$

6. If $PC = NPC$, go to 8.
7. Set

$$\begin{aligned} PC &:= NPC; \\ NPC &:= \emptyset; \end{aligned}$$

Go to 5;

Construction of C

8. Define vertices, events and dynamics:

$$\begin{aligned} Q^c &:= PC; \\ \Sigma^c &:= \Sigma \cup \{\underline{\tilde{\sigma}} : \sigma \in \Sigma\}; \\ D^c &:= \emptyset; \end{aligned}$$

9. Define transitions:

$$\begin{aligned} E^c &:= \{(q, critical(I_q) \wedge wp(q,\underline{\sigma},q') \\ &\quad \wedge(\neg(\wedge_{(q,G,q'')\in DT_b(q,BC)} pc(q,G,q''))) \to \overline{\sigma}, q') : \\ &\quad q,q' \in Q^c \wedge (CO(q),\underline{\sigma},CO(q')) \in E\}; \\ E^c &:= E^c \cup \{(q, wp(q,\underline{\sigma},q') \wedge \underline{\tilde{\sigma}} \to \overline{\sigma}, q') : \\ &\quad q,q' \in Q^c \wedge (CO(q),\underline{\sigma},CO(q')) \in E\}; \end{aligned}$$

10. End.

Therefore, the controller C has no dynamics. Its vertices are copies of the legal configurations of CHM that survive after the partition. Its events include the output-events $\overline{\sigma}$ and the input-events $\underline{\tilde{\sigma}}$ from the environment or other controllers. Its transitions are of two types: (1) dynamic transitions that are triggered when the CHM is about to become potentially illegal; and (2) guarded event transitions that are triggered by input-events.

Another controller D can be embedded into C as follows. First, all the output-events $\overline{\sigma}$ in D are replaced by $\overline{\tilde{\sigma}}$ to obtain $\tilde{D}$. Then the embedded control system is given by

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

We can now prove the following

Theorem 1. If Algorithm 1 terminates in a finite number of steps and if there is no instantaneous configuration-cluster (that may force an unbounded instantaneous sequence of consecutive transitions), then the controller synthesized is a minimally restrictive legal controller in the following sense.

1. For any controller D, an execution in $CHM||C||\tilde{D}$ will never visit illegal configurations Q_b.
2. For any legal controller D, an execution is possible in $CHM||D$ if and only if its corresponding execution is possible in $CHM||C||\tilde{D}$.

Proof

Since Algorithm 1 terminates in a finite number of steps and no sequence of instantaneous transitions form a loop, the controller is well defined. In particular, time progresses as execution continues and during any finite interval of time only a finite number of transitions take place.

To prove part 1, it is sufficient to show that an execution 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 an execution

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

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, let us assume that

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

is a possible execution of $CHM||D$ but the last transition from q_{n-1} to q_n is impossible in $CHM||C||\tilde{D}$, that is, $q_n \notin Q^c$. Then by our construction of q_n, there exists a continuation of the execution in $CHM||D$

$$q_n \xrightarrow{e_{n+1},t_{n+1}} q_{n+1} \longrightarrow \ldots \xrightarrow{e_{n+m},t_{n+m}} q_{n+m}$$

that will lead to an illegal configuration $q_{n+m} \in Q_b$. This execution cannot be prevented by D, a contradiction to the hypothesis that D is legal.

On the other hand, if

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

is a possible execution of $CHM||C||\tilde{D}$ but the last transition from q_{n-1} to q_n is impossible in $CHM||D$, then this last transition must be triggered by a dynamic transition in C when the following guard becomes true:

$$\begin{aligned} G_c = {} & critical(I_{q_{n-1}}) \wedge wp(q_{n-1}, \underline{\sigma}, q_n) \\ & \wedge(\neg(\wedge_{(q_{n-1},G,q') \in DT_b(q_{n-1},BC)} pc(q_{n-1}, G, q'))). \end{aligned}$$

Since the transition (q_{n-1}, G_c, q_n) does not take place in $CHM||D$, by our construction of G_c, the next transition

$$q_{n-1} \xrightarrow{e'_n,t'_n} q'_n$$

could lead to $q'_n \notin Q^c$. By the same argument as above, we conclude that D is illegal, a contradiction.

■

Examples will be given in the next section to illustrate the algorithm.

6 Discussion and Examples

Our algorithm works for most examples we encounted in practice. An example to show how our algorithm solves a steam boiler control problem [1] has been given in [15].

There are, however, situations in which our algorithm does not resolve the controller design problem. In the first type of situations, the algorithm terminates finitely, but the closed loop system includes instantaneous configuration-cluster and possibly inescapable instantaneous unbounded sequences of transitions. In this case, a minimally restrictive legal controller may or may not exist as is shown in Examples 1 and 2 below, and further analysis beyond the algorithm is necessary. In the second type of situations, the algorithm does not terminate as shown in Example 3.

Example 1. In this example, we will see that although the algorithm terminates, it does so with instantaneous loops (a special case of instantaneous configuration-cluster), and the controller obtained is not legal (since a legal controller does not exist).

Consider the hybrid system shown in Figure 3. It models a two-tank system, where both tanks are leaking with rate 2. A pump with rate 3 can be switched between the two tanks (event $\underline{\sigma_1}$ and $\underline{\sigma_2}$). The system starts with both tanks non-empty $(x_1(0) > 0, x_2(0) > 0)$. The system becomes illegal when one of the tanks becomes empty, which is represented by illegal configuration c that has no dynamics and true invariant. Since $2 + 2 > 3$, no legal controller exists that can prevent the system from becoming illegal eventually. However, as we will show below, the algorithm terminates.

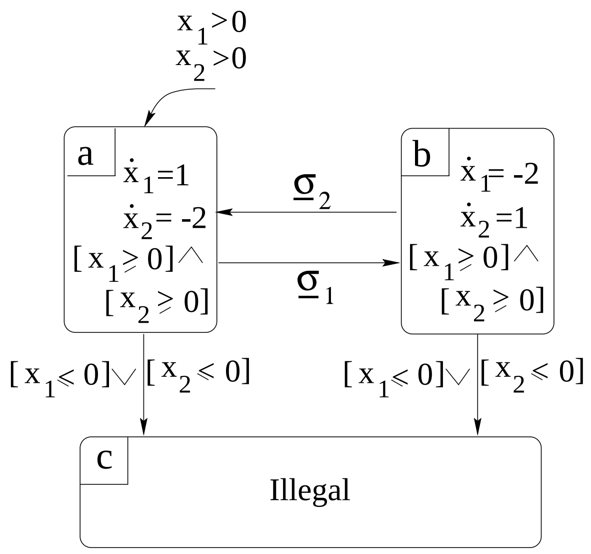

Fig. 3. CHM of Example 1

Initialization

$$\begin{aligned} BC &= \{c\}, \\ PC &= \{a, b\}, \\ I_a &= I_b = [x_2 \geq 0] \wedge [x_1 \geq 0]. \end{aligned}$$

1st Iteration

$$\begin{aligned} &pc(a, [x_1 \leq 0] \vee [x_2 \leq 0], c) = false, \\ &sc(a, \underline{\sigma_1}, b) = I_a. \end{aligned}$$

Therefore,

$$I_{a_1} = I_1 \wedge (pc(a, [x_1 \leq 0] \vee [x_2 \leq 0], c) \vee sc(a, \underline{\sigma_1}, b)) = I_a.$$

Similarly,

$$I_{b_1} = I_b.$$

Since there is no change to I_a and I_b, the algorithm terminates after just one step.

The controller C generated by the algorithm is shown in Figure 4. The guard G_1 trigging the event transition $\overline{\sigma_1}$ is calculated as follows.

$$critical(I_a) = [x_1 \leq 0] \vee [x_2 \leq 0],$$
$$wp(a, \underline{\sigma_1}, b) = [x_1 \geq 0] \wedge [x_2 \geq 0],$$
$$G_1 = critical(I_a) \wedge wp(a, \underline{\sigma_1}, b).$$

Similarly, G_2 can be calculated, which is the same as G_1. Clearly, the controller in Figure 4 is not a legal controller and, in fact, a minimally restrictive legal controller does not exist. This however does not contradict Theorem 1 because there exists an instantaneous loop in C that occurs when $x_1 = x_2 = 0$.

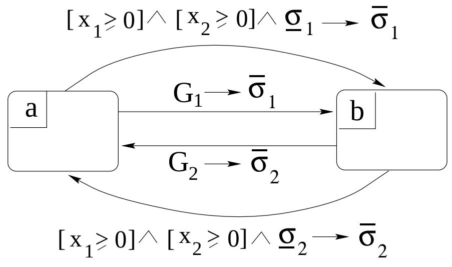
$[x_1 \geq 0] \wedge [x_2 \geq 0] \wedge \underline{\sigma}_1 \rightarrow \overline{\sigma}_1$
a
$G_1 \rightarrow \overline{\sigma}_1$
b
$G_2 \rightarrow \overline{\sigma}_2$
$[x_1 \geq 0] \wedge [x_2 \geq 0] \wedge \underline{\sigma}_2 \rightarrow \overline{\sigma}_2$

Fig. 4. Controller of Example 1

Example 2. Although the controller designed by the algorithm is not guaranteed to be legal when there exist instantaneous loops, such instantaneous loops do not necessarily invalidate the resulting controller. This can be shown by changing the pumping rate from 3 to 5. This change will not affect the synthesis procedure and the resulting controller is the same. Under this rate, however, the controller is legal and minimally restrictive.

Example 3. This example shows that our algorithm may not terminate. In this example there exists no minimally restrictive legal controller.

Let us modify Example 1 by assuming that there is a one second delay in switching the pump. That is, it takes one second for the switching command to be actually executed. The modified two-tank system is shown in Figure 5. As in Example 1, no legal controller exists because $2 + 2 > 3$.

Table 1 illustrates the computation of the algorithm and shows that the algorithm does not terminate.

Example 4. Consider the same system as in Example 3, but change the pumping rate from 3 to 5. Under this rate, the algorithm terminates and generates the following legal and minimally restrictive controller: Switching the pump to tank i when $x_i = 2, i = 1, 2$.

Examples 3 and 4 show that whether the algorithm terminates may depend on the (continuous) dynamics of the system.

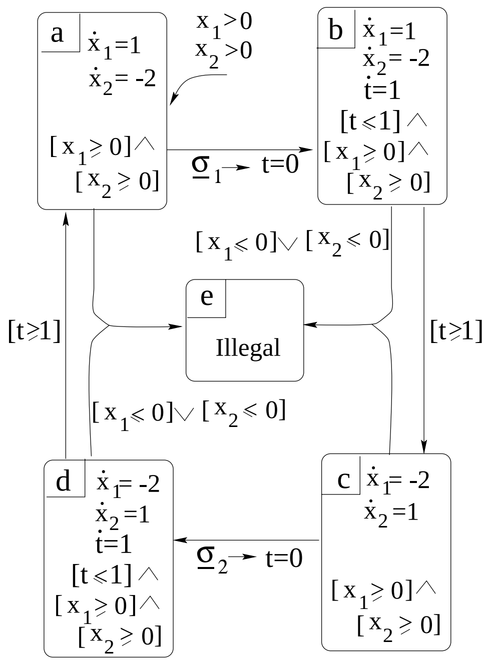

Fig. 5. CHM of Example 3

	a	b	c	d
initial	$[x_1\geq 0]\wedge[x_2\geq 0]$	$[x_1\geq 0]\wedge[x_2\geq 0]\wedge[t<1]$	$[x_1\geq 0]\wedge[x_2\geq 0]$	$[x_1\geq 0]\wedge[x_2\geq 0]\wedge[t<1]$
1st	$[x_1\geq 0]\wedge[x_2\geq 0]$	$[x_1\geq 0]\wedge[x_2>2-2t]\wedge[t<1]$	$[x_1\geq 0]\wedge[x_2\geq 0]$	$[x_1>2-2t]\wedge[x_2\geq 0]\wedge[t<1]$
2nd	$[x_1\geq 0]\wedge[x_2>2]$	$[x_1\geq 0]\wedge[x_2>2-2t]\wedge[t<1]$	$[x_1>2]\wedge[x_2\geq 0]$	$[x_1>2-2t]\wedge[x_2\geq 0]\wedge[t<1]$
3rd	$[x_1\geq 0]\wedge[x_2>2]$	$[x_1>t+1]\wedge[x_2>2-2t]\wedge[t<1]$	$[x_1>2]\wedge[x_2\geq 0]$	$[x_1>2-2t]\wedge[x_2>t+1]\wedge[t<1]$
4th	$[x_1>1]\wedge[x_2>2]$	$[x_1>t+1]\wedge[x_2>2-2t]\wedge[t<1]$	$[x_1>2]\wedge[x_2>1]$	$[x_1>2-2t]\wedge[x_2>t+1]\wedge[t<1]$
...	...	...	...	...

Table 1. Controller synthesis of Example 3

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. 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.
6. 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.
7. M. S. Branicky, 1995. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science, 138*, pp. 67-100.
8. R. W. Brockett, 1993. Hybrid models for motion control systems. In H.L. Trentelman and J.C.Willems (Eds.), *Essays in Control: Perspectives in the theory and its applications*, pp. 29-53, Birkhauser, Boston.
9. 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.
10. 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.*
11. M. Heymann 1990. Concurrency and discrete event control, *IEEE Control Systems Magazine, Vol. 10, No.4,* pp 103-112.
12. 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.
13. M. Heymann and F. Lin, 1996. Discrete event control of nondeterministic systems. control of nondeterministic systems, *CIS Report 9601*, Technion, Israel.
14. M. Heymann and F. Lin, 1996. Nonblocking supervisory control of nondeterministic systems, *CIS Report 9620*, Technion, Israel.
15. 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", Proceedings of HART97, *Lecture Notes in Computer Science 1201* pp. 376-390, Springer Verlag.
16. F. Lin and W. M. Wonham, 1988. On observability of discrete event systems. *Information Sciences, 44(3)*, pp. 173-198.
17. 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.

18. 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.
19. 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.
20. 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.
21. Z. Manna and A. Pnueli, 1993. Verifying hybrid systems. *Hybrid Systems, Lecture Notes in Computer Science, 736*, Springer-Verlag, pp. 4-35.
22. 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.
23. X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, 1993. Am approach to the description and analysis of hybrid systems. *Hybrid Systems, Lecture Notes in Computer Science, 736*, Springer-Verlag, pp. 149-178.
24. X. Nicollin, J. Sifakis, and S. Yovine, 1991. From ATP to timed graphs and hybrid systems. In *Real Time: Theory in Practice*, Lecture Notes in Computer Science 600, Springer-Verlag, pp. 549-572.
25. 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.
26. P. J. Ramadge and W. M. Wonham, 1989. The control of discrete event systems. *Proceedings of IEEE, 77(1)*, pp. 81-98.

This article was processed using the LaTeX macro package with LLNCS style