/N-31

NASA Technical Memorandum 112196

Control Synthesis for a Class of Hybrid Systems Subject to Configuration-Based Safety Constraints

Michael Heymann, Feng Lin, and George Meyer

June 1997

National Aeronautics and
Space Administration

NASA Technical Memorandum 112196

Control Synthesis for a Class of Hybrid Systems Subject to Configuration-Based Safety Constraints

Michael Heymann, Technion, Israel Institute of Technology, Haifa, Israel
Feng Lin, Wayne State University, Detroit, Michigan
George Meyer, Ames Research Center, Moffett Field, California

June 1997

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

CONTROL SYNTHESIS FOR A CLASS OF HYBRID SYSTEMS SUBJECT TO CONFIGURATION-BASED SAFETY CONSTRAINTS[*]

Michael Heymann,[†] Feng Lin,[‡] and George Meyer

Ames Research Center

SUMMARY

We examine a class of hybrid systems which we call *Composite Hybrid Machines* (CHMs) that consists of the concurrent (and partially synchronized) operation of *Elementary Hybrid Machines* (EHMs).

Legal behavior, specified by a set of *illegal* configurations that the CHM may not enter, is to be achieved by the concurrent operation of the CHM with a suitably designed *legal* controller. In the present paper we focus on the problem of synthesizing a legal controller, whenever such a controller exists. More specifically, we address the problem of synthesizing the *minimally restrictive* legal controller.

A controller is minimally restrictive if, when composed to operate concurrently with another legal controller, it will never interfere with the operation of the other controller and, therefore, can be composed to operate concurrently with any other controller that may be designed to achieve liveness specifications or optimality requirements without the need to reinvestigate or reverify 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 $x \leq x_0$ or $x \geq x_0$). We present an algorithm for synthesis of the minimally restrictive legal controller.

We demonstrate our approach by synthesizing a minimally restrictive controller for a steam boiler (the verification of which recently received a great deal of attention).

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 (refs. 1–6). Broadly speaking, they are systems in which change occurs in response to events that take place discretely, asynchronously, and sometimes nondeterministically and also 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.

[*]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.

[†]Technion, Israel Institute of Technology, Haifa 32000, Israel.

[‡]Wayne State University, Detroit, Michigan.

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., references 2 and 7). This interest evolved progressively from logical systems, through "logically timed" temporal systems (refs. 8 and 9) to real-time systems modeled as timed-automata and, most recently, to a restricted class of hybrid systems called *hybrid automata* (ref. 1). 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 (refs. 3, 4, and 10). More recently, control of purely discrete systems, modeled as discrete-event systems, also received attention in the literature (refs. 11 and 12). 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 (refs. 13–15), 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 restricted class of hybrid systems that we call *composite hybrid machines* (CHMs). We confine our attention to bounded rate CHMs, in which the dynamic rates are bounded by lower and upper constant bounds. Control is confined to event synchronization; that is, the controller can affect the system's behavior only by discrete commands. These hybrid systems are a generalization of timed automata, which in turn generalize discrete event systems by introducing real-time constraints. For such systems it is natural to specify the control objective in terms of safety constraints and liveness constraints, much in the spirit of the control of discrete-event systems. Indeed, this generalization is on one hand simple enough to be computationally tractable, and on the other hand complex enough to provide some substantial new insight and a sense of new research direction.

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 this happens, the controller is expected to do no more than prevent the system from becoming "illegal." This is a familiar setting in the discrete-event control literature, where 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 (refs. 11 and 12). 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* (ref. 11). 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 (refs. 16 and 17). The *subspecification* 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 subspecification.

In the present paper we shall employ the same design philosophy for the synthesis of minimally restrictive controllers of hybrid systems. While the approach is, in principle, very general and can be employed for a wide range of specifications, we confine our attention in the present paper to a restricted class of *safety* specifications. In particular, we shall consider only the problem where the controller is required to prevent the system from entering a specified set of *illegal* configurations. Although we shall not show this explicitly in this paper, a wide class of specifications can be transformed into the setting considered here.

We shall restrict our attention further to *bounded-rate* hybrid systems. That is, we consider systems in which the rates of the dynamic variables are bounded by finite constants. It is not difficult to show that, even in this simple case, the question of existence of a controller may be computationally rather tricky.

3 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 (Heymann and Lin, Hierarchical Hybrid Machines, in preparation). Hybrid machines are similar in spirit to hybrid automata as introduced in reference 1. We begin by an informal example.

3.1 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.

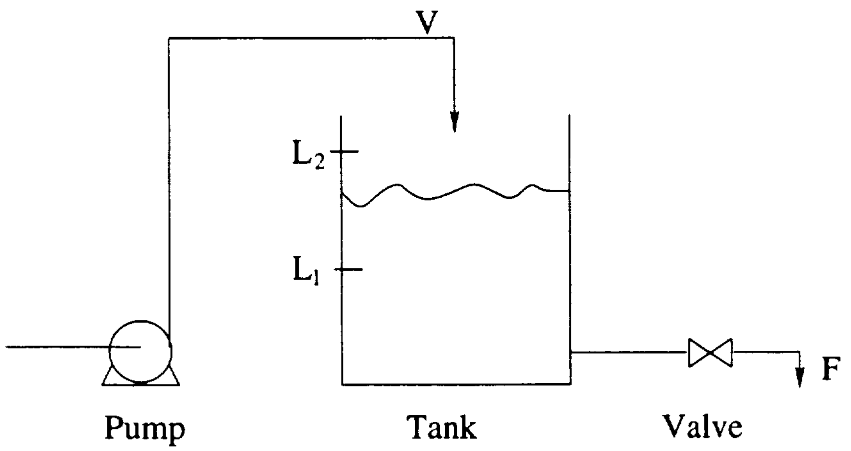

Figure 1. Water tank system.

The system is described graphically in figure 2 as a 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. 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

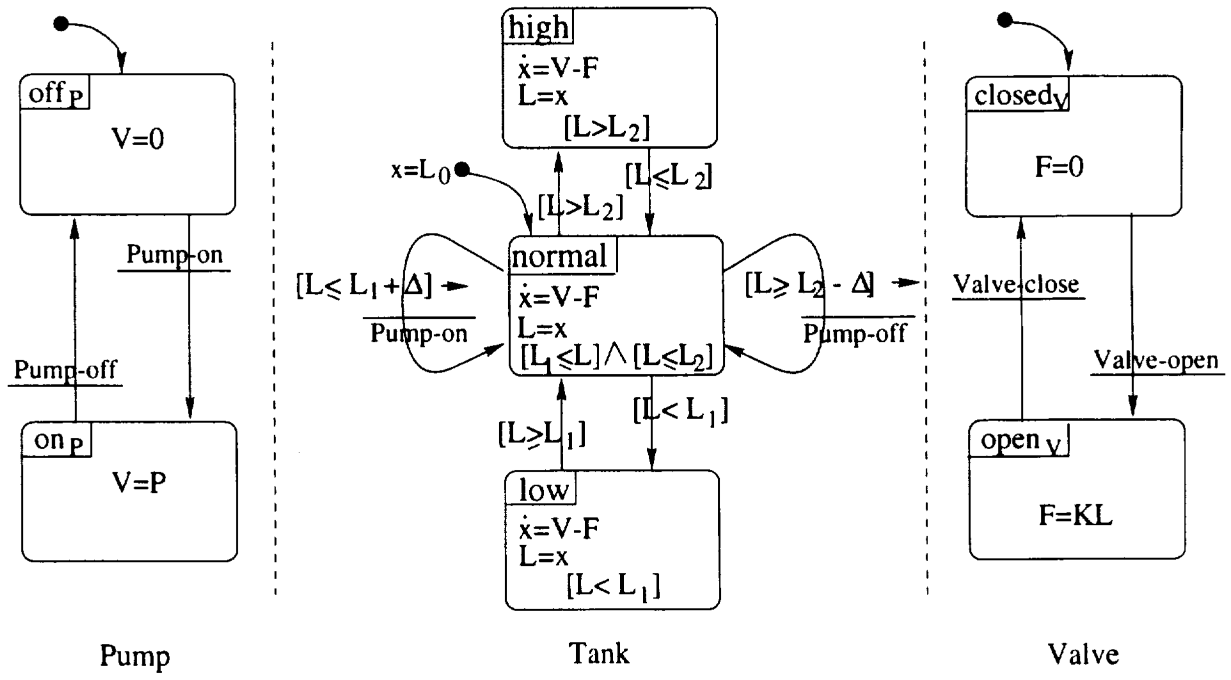

Figure 2. Water tank system CHM.

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 by $[L \leq L_1 + \Delta] \rightarrow \overline{pump - on}$ is guarded by the predicate $[L \leq L_1 + \Delta]$ 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$_P$ $>$ and $<$ on$_P$ $>$. At the vertex $<$ off$_P$ $>$, the pump is off, reflected by the vertex output $V = 0$. Similarly, at the vertex $<$ on$_P$ $>$, 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$_V$ $>$ or $<$ closed$_V$ $>$. Transitions 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 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. (2) Signal sharing, by which outputs (output signals) of a vertex are available as vertex inputs to any other parallel EHM.

3.2 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 $\underline{\sigma}$ is absent, then the transition takes place immediately upon G becoming true; if $\overline{\sigma'}$ is absent, then no output event is transmitted; if G is absent, the guard is always true and the transition will be triggered by the input event $\underline{\sigma}$; and 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).

- (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 the vertices be completely guarded with each possible invariant violation. That is, every invariant violation implies that some guard becomes true and the associated transition is input event–free in the sense that it has the form $(q, G \rightarrow \overline{\sigma'}, q', x^0_{q'})$. (It is, in principle, permitted that more than one guard become true at the same instant. In this 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 do require that, upon entry to q', the invariant $I_{q'}$ not be violated. It is, however, possible that upon entry to q' one of the guards at q' is already true. In this case, the EHM will immediately exit q' and go to the vertex specified by the guards. Such a transition is considered instantaneous. Naturally, we only allow finite chains of such instantaneous transitions. That is, the guards must be such that no sequence of instantaneous transitions will form a loop.

In this paper we shall study a restrictive class of hybrid machines by making 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^L_q and an upper limit k^U_q, that is, $f_q(x_q, u_q) \in [k^L_q, k^U_q]$.

An execution of the EHM is a sequence

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

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:

$$x_{q0}, x_{q1}, x_{q2}, \dots$$

where $x_{qi} = \{x_{qi}(t) : t \in [t_i, t_{i+1})\}$.

- 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 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 note 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.

3.3 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 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 (ref. 18), 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*)

$$S_i > C_i \text{ or } S_i < C_i$$

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

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 $< q^1_{i_1}, q^2_{i_2}, ..., q^n_{i_n} >$ is the source configuration, $< q^1_{i'_1}, q^2_{i'_2}, ..., q^n_{i'_n} >$ the target configuration, and l the label that triggers the transition. l can be either an event or a guard (becoming true). Thus, if $l = \sigma$ 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 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 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.

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, which are labeled by guards only, and (2) event transitions, which are labeled by events.

The transitions are considered to occur instantaneously, and concurrent vertex changes in parallel components occur at exactly 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

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

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

4 CONTROL

4.1 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 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 (refs. 11 and 12) 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 here, 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.

4.2 Control Synthesis

As stated, our control objective is to ensure that the system CHM never enters 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.

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

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

where $G_{ij} = (S_{ij} > C_{ij})$ or $G_{ij} = (S_{ij} < C_{ij})$, representing some semi-open intervals.[1] Without loss of generality, we shall assume that the invariant is violated if and only if one or more of the guards is true. (Otherwise, we can conjoin with the invariant the negation of the guards.)

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

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

2. Illegal event transitions that lead to illegal configurations

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

[1]More generally, we only require that guards leading to illegal configurations be described by semi-open intervals.

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'. However, such a transition may be legally triggered only if the invariant $I_{q'}$ is 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 guaranteed when $\underline{\sigma}$ is triggered.) Thus, let us define $wp(q, \underline{\sigma}, q')$ to be the weakest precondition under which the transition $(q, \underline{\sigma}, q')$ 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 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

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

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

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

Clearly, the dynamics of and the transitions leaving and entering the configurations q_1 and q_2 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 $\neg wp(q, \underline{\sigma}, q')$, and from q_2 to q_1 with the guard $wp(q, \underline{\sigma}, q')$.

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

$(q_1, \neg wp(q, \underline{\sigma}, q'), q_2)$ 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. To find the condition under which $\underline{\sigma}$ needs to be forced, we note that, by our assumption on invariants, I_{q_1} will have the conjunctive normal form

$$I_{q_1} = (P_{11} \vee ... \vee P_{1l_1}) \wedge ... \wedge (P_{m1} \vee ... \vee P_{ml_m})$$

where $P_{ij}=(S_{ij} \geq C_{ij})$ or $P_{ij}=(S_{ij} \leq C_{ij})$, representing semi-closed intervals. Therefore, we would like to force $\underline{\sigma}$ exactly on the boundary. Recall that, by assumption, the shared variables S_i are rate-bounded; that is, $\dot{S}_i \in [r_i^L, r_i^U]$, where r_i^L and r_i^U are the lower and upper bounds, respectively. Thus, for a predicate $P = (S_i \leq C_i)$, 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 $P = P_1 \vee P_2$,

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

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

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

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

$$I_{q_1} = I_q \wedge (\vee_{(q,\underline{\sigma},q') \in ET_g(q,Q_b)} wp(q, \underline{\sigma}, q'))$$
$$I_{q_2} = I_q \wedge \neg(\vee_{(q,\underline{\sigma},q') \in ET_g(q,Q_b)} wp(q, \underline{\sigma}, q'))$$

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

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

where $wp(q, G, q')$ is the weakest precondition under which the transition (q, G, q') will not violate the invariant $I_{q'}$ upon entry to q'. $wp(q, G, q')$ is calculated in the same way as $wp(q, \underline{\sigma}, q')$.

To find the condition under which a dynamic transition $(q, G, q') \in DT_b(q, Q_b)$ will be preempted by another dynamic transition (i.e., (q, G, q') will not take place), let us consider first the time at which a predicate will become true. We begin by considering an atomic formula

$$P{=}(S_i > C_i)$$

Suppose that at a given instant t at which $S_i(t) = S_i$, P is false; that is, $S_i{\le}C_i$. 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, as before, $r_i{}^L$ and $r_i{}^U$ are the lower and upper bounds of $\dot{S}$, respectively.

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

$$P{=}(S_i < C_i)$$

then if, at the instant t, P is true, $T_{min}(true(P)) = T_{max}(true(P)) = 0$; 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 $T_{min}(true(P)) = T_{max}(true(P)) = \infty$.

Now, 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 that $G \wedge wp(q, G, q')$ will become true is $T_{min}(G \wedge wp(q, G, q'))$ and the latest time that I_q will become false is given by $T_{max}(false(I_q)) = T_{max}(true(\neg I_q))$. It is clear that to ensure that the transition (q, G, q') will not take place, it must be required that the following preemptive condition[2]

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

be satisfied. Therefore, we shall split the configuration q into two subconfigurations q_1 and q_2, by partitioning the invariant I_q as

$$\begin{aligned} I_{q1} &= I_q \wedge pc(q, G, q') \\ I_{q2} &= I_q \wedge \neg pc(q, G, q') \end{aligned}$$

Clearly, the dynamics of and the transitions leaving and entering the configurations q_1 and q_2 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 shall split q into q_1 and q_2 as follows.

$$\begin{aligned} I_{q1} &= I_q \wedge (\wedge_{(q,G,q') \in DT_b(q,Q_b)} pc(q, G, q')) \\ I_{q2} &= I_q \wedge \neg(\wedge_{(q,G,q') \in DT_b(q,Q_b)} pc(q, G, q')) \end{aligned}$$

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

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.[3]

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

The condition under which a legal event transition $(q,\underline{\sigma},q')$ needs to be forced is now given by[4]

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

Note that if we adopt the convention that

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

then this general case covers all the cases above, including the case when $DT_b(q,Q_b) = \emptyset$.

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

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

[3]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 $I_{q_2} = I_q$.

[4]There is a possible complication if the newly defined guards form an instantaneous loop of consecutive transitions. If this occurs, further analysis will be required.

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

5. For all $q \in PC$ do

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

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

$$Q^c := PC;$$
$$\Sigma^c := \Sigma \cup \{\tilde{\sigma} : \sigma \in \Sigma\};$$
$$D^c := \emptyset;$$

9. Define transitions

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

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 $\tilde{\underline{\sigma}}$ 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 no sequence of instantaneous transitions forms a loop, then the controller synthesized is the 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 it is possible in $CHM||C||\tilde{D}$.

Proof

Since Algorithm 1 terminates in a finite number of steps and no sequence of instantaneous transitions forms 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 $q_0, q_1, \ldots, q_{n-1} \in Q^c$ 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 $CHM||D$

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

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:

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

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.

5 STEAM BOILER EXAMPLE

In this section, we shall illustrate application of the control synthesis algorithm developed in the previous section by synthesizing a controller for the familiar steam boiler example that was proposed in reference 19 as a benchmark problem for modeling and verification of hybrid systems (see also, e.g., references 20 and 21). This example was proposed as a benchmark problem because it has many essential properties that are found in some commonly used industrial processes, such as chemical reactors, oil refineries, etc.

We use a simplified model of the steam boiler described in reference 19. Some parameters are set at the same values as in reference 20. This simplified model captures the essence of the control problem addressed in this paper.

The steam boiler consists of a water tank (boiler) equipped with two pumps (instead of four pumps as in reference 19). Each pump can supply water to the boiler at the rate of 4 liters/sec. The pump can be switched on (event *start_i*) and off (event *stop_i*) by a controller. Due to the fact that the pump cannot balance the presure inside the boiler instantaneously, there is a five-second delay before water starts pouring into the boiler after the pump is switched on.

Steam is generated by an unmodeled mechanism. The rate at which steam is generated is therefore nondeterministic. But we do know that the rate is bounded between 0 and 6 liters/sec.

The control objective is to maintain the water level L in the boiler between the minimal level of 5 liters and the maximal level of 220 liters. This is achieved by turning the two pumps on and off. Since we are interested in synthesizing the minimally restrictive controller, our controller will accept (i.e., permit) all behaviors (turning pumps on and off) that do not imply possible violation of the level constraints and will intervene by forcing the pumps (on or off) only whenever it is absolutely necessary to do so in order to guarantee constraint satisfaction.

The controller can sample the water level in the boiler only every five seconds. Since this implies sampled decision making, there is no loss in generality in assuming that control (turning the pumps on and off) can only be applied at the sampling instants.

In summary, the steam boiler to be controlled is modeled by the CHM in figure 3.

As stated above, the parameters are given by

$$\begin{aligned} P_1 &= 4, \quad P_2 = 4, \\ V_L &= 0, \quad V_H = 6, \\ L_L &= 5, \quad L_H = 220 \end{aligned}$$

Without changing the nature of the problem, but to avoid nondeterminism in the controller, we shall assume that Pump 1 will be turned on before Pump 2 can be turned on; and Pump 1 cannot be turned off before Pump 2 is turned off. Therefore, the pump logic is shown in figure 4.

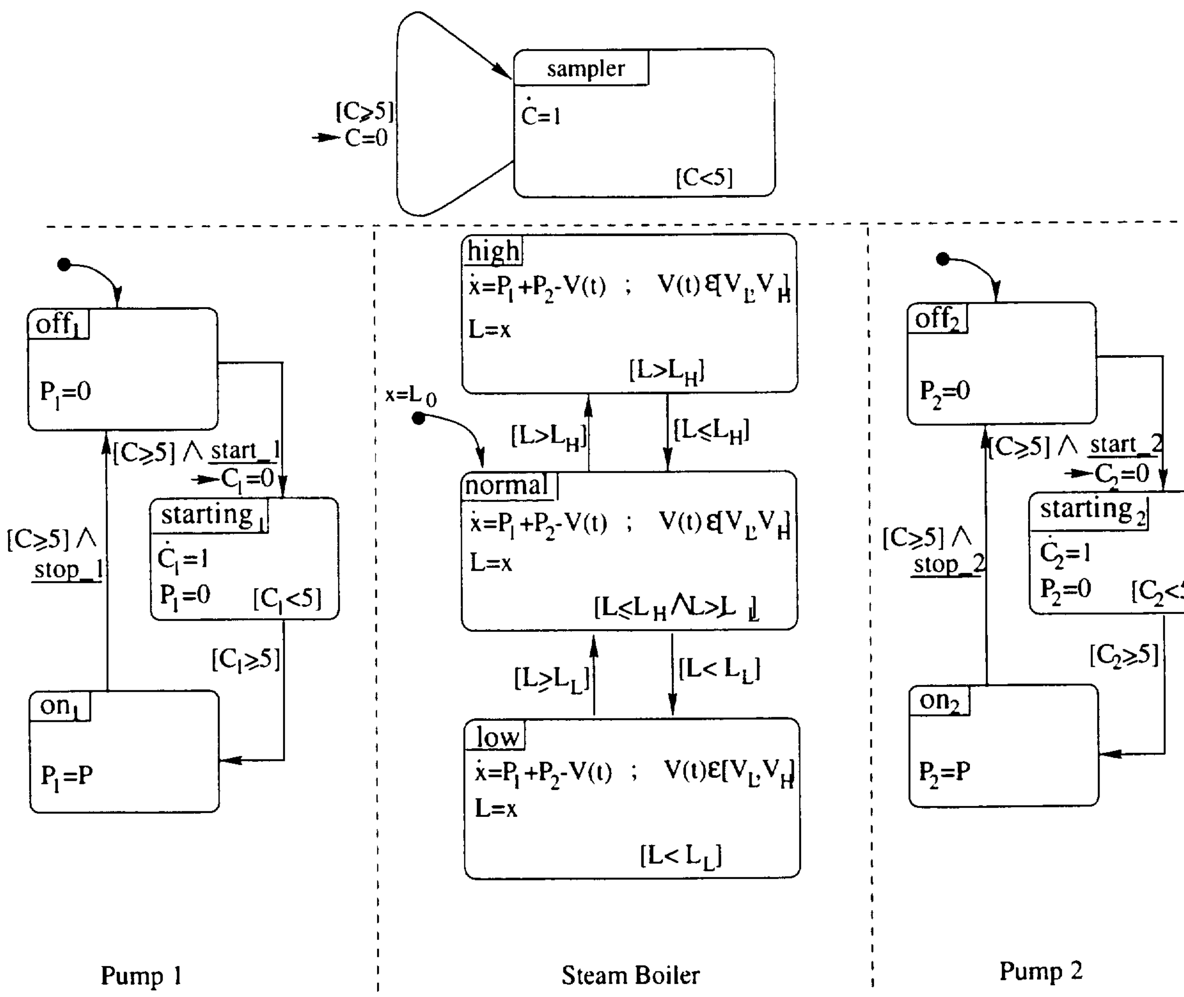

Figure 3. Steam boiler system.

Thus, the configurations of the CHM to be controlled can be denoted by the legal configurations

$$
\begin{aligned}
q^1 &=< off_1, off_2, normal > \\
q^2 &=< starting_1, off_2, normal > \\
q^3 &=< on_1, off_2, normal > \\
q^4 &=< starting_1, starting_2, normal > \\
q^5 &=< on_1, starting_2, normal > \\
q^6 &=< on_1, on_2, normal >
\end{aligned}
$$

and illegal configurations where *normal* ($[L \geq 5] \wedge [L \leq 220]$) is replaced by *high* ($[L > 220]$) or *low* ($[L < 5]$). That is,

$$Q_b =< high > \cup < low >$$

Because of the delays in turning the pumps on and the delays caused by sampling, there are configurations in $< normal >$ from which unavoidable dynamic transitions may lead to illegal configurations in Q_b. Therefore, we must partition $< normal >$ properly using the synthesis algorithm.

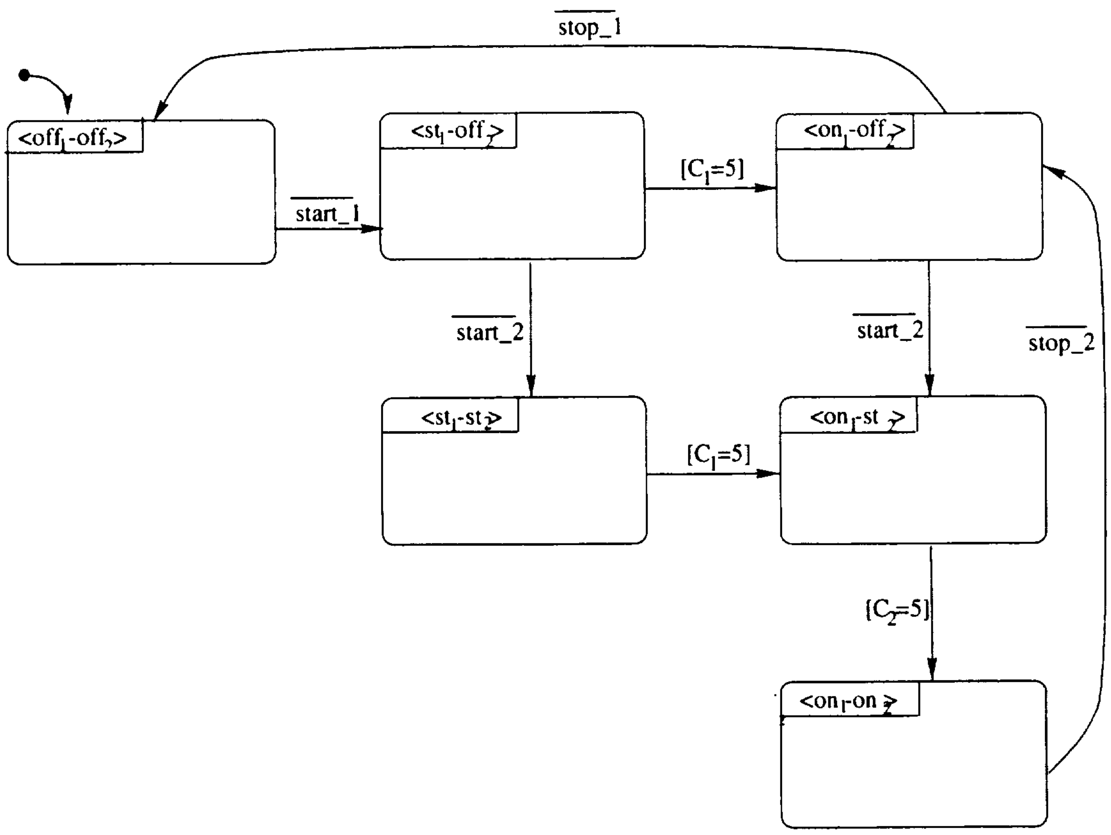

Figure 4. Pump logic.

Before applying the algorithm, we first replace the guarded event transitions by dynamic and event transitions. Also, note that since $C_1 = C_2 = C$ whenever they are not equal to 0 or 5, only one clock is sufficient (to be denoted by C). Thus, the equivalent CHM is shown in figure 5, where, for clarity, the illegal configurations are not drawn.

We shall only illustrate how the algorithm performs on q^6 and $q^{6'}$, where

$$I_{q^6} = [L \geq 5] \wedge [L \leq 220] \wedge [C < 5]$$
$$I_{q^{6'}} = [L \geq 5] \wedge [L \leq 220] \wedge [C \geq 5]$$

By our algorithm,

$$wp(q^{6'}, \underline{stop\_2}, q^{3'}) = [L \geq 5] \wedge [L \leq 220]$$

Therefore, $q^{6'}$ will not be split. On the other hand, q^6 will be split as follows (note that at q_6, $\dot{L} \in [2, 8]$).

$$\begin{aligned}
&pc(q^6, [L > 220], < illegal >) \\
&= (T_{min}([L > 220]) > T_{max}([L < 5] \vee [L > 220] \vee [C \geq 5])) \\
&= ((220 - L)/8 > min\{\infty, (220 - L)/8, 5 - C\}) \\
&= ((220 - L)/8 > (5 - C)) \\
&= (L < 180 + 8C)
\end{aligned}$$

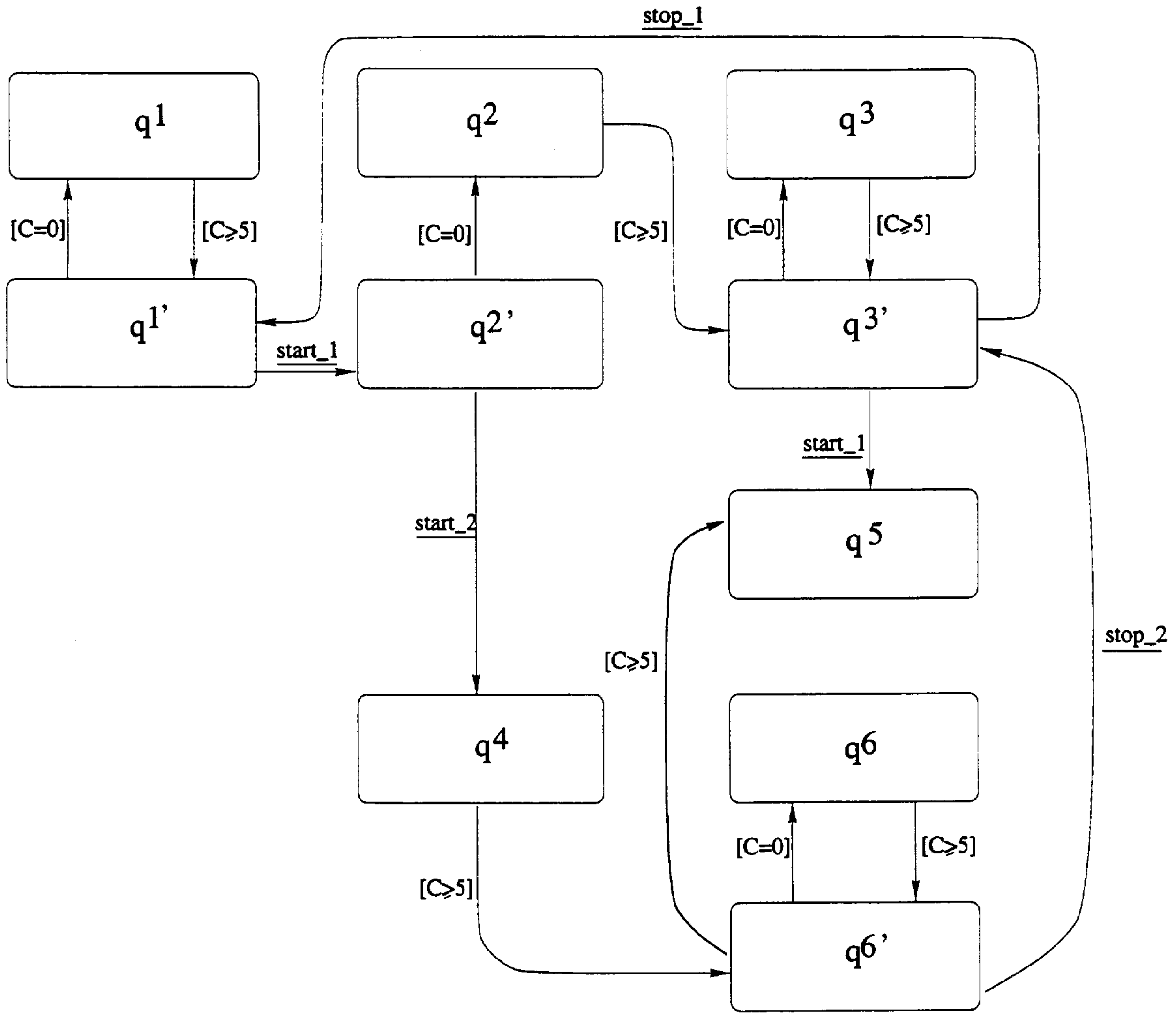

Figure 5. Composite hybrid machine.

Similarly,

$$\begin{aligned}&pc(q^6, [L < 5], < illegal >)\\&= (T_{min}([L < 5]) > T_{max}([L < 5] \vee [L > 220] \vee [C \geq 5]))\\&= (\infty > T_{max}([L < 5] \vee [L > 220] \vee [C \geq 5]))\\&= true\end{aligned}$$

Therefore, q^6 will be split into q^6_1 and q^6_2 with invariants

$$\begin{aligned}I_{q^6_1} &= I_{q^6} \wedge pc(q^6, [L > 220], < illegal >) \wedge pc(q^6, [L < 5], < illegal >)\\&= [L \geq 5] \wedge [L \leq 220] \wedge [C < 5] \wedge [L < 180 + 8C]\\&= [L \geq 5] \wedge [C < 5] \wedge [L < 180 + 8C]\\I_{q^6_2} &= [L \geq 5] \wedge [L \leq 220] \wedge [C < 5] \wedge [L \geq 180 + 8C]\end{aligned}$$

In the next iteration, $q^{6'}$ will be analyzed as follows. There are five transitions leaving $q^{6'}$:

$$
\begin{array}{l}
(q^{6'}, [C{\geq}5], q^6_1) \\
(q^{6'}, [C{\geq}5], q^6_2) \\
(q^{6'}, [L < 5], < illegal >) \\
(q^{6'}, [L > 220], < illegal >) \\
(q^{6'}, stop\_2, q^{3'})
\end{array}
$$

It can be calculated that

$$
\begin{array}{l}
pc(q^{6'}, [C{\geq}5], q^6_2) \\
= (T_{min}([C{\geq}5] \wedge [L > 180]) > T_{max}([L < 5] \vee [L > 220] \vee [C < 5])) \\
= (max\{0, (180 - L)/8\} > min\{\infty, (220 - L)/2, 0\}) \\
= ((180 - L/8 > 0) \\
= [L < 180] \\
pc(q^{6'}, [L < 5], < illegal >) = true \\
pc(q^{6'}, [L > 220], < illegal >) = true \\
wp(q^{6'}, stop\_2, q^{3'}) = [L \geq 5] \wedge [L \leq 220]
\end{array}
$$

Therefore $q^{6'}$ will not be split and event *stop_2* will be forced under the condition

$$
\begin{array}{l}
critical(I_{q^{6'}}) \wedge wp(q^{6'}, stop\_2, q^{3'}) \wedge \\
\quad \neg(pc(q^{6'}, [C{\geq}5], q^6_2) \wedge pc(q^{6'}, [L < 5], < illegal >) \wedge pc(q^{6'}, [L > 220], < illegal >)) \\
= ([L \leq 5] \vee [L \geq 220] \vee [C \leq 5]) \wedge [L > 180] \wedge [L \geq 5] \wedge [L \leq 220]
\end{array}
$$

Since $[C \leq 5], [L \geq 5], [L \leq 220]$ are satisfied at $q^{6'}$, the forcing will actually take place when $[L > 180]$.

Table 1 summarizes the results of the synthesis algorithm at each iteration.

Table 1. Steam boiler controller synthesis

	Initial	First iteration	Second iteration	Third iteration
q^1	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L > 35 - 6C]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L > 35 - 6C]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L > 65 - 6C]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$
$q^{1'}$	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L > 35]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L > 35]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$
q^2	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L > 35 - 6C]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L > 35 - 6C]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L > 45 - 6C]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$
$q^{2'}$	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L > 35]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L > 35]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$
q^3	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L > 15 - 2C]$ $\wedge[L \leq 200 + 4C]$ $\wedge[C < 5]$	$[L > 15 - 2C]$ $\wedge[L \leq 200 + 4C]$ $\wedge[C < 5]$	$[L > 25 - 2C]$ $\wedge[L \leq 200 + 4C]$ $\wedge[C < 5]$
$q^{3'}$	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L > 15]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L > 15]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$
q^4	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L > 35 - 6C]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L > 35 - 6C]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L > 35 - 6C]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$
q^5	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L > 15 - 2C]$ $\wedge[L < 200 + 4C]$ $\wedge[C < 5]$	$[L > 15 - 2C]$ $\wedge[L < 200 + 4C]$ $\wedge[C < 5]$	$[L > 15 - 2C]$ $\wedge[L < 200 + 4C]$ $\wedge[C < 5]$
q^6	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C < 5]$	$[L \geq 5]$ $\wedge[L < 180 + 8C]$ $\wedge[C < 5]$	$[L \geq 5]$ $\wedge[L < 180 + 8C]$ $\wedge[C < 5]$	$[L \geq 5]$ $\wedge[L < 180 + 8C]$ $\wedge[C < 5]$
$q^{6'}$	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$	$[L \geq 5]$ $\wedge[L \leq 220]$ $\wedge[C \geq 5]$

Finally, the minimally restrictive controller is shown in figure 6.

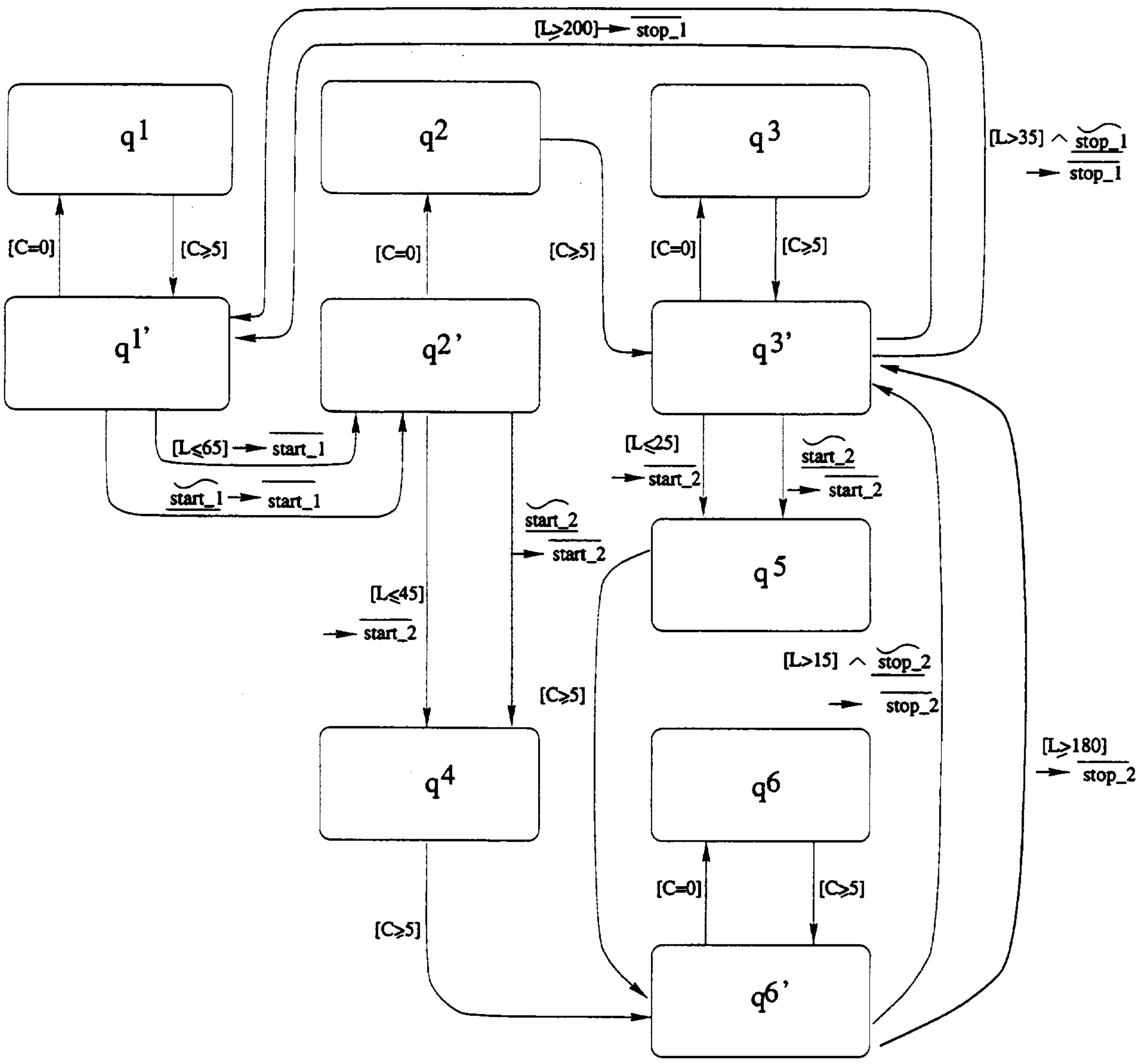

Figure 6. Steam boiler controller.

REFERENCES

1. Alur, R.; Courcoubetis, C.; Henzinger, T. A.; and Ho, P.-H.: Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. Hybrid Systems, Lecture Notes in Computer Science, vol. 736, Springer-Verlag, 1993, pp. 209–229.

2. Alur, R.; Courcoubetis, C.; Halbwachs, N.; Henzinger, T. A.; Ho, P.-H.; Nicollin, X.; Olivero, A.; Sifakis, J.; and Yovine, S.: The Algorithmic Analysis of Hybrid Systems. Theoretical Computer Science, vol. 138, 1995, pp. 3–34.

3. Branicky, M. S.: Universal Computation and Other Capabilities of Hybrid and Continuous Dynamical Systems. Theoretical Computer Science, vol. 138, 1995, pp. 67–100.

4. Brockett, R. W.: Hybrid Models for Motion Control Systems. In Essays in Control: Perspectives in the Theory and Its Applications, H. L. Trentelman and J. C. Willems, eds., Birkhauser, Boston, 1993, pp. 29–53.

5. Maler, O.; Manna, Z.; and Pnueli, A.: From Timed to Hybrid Systems. In Real Time: Theory in Practice, Lecture Notes in Computer Science, vol. 600, Springer Verlag, 1991, pp. 447–484.

6. Nerode, A.; and Kohn, W.: Models for Hybrid Systems: Automata, Topologies, Controllability, Observability. Hybrid Systems, Lecture Notes in Computer Science, vol. 736, Springer-Verlag, 1993, pp. 317–356.

7. Nicollin, X.; Sifakis, J.; and Yovine, S.: From ATP to Timed Graphs and Hybrid Systems. In Real Time: Theory in Practice, Lecture Notes in Computer Science, vol. 600, Springer-Verlag, 1991, pp. 549–572.

8. Alur, R.; and Dill, D.: Automata for Modeling Real-Time Systems. Proc. of the 17th International Colloquium on Automata, Languages and Programming, 1990, pp. 322–336.

9. Lin, F.; and Wonham, W. M.: Supervisory Control of Timed Discrete Event Systems under Partial Observation. IEEE Transactions on Automatic Control, vol. 40, no. 3, 1994, pp. 558–562.

10. Antsaklis, P. J.; Stiver, J. A.; and Lemmon, M.: Hybrid System Modeling and Autonomous Control Systems. Hybrid Systems, Lecture Notes in Computer Science, vol. 736, Springer-Verlag, 1993, pp. 366-392.

11. Ramadge, R. J.; and Wonham, W. M.: Supervisory Control of a Class of Discrete Event Processes. SIAM J. Control and Optimization, vol. 25, no. 1, 1987, pp. 206–230.

12. Ramadge, P. J.; and Wonham, W. M.: The Control of Discrete Event Systems. Proceedings of IEEE, vol. 77, no. 1, 1989, pp. 81–98.

13. Henzinger, T.; Kopke, P.; Puri, A.; and Varaiya, P.: What's Decidable About Hybrid Automata. Proc. of the 27th Annual ACM Symposium on the Theory of Computing, 1995.

14. Manna, Z.; and Pnueli, A.: Verifying Hybrid Systems. Hybrid Systems, Lecture Notes in Computer Science, vol. 736, Springer-Verlag, 1993, pp. 4–35.

15. Nicollin, X.; Olivero, A.; Sifakis, J.; and Yovine, S.: An Approach to the Description and Analysis of Hybrid Systems. Hybrid Systems, Lecture Notes in Computer Science, vol. 736, Springer-Verlag, 1993, pp. 149–178.

16. Heymann, M.; and Lin, F.: On-Line Control of Partially Observed Discrete Event Systems. Discrete Event Dynamic Systems: Theory and Applications, vol. 4, no. 3, 1994, pp. 221–236.

17. Heymann, M.; and Lin, F.: Discrete Event Control of Nondeterministic Systems. Control of Non-deterministic Systems, CIS Report 9601, Technion, Israel, 1996.

18. Heymann, M.: Concurrency and Discrete Event Control. IEEE Control Systems Magazine, vol. 10, no. 4, 1990, pp. 103–112.

19. Abrial, J.-R.: Steam-Boiler Control Specification Problem. Dagstuhl Meeting: Method for Semantics and Specification, 1995.

20. Henzinger, T. A.; and Wong-Toi, H.: Using HYTECH to Synthesize Control Parameters for a Steam Boiler. 1996. Preprint.

21. Bussow, R.; and Weber, M.: A Steam-Boiler Control Specification with Statecharts and Z. 1996. Preprint.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1997	3. REPORT TYPE AND DATES COVERED Technical Memorandum

4. TITLE AND SUBTITLE Control Synthesis for a Class of Hybrid Systems Subject to Configuration-Based Safety Constraints	5. FUNDING NUMBERS 548-40-12
6. AUTHOR(S) Michael Heymann,[†] Feng Lin,[‡] and George Meyer	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Ames Research Center Moffett Field, CA 94035-1000	8. PERFORMING ORGANIZATION REPORT NUMBER A-976702
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA TM-112196

11. SUPPLEMENTARY NOTES
Point of Contact: George Meyer, Ames Research Center, MS 262-2, Moffett Field, CA 94035-1000
(415) 604-5750
[†]Technion, Israel Institute of Technology, Haifa 32000, Israel. [‡]Wayne State University, Detroit, Michigan.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified — Unlimited Subject Category 31	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

We examine a class of hybrid systems which we call *Composite Hybrid Machines* (CHMs) that consists of the concurrent (and partially synchronized) operation of *Elementary Hybrid Machines* (EHMs).

Legal behavior, specified by a set of *illegal* configurations that the CHM may not enter, is to be achieved by the concurrent operation of the CHM with a suitably designed *legal* controller. In the present paper we focus on the problem of synthesizing a legal controller, whenever such a controller exists. More specifically, we address the problem of synthesizing the *minimally restrictive* legal controller.

A controller is minimally restrictive if, when composed to operate concurrently with another legal controller, it will never interfere with the operation of the other controller and, therefore, can be composed to operate concurrently with any other controller that may be designed to achieve liveness specifications or optimality requirements without the need to reinvestigate or reverify 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 $x \leq x_0$ or $x \geq x_0$). We present an algorithm for synthesis of the minimally restrictive legal controller.

We demonstrate our approach by synthesizing a minimally restrictive controller for a steam boiler (the verification of which recently received a great deal of attention).

14. SUBJECT TERMS Automatic control, Hybrid systems, Safety	15. NUMBER OF PAGES 31
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT