[21] A. J. van der Schaft and J. M. Schumacher, 1998. Complementarity Modeling of Hybrid Systems. *IEEE Transactions on Automatic Control, 43(4)*, pp. 483-490.

[22] H. Wong-Toi, 1997. The synthesis of controllers for linear hybrid automata. *Proc. of 36th IEEE Conference on Decision and Control*, pp. 4607-4613.

[8] 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.*

[9] T. A. Henzinger and P. W. Kopke, 1997. Discrete time control for rectangular hybrid automata. *Proc. of 24th International Colloquium on Automata, Languages and Programming.*

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

[11] M. Heymann, F. Lin and G. Meyer, 1997. Synthesis of Minimally Restrictive Controllers for a Class of Hybrid Systems. in A. Nerode (Ed.), Hybrid Systems IV, *Lecture Notes in Computer Science*, Springer Verlag.

[12] M. Heymann, F. Lin and G. Meyer, 1998. Synthesis and viability of minimally interventive legal controllers for hybrid systems. *Discrete Event Dynamic Systems: Theory and Applications, 8(2)*, pp. 105-135.

[13] D. Kapur and R. K. Shyamasundar, 1997. Synthesizing Controllers for Hybrid Systems. *Lecture Notes in Computer Science 1201*, Springer Verlag, pp. 361-375.

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

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

[16] David L. Pepyne and Christos G. Cassandras, 1998. Modeling, Analysis, and Optimal Control of a Class of Hybrid Systems. *Discrete Event Dynamic Systems: Theory and Applications, 8(2)*, pp. 175-202..

[17] A. Pnueli, 1997. Verifying liveness properties of reactive systems. *Lecture Notes in Computer Science 1201*, Springer Verlag.

[18] J. Raisch and S. D. O'Young, 1998. Discrete Approximation and Supervisory Control of Continuous Systems. *IEEE Transactions on Automatic Control, 43(4)*, pp. 569-573.

[19] 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.

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

Q^c will be disabled by C and all the dynamic transitions leaving Q^c will be preempted by the forced transitions $(q, \vee_{(q,G,q')\in DT(q,Q-Q^c)} G \rightarrow \overline{\sigma}, q')$.

To prove Part 2, observe first that in view of the fact that Algorithm 1 progressively adding live configurations to Q^c until no further addition is possible. Therefore, a controller will be live only if it does not exceed the configurations and invariants of C. Assume that

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

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

■

References

[1] 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.

[2] 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.

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

[4] Yitzhak Brave and Michael Heymann, 1990. On Stabilization of Discrete-Event Processes, *International J. on Contr., Vol. 51, No. 5*, pp.1101-1117.

[5] Yitzhak Brave and Michael Heymann, 1993. On Optimal Attraction in Discrete-Event Processes, *Information Sciences, Vol. 67*, pp.245-267.

[6] 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.

[7] J.E.R. Cury, B. H. Krogh, and T. Niinomi, 1998. Synthesis of Supervisory Controllers for Hybrid Systems Based on Approximating Automata. *IEEE Transactions on Automatic Control, 43(4)*, pp. 564-568.

13. Define transitions:

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

14. End.

It is readily seen that the configurations of the controller C consist of the set of all live configurations with their invariants as calculated during the iteration phase of the algorithm. The controller C has no continuous dynamics, so it is "driven" by the dynamics of the CHM. The transitions of C are then triggered when the boundary of some unlive dynamic transitions is reached. The controller thus synthesized is minimally interventive. Its interaction with the system is restricted to the exclusive objective of preventing the system from violating the liveness constraints. The controller is augmented to allow "environment-triggered" transitions labeled by $\underline{\tilde{\sigma}}$, which are allowed to be generated by the environment (possibly by an additional controller) and trigger transitions in C and hence in the CHM whenever such transitions are not disabled or disallowed by C. C will force an (event) transition only if otherwise the live constraint could be violated. We will illustrate the algorithm by the following example.

Note that the controlled system $CHM||C$ is also an open system (but with input events $\underline{\sigma}$ replaced by $\underline{\tilde{\sigma}}$). Therefore, we can combine $CHM||C$ with other controller D as follows. First, all the output-events $\overline{\sigma}$ in D are replaced by $\overline{\tilde{\sigma}}$ to obtain $\tilde{D}$. Then the composite controlled system is given by

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

The following theorem shows the correctness of our algorithm.

Theorem 2 If Algorithm 1 terminates in a finite number of steps, then the controller synthesized is a minimally interventive live controller in the following sense.

1. $CHM||C$ is live.

2. For any live controller D, every run of $CHM||D$ has a corresponding run in $CHM||C||\tilde{D}$.

Proof

Since Algorithm 1 terminates in a finite number of steps, by Theorem 1, from a live configuration (those that are in Q^c), the system can always be forced to its final configurations in bounded time. Therefore, to prove Part 1, it is sufficient to show that $CHM||C$ can never go to a configuration outside Q^c. This is obvious because all the event transitions leaving

If $I_{q_1} = false$, then

$skip$;

If $I_{q_2} = false$, then

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

Else do

begin

$NLC := NLC \cup \{q_1\}$;
$PC := PC \cup \{q_2\}$;
$E := E \cup \{(q_1, \neg pd(q, LC), q_2), (q_2, pd(q, LC), q_1)\}$;

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

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

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

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

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

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

11. If $LC \neq NLC$, then

$LC := NLC$;

Go to 4;

Construction of C

12. Define vertices, events, dynamics and invariants:

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

begin

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

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

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

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

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

end;

end;

7. If $Repeat = true$, go to 4;
8. For all $(q, G, q') \in E$ do

$G := cl(G)$;

9. For all $q \in PC$ do

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

10. For all $q \in PC$ do

begin

If $ET(q, Q_f) \neq \emptyset$ or $DT(q, Q - Q_f) = \emptyset$, then

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

Else do

Begin

$G_g := \vee_{(q,G,q') \in DT(q,LC)} G$;
$G_b := \vee_{(q,G,q') \notin DT(q,LC)} G$;
$pd(q, LC) := T_{max}(true(G_g)) < T_{min}(true(G_b))$;;
$I_{q_1} := I_q \wedge pd(q, LC)$;
$I_{q_2} := I_q \wedge \neg pd(q, LC)$;

Initialization

1. Set of live configurations

   $$LC := Q_f;$$

2. New set of live configurations

   $$NLC := Q_f;$$

3. Set of pending configurations

   $$PC := Q - Q_f;$$

Iteration

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

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

5. Let

   $$Repeat := false;$$

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

   begin

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

   If $I_{q_1} = false$, then

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

   If $I_{q_2} = false$, then

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

   Else do

by $\underline{\sigma}$ will be replaced by a guarded event transition $wp(q, \underline{\sigma}, q') \wedge \underline{\sigma}$, which in turn will be decomposed into event and dynamic transitions.

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

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

When the iterations terminate (i.e., when there are no more live configurations to be added), the resulting live configurations (and their invariant) have the following property: Either all the possible dynamic transitions are live or there exists at least one event transition to a live configuration that can be forced. Although the forcing can be done at any time when the CHM is in the corresponding configuration, the minimally interventive live controller will not force the event transition, unless a dynamic transition to a blocked configuration (blocked dynamic transition) is about to take place. In other words, the forcing will occur when a blocked guard of a blocked dynamic transition becomes true. We assume that the live guards have precedence over blocked guards. Hence, the forcing will take precedence over the blocked dynamic transition.

In summary, we present our synthesis algorithm as follows.

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

- The model of the system

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

- The set of final configurations $Q_f \subseteq Q$.

Output

- The controller

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

Theorem 1 The precedent condition $pd(q, Q_f)$ is true if and only if for any trajectory $x(t)$ in any run,

$$T(G_g(x(t))) < T(G_b(x(t))).$$

Proof

If $pd(q, Q_f)$ is true, that is,

$$\max_{x(t)} T(G_g(x(t))) < \min_{x(t)} T(G_b(x(t))),$$

then, clearly,

$$(\forall x(t))T(G_g(x(t))) < T(G_b(x(t))).$$

On the other hand, if $pd(q, Q_f)$ is not true, then

$$T_{max}(true(G_g)) \geq T_{min}(true(G_b)).$$

Let $x(t)$ be the trajectory achieving $T_{min}(true(G_b))$. Clearly, G_b will become true earlier that G_g along the trajectory $x(t)$, otherwise the system will exit q when G_g becomes true and hence $x(t)$ will not be a valid trajectory. Therefore,

$$(\exists x(t))T(G_g(x(t))) \geq T(G_b(x(t))),$$

a contradiction.

■

The above procedure, of identifying live configurations and calculating live subconfigurations, must be repeated. This is because a configuration from which a live configuration can be forced to be reached in bounded time, is also live. To repeat the procedure, let us consider a transition (q, l, q'). Suppose that q' has been split into its live subconfiguration q'_1 and its unlive subconfiguration q'_2. Then transition from q into q'_1 (rather than into q'_2) depends on satisfaction, upon entry into q', of the invariant $I_{q'_1}$ (rather than $I_{q'_2}$). Thus, let us define $wp(q, l, q')$ to be the weakest precondition under which the transition (q, l, q') will not violate the invariant $I_{q'}$ upon entry into q'. Since some of the shared variables that appear in $I_{q'}$ are possibly (re-) initialized upon entering q' because $x_{q'}$ is (re-)initialized, the condition $wp(q, l, q')$ 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 $s_i = h_j(x^0_{q'}, u_{q'})$.

Using this weakest precondition, we can replace each transition (q, l, q') by its equivalence $(q, wp(q, l, q') \wedge l, q')$. That is, a dynamic transition with guard G will be replaced by a dynamic transition with guard $G \wedge wp(q, G, q')$. Similarly, an event transition triggered

the configuration q (or equivalently, its invariant I_q) into live subconfiguration q_1 and unlive subconfiguration q_2.

To describe this partition formally, let us define, for $q \in Q$ and $Q' \subseteq Q$, the set of event transitions from q to Q':

$$ET(q, Q') = \{(q, \underline{\sigma}, q') \in E : q' \in Q'\}.$$

Similarly, the set of dynamic transitions from q to Q' is

$$DT(q, Q') = \{(q, G, q') \in E : q' \in Q'\}.$$

As we said, if $ET(q, Q_f) \neq \emptyset$ or $DT(q, Q - Q_f) = \emptyset$, then q is live. Otherwise, we must partition the invariant I_q into the live part I_{q_1} and the unlive part I_{q_2} as follows.

We first consider the time at which a predicate P will become true. Thus, let $T(P(x(t)))$ $(=T(true(P(x(t)))))$ be the time at which P becomes true for the first time along the trajectory $x(t)$. Since our goal is to guarantee that the liveness specification will not be violated under any condition, we must consider the maximal and minimal values of $T(P(x(t)))$ when evaluated over all possible trajectories of all possible runs. Thus, let us define

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

These maximum and minimum values can be calculated from the expression of P and the associated dynamics, which is discussed in [10].

Let

$$G_g = \vee_{(q,G,q') \in DT(q,Q_f)} G$$
$$G_b = \vee_{(q,G,q') \notin DT(q,Q_f)} G.$$

We define a precedent condition as

$$pd(q, Q_f) = T_{max}(true(G_g)) < T_{min}(true(G_b)).$$

We will now split the configuration q into live subconfigurations q_1 and unlive subconfiguration q_2, by partitioning the invariant I_q as

$$I_{q_1} = I_q \wedge pd(q, Q_f)$$
$$I_{q_2} = I_q \wedge \neg pd(q, Q_f).$$

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 all the dynamic transitions in $DT(q_1, Q - Q_f)$ are now impossible (because the dynamic transitions in $DT(q_1, Q_f)$ will take precedence). Also the transition from q_1 to q_2 is dynamic with the guard $\neg pd(q, Q_f)$, and from q_2 to q_1 with guard $pd(q, Q_f)$.

The justification for the above partition is given in the following

4 Control

In this section, we study how to control a hybrid system to achieve Bounded-rate liveness. 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, in this case, to satisfy the liveness constraints. A controller that achieves this objective is then said to be *live*.

In this paper, we shall consider only restricted interaction between the controller and the CHM by permitting the controller to interact 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. Furthermore C can control all the event transitions in the CHM. That is, all the (externally triggered) event transitions are available to the controller.

The assumption that C can control all the event transitions in the CHM 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 [19] [20] to deal with uncontrollable event transitions.

Obviously, there may exist many live controllers with different degree of restrictiveness. A live controller C is said to be *less interventive* (or restrictive) than another live controller C' if every run permitted by C' is also permitted by C. A live controller is said to be *minimally interventive* if it is less interventive than any live controller. In most cases, we are interested in the minimally interventive live controller.

To synthesize such a controller, we would like to find all the configurations from which the system can be forced to reach the final configurations Q_f in bounded time (we call these configurations live). We start with all the neighboring configurations of Q_f (that have at least one transition leading to Q_f). For any neighboring configuration q, if it has an event transition leading to Q_f, then clearly q is live. If no such event transitions exist, then we must consider all the dynamic transitions leaving q. If all the dynamic transitions go to Q_f, then q is again live. Otherwise, some of these dynamic transitions go to Q_f. We take the disjunction of their guards and denote it G_g. The remaining dynamic transitions do not go to Q_f. We denote the disjunction of their guards by G_b. Clearly, q will be live if and only if G_g is guaranteed to become true before G_b becomes true. This gives a way to partition

A still weaker definition of liveness is given by

Definition 3 *(Finite-time liveness for closed systems)*

A closed system is *finite-time live* if every possible run reaches its set of final configurations from the initial configuration in finite time. ■

Clearly, if a closed system is bounded-time live or finite-time live, then it can always reach its set of final configurations from any (reachable) configuration of the system in bounded or finite time, respectively.

The runs of open systems may depend on input events to be triggered by the environment. Thus, we cannot insist that open systems reach their final configurations in fixed, bounded or finite time without considering input events. Therefore, the liveness definitions need to be modified as follows.

Definition 4 *(Fixed-time liveness for open systems)*

For a fixed time T, an open system is *T-live* if every possible run can be forced (by the environment or user) to reach its set of final configurations from the initial configuration within T units of time. ■

Definition 5 *(Bounded-time liveness for open systems)*

An open system is *bounded-time live* if there exists a finite bound T such that every possible run can be forced (by the environment or user) to reach its set of final configurations from the initial configuration within T units of time. ■

Definition 6 *(Finite-time liveness for open systems)*

An open system is *finite-time live* if any run of the system can be forced (by the environment or user) to reach its set of final configurations in finite time. ■

Clearly, if a system is bounded-time live, then there exists an infimal time bound T_{inf} such that for all $T > T_{inf}$, the system is T-live.

In the present paper we shall consider only hybrid-machines that satisfy the following

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 v_q^L and an upper limit v_q^U; that is, the only information given about $f_q(x_q, u_q)$ is that $f_q(x_q, u_q) \in [v_q^L, v_q^U]$. ■

Under this assumption, we consider only *rate bounded* hybrid machines in which all the rates are bounded by closed intervals. Such systems are either bounded-time live or not live. Therefore, in the remainder of the paper, by liveness we shall simply mean bounded-time liveness. As we stated in the introduction, fixed time liveness need not be considered further, because it can always be viewed as a special case of safety.

where

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

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

3 Liveness

In our earlier work [10] [12] [11], we developed a synthesis method for designing a safety controller that guarantees the controlled system never to exit a set of specified legal (safety) configurations. Furthermore, the controller was designed to be minimally interventive in the sense that it interferes with the system's operation only when safety violation is otherwise inevitable.

In this paper, our objective is to synthesize a liveness controller. To define liveness, we first specify a set of marked or *final* configurations $Q_f \subseteq Q$ in the CHM. Liveness is then regarded to be the ability to reach this set final configurations as discussed below.

To define liveness formally, we must classify hybrid systems into *closed* systems and *open* systems. A closed system accepts no input events from the environment, and all its transitions are triggered dynamically in its EHMs. Therefore, in the CHM model of a closed system, all transitions are dynamic transitions. On the other hand, an open system accepts input events from its environment, and its CHM model includes event transitions.

Definition 1 *(Fixed-time liveness for closed systems)*

For a fixed time T, a closed system is *T-live* if every possible run reaches its set of final configurations from the initial configuration within T units of time. ■

A weaker version of liveness is the following

Definition 2 *(Bounded-time liveness for closed systems)*

A closed system is *bounded-time live* if there exists a finite bound T such that every possible run reaches its set of final configurations from the initial configuration within T units of time. ■

EHMs. A shared variable s_i can be the output of at most one EHM. The set of shared variables defines a signal space $S = \{[s_1, s_2, ..., s_m] \in \mathcal{R}^m\}$.

Transitions are synchronized by an input/output synchronization formalism. That is, if an output-event $\overline{\sigma}$ is either generated by one of the EHMs or received from the environment, then all EHMs for which $\underline{\sigma}$ is an active transition label (i.e., $\underline{\sigma}$ is defined at the current vertex with an absent guard or a true guard) will execute $\underline{\sigma}$ (and its associated transition) concurrently with the occurrence of $\overline{\sigma}$. A specific output-event can be generated by at most one EHM.

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).

A transition

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

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[5]. Thus, if $l = \underline{\sigma}$ is an event (generated by the environment), then either $q^j_{i'_j} = q^j_{i_j}$ if $\underline{\sigma}$ is not active at $q^j_{i_j}$, or $q^j_{i'_j}$ is such that $(q^j_{i_j}, \underline{\sigma} \rightarrow \overline{\sigma'}, q^j_{i'_j}, x^0_{q^j_{i'_j}})$ is a transition (edge) in E^j. On the other hand, if $l = G$ is a guard, then there must exist a transition $(q^m_{i_m}, G \rightarrow \overline{\sigma'}, q^m_{i'_m}, x^0_{q^m_{i'_m}})$ in some EHM^m, and for $j \neq m$, $q^j_{i'_j} = q^j_{i_j}$. The event $\overline{\sigma'}$ (generated as an output event) can trigger a successor event transition if $\underline{\sigma'}$ is active at some vertex $q^k_{i_k}$ $(j \neq m)$; that is $(q^k_{i_k}, \underline{\sigma'} \rightarrow \overline{\sigma''}, q^k_{i'_k}, x^0_{q^k_{i'_k}})$ is a transition in E^k. 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). We shall always assume that only a finite chain of instantaneously triggered transitions can occur in succession.

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

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

[5]This follows from the decomposition of guarded event transitions into dynamic and event transitions as described previously.

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

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

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

- The path of the run is the sequence of the vertices.

- The input trace of the run is the sequence of the input-events.

- The output trace of the run is the sequence of the output-events.

To facilitate our ensuing exposition, we will standardize EHMs as follows. Recall that our model allows guarded event transitions of the form

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

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

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

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

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

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

If $\overline{\sigma'}$ is absent, then no output-event is transmitted. If $x^0_{q'}$ is absent (or partially absent), then the initial condition is inherited (or partially inherited) from x_q (assuming x_q and $x_{q'}$ represent the same physical object, and hence are of the same dimension). We often write the transition as $q \stackrel{G \wedge \underline{\sigma}}{\longrightarrow} q'$ or $(q, G \wedge \underline{\sigma}, q')$ if $\overline{\sigma'}$ and $x^0_{q'}$ are either absent or understood.

If $\underline{\sigma}$ is absent, then the transition takes place immediately upon G becoming true. Such a transition is called a *dynamic* transition. 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.* When both G and $\underline{\sigma}$ are present, the transition is called a *guarded event transition.*

- $I = \{I_q : q \in Q\}$ is a set of invariants. For each $q \in Q$, I_q is defined as $I_q = cl(\neg(G_1 \vee ... \vee G_k))$, where $G_1, ..., G_k$ are the guards at q, and where $cl(.)$ denotes set closure[3].

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

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

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

A *run* of the EHM is a sequence

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

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

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

2 Hybrid Machines

In this section we briefly review the hybrid-machine formalism as described e.g. in [12]. An elementary hybrid machine is defined as a tuple

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

whose elements are defined as follows:

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

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

  with x_q, u_q, and y_q, respectively, the state, input, and output variables of appropriate dimensions. f_q is a Lipschitz continuous function and h_q a continuous function. (A vertex need not have dynamics associated with it; that is, we permit $d_q = \emptyset$, in which case we say that the vertex is *static*.) Note that the dynamics, and in particular the dimension of x_q, can change from vertex to vertex.
- $E = \{(q, G \wedge \underline{\sigma} \rightarrow \overline{\sigma'}, q', x^0_{q'}) : q, q' \in Q\}$ is a set of *edges* (or *transition-paths*), where q is the vertex exited, q' is the vertex entered, $\underline{\sigma}$ is the input-event, $\overline{\sigma'}$ the output-event. G is the guard, formally given as a Boolean combination of inequalities of the form $\Sigma_i a_i s_i \geq C_j$ or $\Sigma_i a_i s_i \leq C_j$, where the s_i are shared (signal) variables, to be defined shortly, and the a_i and C_j are real constants. Finally, $x^0_{q'}$ is the initialization value for $x_{q'}$ upon entry to q'.

  $(q, G \wedge \underline{\sigma} \rightarrow \overline{\sigma'}, q', x^0_{q'})$ is interpreted as follows: If G is true and the event $\underline{\sigma}$ is received as an input, then the transition to q' takes place at the instant $\underline{\sigma}$ is received[1], with the assignment of the initial condition $x_{q'}(t_{0,q'}) = x^0_{q'}$ (where $t_{0,q'}$ denotes the time at which the vertex q' is entered and $x^0_{q'}$ is a constant vector[2]). The output-event $\overline{\sigma'}$ is transmitted at the same time.

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

[2] More general assignments of the initial conditions such that $x^0_{q'}$ is a function of x_q can also be introduced without much difficulty.

where system dynamics is *rate-bounded* 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 $S \geq C$).

The control problem that we focus on in the present paper, is the synthesis of a supervisory controller, where the objective is to guarantee that the system satisfies a set of *legal* specifications. Legal specifications are traditionally partitioned into *safety* specifications that state what the system must be prevented from being able to do, and *liveness* specifications that state what the system is required to do. A typical safety specification is to ensure that the system will never enter a specified set of *illegal* configurations. A typical liveness specification is to ensure that every run of the system will reach a set of *marked* configurations, that represent task completion.

The synthesis of legal safety controllers for rate-bounded hybrid machines, was investigated in [10] [11] [12]. Among all legal controllers, we were particularly interested in minimally restrictive (or minimally interventive) ones, that allow the maximal possible set of legal behaviors to survive. Synthesis algorithms for minimally interventive controllers were developed, and the problem of system viability was examined. Synthesis of safety controllers for hybrid systems was also studied in [9] [22].

In the present paper we present an initial investigation of synthesis of liveness controllers for hybrid machines. To this end we define *open* hybrid machines (as opposed to *closed* hybrid machines) as systems that can interact with the environment through event synchronization and can therefore be "driven" to their marked configurations by the user (controller). In view of obvious timing constraints, liveness specifications for hybrid systems must be associated with explicit timing constraints. Thus, we may require that for a specified time limit, every run reach a marked configuration within that time limit. We call such a specification a *fixed-time* liveness specification. Alternatively, a more relaxed specification may be that, for some (unspecified) global time bound, every run of the system reach a marked configuration within that time bound. We call this the *bounded-time* liveness specification. Finally, the least restrictive liveness requirement is that every run reach a marked configuration within a finite time limit (but we do not insist on the existence of a global time bound for all runs). We call this the *finite-time* liveness specification.

It is not hard to see that a fixed time liveness specification can be readily translated into a safety requirement, by conjoining a global clock to the system, and calling "unsafe" each configuration whose clock value exceeds the (specified) time bound. Therefore, the fixed time liveness case can be dealt with algorithmically, just as a control problem with safety specifications. In contrast, the bounded time liveness controller must be handled differently, and this is the focus of the present paper, where we present a synthesis algorithm for a minimally interventive controller with bounded-time liveness specifications.

Abstract

Liveness in hybrid systems is defined as the ability of the system to complete a specified task under all operating conditions and for all possible runs. Liveness is classified in the present paper into *fixed-time*, *bounded-time*, and *finite-time* liveness. We present an algorithm for synthesis of minimally-interventive controllers that achieve liveness in rate-bounded hybrid systems.

Keywords: Hybrid systems, liveness, control synthesis

1 Introduction

Hybrid systems are dynamical systems in which discrete and continuous behaviors coexist and interact [1] [3] [6]. Such systems frequently arise, for example, from computer aided control of continuous (and discrete) processes in manufacturing, communication networks, flight control systems, traffic control, industrial process control, and the like. Various formalisms have been proposed in the literature for the mathematical description of these systems [6] [3] [1] [12]. Among these, the formalism of hybrid-automata [1], which augments the state-machine framework with dynamics to capture timing constraints and continuous dynamics, gained fairly wide acceptance. A formalism related to hybrid automata for modeling hybrid systems, called *hybrid machines*, that differs from the latter in some substantial detail [12], was developed in [10] [11] [12] to capture *open* hybrid systems that interact with their 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 be achieved by employing both interaction mechanisms, to modify and restrict system's behavior. This flexibility adds significantly to the potential control capabilities of hybrid system (as compared to either discrete-event or continuous systems), but clearly makes the problem of controller 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. Thus, most attention to date, in control of hybrid systems, has focused either on continuous aspects (i.e., signal-sharing) or discrete aspects (i.e., event-synchronization) but not both.

In the present paper we examine the control problem for a class of *composite hybrid machines* (CHMs) that consist of the concurrent operation (employing synchronous composition) of *elementary hybrid machines* (EHMs), that allows both signal sharing and event synchronization. A controller can then be coupled with the plant by means of synchronous composition. We confine our attention to controllers that interact with the system only through event synchronization. We further restrict ourselves to a special class of CHMs,

Control of Rate-Bounded Hybrid Systems with Liveness Constraints*

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

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

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

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

Dedicated to Didi Hinrichsen on the Occasion of his 60th Birthday

*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.