Computing Approximating Automata for a Class of Linear Hybrid Systems

Alongkrit Chutinan
ac4c@andrew.cmu.edu

Bruce H. Krogh
krogh@ece.cmu.edu

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890

to appear in *Springer Lecture Notes in Computer Science: Hybrid Systems V*

Abstract

Approximating automata are finite-state representations of the sequential input-output behaviors of hybrid systems characterized by threshold events that trigger discrete changes in the continuous dynamic equations. Procedures proposed for constructing approximating automata require forward and backward mappings of sets of continuous state trajectories – mappings which are not available for arbitrary continuous dynamics. This paper develops the foundations for constructing approximating automata automatically for hybrid systems in which the continuous dynamics are defined by convex polytopes in the vector space of the derivatives of the continuous state trajectories. The computations are illustrated for a simple example which also demonstrates the use of approximating automata to solve verification problems that may be intractable using fixed-point computations for linear hybrid automata.

1 Introduction

This paper concerns the generation of purely discrete models (finite automata) for a class of hybrid systems in which the continuous-state trajectories generate events when specified threshold hypersurfaces are reached. The continuous dynamics are selected by a discrete input signal which switches only when threshold events occur. An algorithm was proposed and illustrated in [3] for computing *approximating automata* for such hybrid systems with arbitrary continuous dynamics. The languages of the approximating automata contain all possible sequences of pairs of inputs and threshold events that are valid for the hybrid system. The approximation can be refined iteratively, converging when possible to a finite automaton that generates the exact sequential behaviors of the hybrid system [9]. In [2] we proposed an alternative algorithm for constructing approximating automata for hybrid systems which can be refined more efficiently than the approximating automata in [3].

The purpose of this paper is to identify a set of hybrid systems for which the approximating automata can be constructed automatically. The algorithms for computing approximating automata rely on mappings that characterize the forward and backward evolution of families of continuous-state trajectories. It is also necessary to compute the set of continuous states from which no events will be generated, called the set of *null event states*. For arbitrary continuous dynamics, there are no computationally effective ways to compute the forward and backward mappings and the set of null event states. In this paper we consider hybrid systems with continuous dynamics in which the derivative of the continuous state vector lies in a convex polytope, similar to the continuous dynamics in so-called linear hybrid automata [1]. For this case, it is possible to carry out all of the computations necessary to generate the approximating automata using standard computations on linear inequalities and convex polytopes. This paper demonstrates how this can be done in the context of the algorithm in [2]. The same computations could be used in the algorithm in [3].

Although many (but not all) of the computations discussed in this paper are similar to computations used in programs such as HyTech [1] for the verification of linear hybrid automata, approximating automata provide a different approach for hybrid systems analysis and synthesis. To perform verification when the logic for selecting the discrete input based on the threshold events is given, an approximating automaton can be generated first. Properties of the hybrid system can then be verified using tools for finite state systems, such as SMV [7] or COSPAN [6]. One approximating automaton may be used to verify several properties of the hybrid system using these faster, more efficient tools. As shown by the example in section 6, approximating automata may make it possible to verify properties for which the fixed-point computations for the linear hybrid automata will not terminate.

Approximating automata can also be used for the synthesis of discrete controllers. For this application, illustrated in [3], the approximating automata represent the sequential behaviors of the hybrid system under all possible choices of the discrete input at each threshold event. This model can be used as the *plant* to synthesize control logic based on the standard discrete event systems theory [11]. Alternatively, an approximating automaton could replace hybrid system dynamics in parts of a large simulation model to enable much faster evaluation of alternative designs. For both controller synthesis and simulation, the approximating automata provide a finite-state models of all possible sequential input-output behaviors of the hybrid dynamics which can be used as subsystems in larger systems of interconnected input-output models, facilitating modular analysis and design based on discrete techniques.

The concept of approximating automata has also been developed by Raisch and O'Young for discrete-time hybrid systems [10]. Their algorithm for constructing approximating automata is similar to the algorithm in [3], but the forward and backward mappings are given immediately by the discrete-time continuous state transition equations. For continuous-time hybrid systems, the approach of [10] would apply to sampled-data implementations where the sampling times are fixed and known a priori. In the continuous-time model considered in this paper, the "sampling times" are generated by the threshold events, which means the forward and backward mappings must be computed by integrating the state equations. It is precisely this problem, computing the continuous state trajectories, that is addressed in this paper for a particular class of hybrid systems.

The remainder of the paper is organized as follows. The following section describes the class of hybrid systems to be considered. Section 3 defines the mappings of continuous-state trajectories to be used in the construction algorithm, which is presented in section 4. Section 5 presents the details for calculating the mappings and null event sets for the class of linear hybrid systems defined in section 2. A simple example is presented in section 6 to illustrate the computation of approximating automata for verification. The concluding section summarizes the results in this paper and discusses current research directions.

2 A Class of Linear Hybrid Systems

In this paper we consider the class of hybrid systems illustrated by the block diagram in Fig. 1. The signal $u(\cdot)$ is a right-continuous piecewise constant *condition signal* taking on discrete values in the finite set U. This signal is generated as the state of a condition/event (C/E) system whose state transitions are forced by the *event signal* $v(\cdot)$, a pointwise-nonzero n_v-dimensional signal taking on values in $\{0,1\}^{n_v}$. The initial discrete state of the C/E system is nondeterministic. The set of possible initial discrete states is given as $U_0 \subseteq U$. The set of events that drive transitions in the C/E system are the nonzero values of the event signal $v(\cdot)$, that is, values in the *threshold event set* $V = \{0,1\}^{n_v} - \{0\}^{n_v}$.

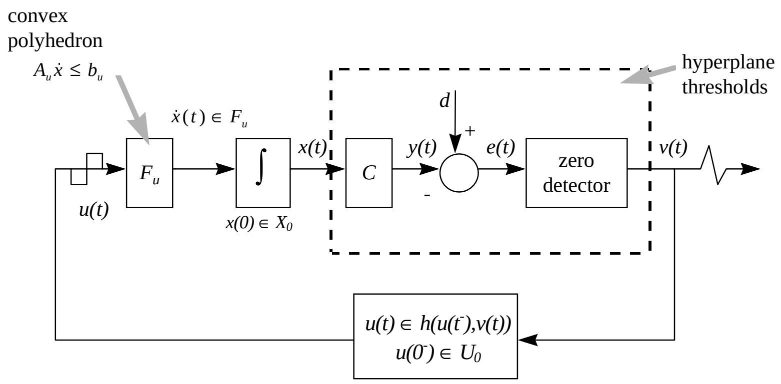


Figure 1: Block diagram illustrating the class of hybrid systems being considered

The block in Fig. 1 receiving the input signal $u(\cdot)$ is a *signal generator* whose output signal $\dot{x}(\cdot)$ is a piecewise continuous signal with values confined at each instant to be in the set $F_u \subseteq R^{n_x}$, where u is the current value of the input signal $u(\cdot)$ and n_x is the dimension of the continuous state space.

We assume that for each $u \in U$, F_u is a convex polytope defined by the matrix-vector inequality

$$F_u = \{s \in R^{n_x} | A_u s \le b_u\} \tag{1}$$

where $A_u \in R^{m_u \times n_x}$, $b_u \in R^{m_u}$ and integer m_u are given for each u. We refer to a set of the form (1) as a *derivative constraint set.*

The integrator block in Fig. 1 operates on each component of its input signal to generate the *continuous state trajectory* $x(\cdot)$. X_0 is the set of possible initial continuous states. The continuous state vector is multiplied in the gain block by the *output matrix* $C \in R^{n_v \times n_x}$ creating the continuous output trajectory $y(\cdot)$.

The *threshold events* which constitute the non-zero values of $v(\cdot)$ are generated as follows. Each component of the continuous output vector is subtracted from a component of a constant *threshold vector* $d \in R^{n_v}$, and the resulting signal $e(\cdot) = d - y(\cdot)$ is the input to a *zero detector* which generates the components of the event signal $v(\cdot)$ according to

$$v_i(t) = \begin{cases} 1 & \text{if } e_i(t) = 0 \text{ and } \exists \triangle > 0 \text{ s.t. } e_i(t-\delta) > 0 \text{ for } 0 < \delta < \triangle \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Note that the zero detector is directional: an event is generated only when a component of $y(t)$ approaches its threshold from below. Note also that a distinct event is generated when more than one threshold is hit at a particular instant. This reflects the fact that when multiple thresholds are encountered simultaneously, the location of the continuous state is known more precisely than when any of the thresholds are encountered individually.

In general, the analysis of the behavior of a hybrid system as described above is virtually intractable if the input signal $u(\cdot)$ can switch at any instant. The scenario of interest is when the input condition signal $u(\cdot)$ changes values only in response to threshold events observed in $v(\cdot)$. This constraint on $u(\cdot)$ is enforced by the assumption that for all u

$$h(u, 0) = \{u\},$$

that is, the C/E system in Fig. 1 allows no transitions when the input event signal is zero. Thus, a signal pair $(u(\cdot), v(\cdot))$ is admissible for the hybrid system only if discontinuities in $u(\cdot)$ occur only when $v(\cdot)$ is non-zero.

To assure $u(\cdot)$ is piecewise constant with only a finite number of switches on any finite interval of time, we assume that $v(\cdot)$ is *nonexplosive* (the term used in the Markov process literature [4]) or *nonzeno* (the term popular with computer scientists). That is, we assume that on any finite interval of time $v(\cdot)$ is nonzero a only a finite number of, or zero, times.

Following the development in [8], we define the discrete sequential behavior of the hybrid system described above as the language of the admissible sequences of pairs of discrete inputs and threshold events. In general this language, denoted L_H, contains strings of finite length and sequences of infinite length. Specifically, the sequential behavior corresponding to a signal pair $(u(\cdot), v(\cdot))$ is the sequence of ordered pairs of the form $(u(t_i^-), v(t_i))$, where the times t_i are the instants at which $v(t) \neq 0$, and $u(t_i^-)$ is the value of the discrete input that lead to the generation of the event $v(t_i)$. If this sequence is finite length, it is terminated by the pair $(u(t_f), v_\perp)$, where t_f is the time of the last threshold event in the signal $v(\cdot)$ and $v_\perp$ is a special symbol introduced to denote this *null threshold event.*.

Given a hybrid system, our interest is in creating a finite-state automaton that generates the language L_H. Since such an automaton may not exist, or may be too complex for the purpose at hand, our objective is to generate an approximating automaton A whose language L_A contains L_H. Such an automaton is useful either for verification of system properties or for discrete controller synthesis as described in the introduction. For controller synthesis, the nondeterministic C/E system in Fig. 1 represents the choices avaliable to the supervisor, where some restrictions may be imposed by logic already incorporated into the hybrid system either physically or by lower-level discrete controllers. A supervisor designed to achieve specifications for the sequential behavior for L_A can then be converted to a supervisor which achieves the same specifications for the hybrid system as described in [3]. The remainder of this paper deals with the construction of approximating automata for hybrid systems of the form illustrated in Fig. 1. Throughout the remaining sections the term *hybrid system* refers to this particular type of hybrid system.

3 Mappings of Families of Continuous Trajectories

The sequential behaviors of the hybrid systems are generated by continuous trajectories that evolve between threshold events. Therefore, computing the mappings of families of trajectories from one threshold hyperplane to another is the key issue in the construction of approximating automata. This section introduces notation and definitions to deal with these mappings formally.

Threshold events in a hybrid systems are generated by a set of n_v *threshold hyperplanes* denoted $M_1, \ldots, M_{n_v}$. Threshold hyperplane M_i is defined by row i of the output matrix C, and component i of the threshold vector d; that is,

$$M_i = \{x | c_i^T x = d_i\}$$

Without lost of generality we assume $\|c_i\| = 1$. We denote the *negative, positive*, and *non-negative half-spaces* determined by c_i and d_i by $M_i^-(c_i^T x < d_i), M_i^+(c_i^T x > d_i), M_i^{0+}(c_i^T x \geq d_i)$, respectively.

Given a continuous state $x_0 \in R^{n_x}$ and a set $F \subseteq R^{n_x}$, we define the set of admissible continuous-state trajectories starting from x_0 with derivatives lying in F as

$$\begin{aligned}\mathcal{X}(x_0, F) = \{ \quad & x(\cdot) \mid x(0) = x_0, \dot{x}(t) \in F \text{ for all } t \geq 0, \\ & \text{and } \dot{x}(\cdot) \text{ is piecewise continuous } \}.\end{aligned}$$

We let $ZD(e(\cdot))$ denote the event signal obtained when the (directional) zero detection operation is applied to the signal $e(\cdot)$.

The *forward dynamic mapping* is defined in terms of an initial continuous state x_0, a discrete state u, a set $T \subset R^{n_x}$, and an event vector $\hat{v} \in V$ as

$$\begin{aligned}\mathcal{M}_u(x_0 \to T \uparrow \hat{v}) = \{ \quad & x_f \in T \mid \exists x(\cdot) \in \mathcal{X}(x_0, F_u), \text{ and } t_f > 0 \text{ s.t.} \\ & x_f = x(t_f), \text{ and for } v(\cdot) = ZD(d - Cx(\cdot)), \\ & v(t_f) = \hat{v} \text{ and } v(t) = 0, \ \forall 0 \leq t < t_f \ \}.\end{aligned}$$

In words, $\mathcal{M}_u(x_0 \to T \uparrow \hat{v})$, is the set of states in T that are reached by trajectories starting from x_0, and when reached, the event $\hat{v}$ is generated. Note that the particular event generated when a trajectory reaches T depends on the direction (derivative) of the trajectory at that instant since the zero detector is directional.

We generalize the above definition for a set of initial states $S \in R^{n_x}$ as

$$\mathcal{M}_u(S \to T \uparrow \hat{v}) = \bigcup_{x_0 \in S} \mathcal{M}_u(x_0 \to T \uparrow \hat{v}).$$

The *backward dynamic mapping* from T to S under u triggering $\hat{v}$ is defined by

$$\mathcal{B}_u(T \uparrow \hat{v} \to S) = \{x_0 \in S \mid \mathcal{M}_u(x_0 \to T \uparrow \hat{v}) \neq \emptyset\}$$

In words, $\mathcal{B}_u(T \uparrow \hat{v} \to S)$ identifies the part of S from which T can be reached under u, triggering $\hat{v}$ as the first event along the trajectory.

Finally, we define the *null event states* as the set of states from which there exists a trajectory which never triggers any event for a given discrete state u. Formally this set is given as

$$\begin{aligned}\mathcal{N}_u = \{ \quad & x_0 \in R^{n_x} \mid \exists x(\cdot) \in \mathcal{X}(x_0, F_u) \text{ s.t.} \\ & v(\cdot) = ZD(d - Cx(\cdot)) \text{ and } v(t) = 0, \forall t \geq 0\}.\end{aligned}$$

States in $\mathcal{N}_u$ can lead to terminating behaviors, that is, to trajectories where no further threshold events are generated by the hybrid system.

4 Approximating Automata

In this section we summarize the method from [2] for constructing an automaton whose behaviors include all possible behaviors of the hybrid system. Let us first discuss the basic ideas of our approach. We start with a finite partition $\mathcal{P}$ of the threshold hyperplanes where each element of $\mathcal{P}$ is connected and completely contained in some subset of the threshold hyperplanes. Given such a partition, we define an approximating automaton A as follows. Except for states labeled q_0 and $q_\perp$, each state of A is labeled by a triple $(P, u, v) \in \mathcal{P} \times U \times V$. There are two types of transitions in the automaton, the *regular event* transitions, each labeled by $(u, v) \in U \times V$, and the *null event* transitions, each labeled by $(u, v_\perp) \in U \times \{v_\perp\}$. A regular event transition (u', v') from (P, u, v) to (P', u', v') is interpreted as follows. Given that a continuous state in P has been reached under the discrete state u and the event v is triggered at P, there is possibly a continuous trajectory from a state in P to another state in P' under u' that triggers the event v'. The discrete state switching from u to u' must be consistent with the discrete transition function h, that is, $u' \in h(u, v)$. A null event transition $(u', v_\perp)$ from (P, u, v) to $q_\perp$ indicates that there is possibly a continuous trajectory under u' from a state in P that never triggers any event. The *null state* $q_\perp$ is a special state that we use to represent the case where no more threshold events occur. q_0 is the initial state.

To ensure that any subset S of a piece of an event hyperplane P can generate any event that is possible for P, the partition $\mathcal{P}$ needs to be *unambiguous*, which is defined as follows.

Definition 1 Let $M = \bigcup_{i=1,\ldots,n_v} M_i$. A partition $\mathcal{P}$ of M is *unambiguous* partition of the threshold hyperplanes if for all $P \in \mathcal{P}$, P is connected and

$$\text{either } P \subseteq M_k \text{ or } P \cap M_k = \emptyset \;\forall\; 1 \leq k \leq n_v. \tag{3}$$

In the following we will also say any set $T \subset M$ is unambiguous with respect to the set of hyperplanes $M_1, \ldots, M_{n_v}$ if T satisfies the condition (3) with T replacing P.

Definition 2 Given a hybrid system and an unambiguous partition $\mathcal{P}$ of the collection of threshold hyperplanes, we define the *approximating automaton* A corresponding to $\mathcal{P}$ as

$$A = (Q, \Sigma, \delta, q_0)$$

where

Q The set of the states $Q = \mathcal{P} \times U \times V \cup \{q_0, q_\perp\}$

Σ The alphabet is $U \times (V \cup \{v_\perp\})$.

δ The nondeterministic transition function δ for A is defined according to the following rules

1. $(P, u, v) \in \delta(q_0, (u, v))$ if $\mathcal{M}_u(X_0 \to P \uparrow v) \neq \emptyset$ and $u \in U_0$
2. $q_\perp \in \delta(q_0, (u, v_\perp))$ if $\mathcal{N}_u \cap X_0 \neq \emptyset$ and $u \in U_0$
3. $(P', u', v') \in \delta((P, u, v), (u', v'))$ if $\mathcal{M}_{u'}(P \to P' \uparrow v') \neq \emptyset$ and $u' \in h(u, v)$

4. $q_\perp \in \delta((P,u,v),(u',v_\perp))$ if $\mathcal{N}_{u'} \cap P \neq \emptyset$ and $u' \in h(u,v)$

q_0 is the initial state of A

Letting L_A denote the collection of all possible finite-length and infinite-length sequences of input-event pairs that can be generated by the automaton A, it can be shown that $L_H \subseteq L_A$ [2]. (In our context, a sequence is *generated* by an automaton A if and only if it is a sequence of labels on either an infinite-length directed path in the automaton starting from q_0, or the string of event labels on a finite-length path from q_0 to $q_\perp$.) The extent to which A approximates H depends on the granularity of the partition $\mathcal{P}$: if partition $\mathcal{P}'$ refines partition $\mathcal{P}$ with corresponding automata A' and A, then $L_H \subseteq L_{A'} \subseteq L_A$.

We now describe the procedure to obtain a finer partition $\mathcal{P}_{N+1}$ from a given partition $\mathcal{P}_N$. This procedure is shown in Fig. 2. Let A_N be the automaton defined from the current partition $\mathcal{P}_N$. For every hyperplane piece $P \in \mathcal{P}_N$, we identify all the distinct evolutions of the continuous trajectories from P under different u's. For each transition $(P,u,v) \stackrel{(u',v')}{\rightarrow} (P',u',v')$, we would like to isolate the subset of P corresponding to this transition. We use the backward mapping for this purpose. We can see that the subset we are looking for is exactly the backward mapping $\mathcal{B}_{u'}(P' \uparrow v' \rightarrow P)$. Similarly, for each transition $(P,u,v) \stackrel{(u',v_\perp)}{\rightarrow} q_\perp$, the subset of P corresponding to this transition is simply $\mathcal{N}_{u'} \cap P$.

By computing the backward mapping and the null event set from all successors of states containing P, we have a collection of subsets of P corresponding to different transitions in the automaton A_N. We then make a subpartition of P from these subsets. This process is represented by the routine `disjoint` in Fig. 2. Given a collection of sets, `disjoint` returns a collection of disjoint sets whose union is the same as the union of the input sets. After we have proceeded through all $P \in \mathcal{P}_N$, all subpartitions of the sets P form the new partition $\mathcal{P}_{N+1}$ which is a refinement of $\mathcal{P}_N$. With the new partition, we can define a new automaton A_{N+1}. Therefore, $L_{A_{N+1}} \subseteq L_{A_N}$, i.e., each refinement step gives an approximating automaton that is at least as good an approximation to the hybrid system as the one obtained in the previous step. Furthermore, it can be shown that if $L_{A_{N+1}} = L_{A_N}$, then $L_{A_{N+1}} = L_H$, i.e. if the refinement ever reaches a fixed point then the approximation, in terms of the sequences of input-event pairs, is exact [2].

5 Calculating Mappings and the Null Event Set

The computations required to generate the approximating automata are similar to the computations employed in the tool HyTech [1]. In particular, the fundamental operation is $Reach_F(X)$, which gives the set of continuous states reachable from the set $X \subseteq R^{n_x}$ when the differential inclusion for the continuous-state trajectories is defined by the set $F \subseteq R^{n_x}$. Formally, $Reach_F(X)$ is defined as

$$Reach_F(X) = \bigcup_{x_0 \in X} \{x_f \in R^{n_x} \mid \exists x(\cdot) \in \mathcal{X}(x_0,F) \text{ and } t_f \geq 0 \text{ s.t. } x(t_f) = x_f\}.$$

For a given set $F \subseteq R^{n_x}$, defining

$$-F = \{-s \mid s \in F\},$$

```
𝒫_{N+1} = refine(𝒫_N = {P_1, ..., P_s} | 𝒫_N is a partition of M)
{
  s = |𝒫_N|
  𝒫' = ∅
  for i = 1 to s {
    k = 0
    /* check regular event transitions */
    for each transition (P_i, u, v) --(u',v')--> (P_j, u', v') {
      S = ℬ_{u'}(P_j ↑ v' → P_i)
      /* add subset of P_i that maps under (u', v') to P_j) */
      if (S ≠ ∅) {
        k = k + 1
        P'_{ik} = S
      }
    }
    /* check null event transitions */
    for each transition (P_i, u, v) --(u',v_⊥)--> q_⊥ {
      S = 𝒩_u ∩ P_i
      /* add subset of P_i that maps under u' to q_⊥ */
      if (S ≠ ∅) {
        k = k + 1
        P'_{ik} = S
      }
    }
    /* call disjoint to compute subpartition of P_i
       and form 𝒫_{N+1} from subpartitions of P_i's */
    𝒫' = 𝒫' ∪ disjoint({P'_{i1}, ..., P'_{ik}})
  }
  return 𝒫'
}
```

Figure 2: Pseudocode for the partition refinement procedure.

we have that $Reach_{-F}(X)$ corresponds to the *time precondition* operator defined in [1]. $Reach_F(X)$ corresponds to the *time postcondition* operator, which can be defined in a similar manner. Note that $Reach_{-F}(X)$ is the set of states from which the set X can be reached when the differential inclusion is defined by F since $-F$ essentially runs the system backwards in time.

The following definitions will also be useful later on in this section. Given an initial state x_0 and a set $F \subseteq R^{n_x}$, the set of states reachable at time t from x_0 under differential inclusion F is defined as

$$Reach_F(t, x_0) = \{x_f \mid \exists x(\cdot) \in \mathcal{X}(x_0, F) \text{ such that } x(t) = x_f\}.$$

For $v \in V$ and T unambiguous with respect to the set of threshold hyperplanes, define the set

$$R_{T\uparrow v} = \bigcap_{i \text{ s.t. } T\subset M_i, v_i=1} M_i^- \bigcap_{i \text{ s.t. } T\subset M_i, v_i=0} M_i^{0+}. \tag{4}$$

In words, $R_{T\uparrow v}$ is the set of states from which a state trajectory must arrive at T to generate the event v.

Throughout this section, we assume that a target set T is unambiguous with respect to the set of threshold hyperplanes. Furthermore, whenever the notation $T \uparrow v$ arises, it is assumed that $v_i = 1 \Rightarrow T \subset M_i$ for all $i = 1, \ldots, n_v$. In other words, it must be possible to trigger v at T.

5.1 Forward Mapping

In this subsection, we present the method for computing the forward mapping set under a discrete state u given a set of states S called a *source*, a set of *target* states T, and the target event $\hat{v}$ to be triggered when T is encountered. The forward mapping $\mathcal{M}_u(S \to T \uparrow \hat{v})$ can be computed as follows.

1. Rule out the part of S for which it is impossible to reach T and trigger $\hat{v}$ before any other event. The remaining part is the candidate initial set.

$$CAN(S \to T \uparrow \hat{v}) = S \bigcap_{i \text{ s.t. } T\subset M_i^+} M_i^{0+} \bigcap_{i \text{ s.t. } T\subset M_i, \hat{v}_i=0} M_i^{0+}$$

2. Compute the set of via points in $R_{T\uparrow v}$ to which any trajectory from S must pass through in order to reach T and trigger the event $\hat{v}$ before any other events.

$$FVIA_u(S \to T \uparrow \hat{v}) = Reach_{F_u}(CAN(S \to T \uparrow \hat{v})) \cap R_{T\uparrow v} \bigcap_{i \text{ s.t. } T\subset M_i^+} M_i^{0+}$$

3. Compute the subset of T that is reachable from $FVIA_u(S \to T \uparrow \hat{v})$

$$FMAP_u(S \to T \uparrow \hat{v}) = Reach_{F_u}(FVIA_u(S \to T \uparrow \hat{v})) \cap T$$

The following lemmas and propositions demonstrate that

$$FMAP_u(S \to T \uparrow \hat{v}) = \mathcal{M}_u(S \to T \uparrow \hat{v}).$$

All proofs are given in the Appendix.

Lemma 1 Given two states x_0 and x_f, if there exists an admissible trajectory $x_1(\cdot) \in \mathcal{X}(x_0, F_u)$ reaching x_f at time t_f, there exists another admissible trajectory $x_2(\cdot) \in \mathcal{X}(x_0, F_u)$ which is a straight line connecting x_0 and x_f.

Lemma 2 If a trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$ from $x_0 \in S$ to T triggers $\hat{v}$ before any other event, then $x_0 \in CAN(S \to T \uparrow \hat{v})$.

Proposition 1 Any trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$ from a state $x_0 \in S$ that reaches T and triggers $\hat{v}$ before any other event must pass through $FVIA_u(S \to T \uparrow \hat{v})$.

Proposition 2 If $x_f \in Reach_{F_u}(FVIA_u(S \to T \uparrow \hat{v})) \cap T$, then for any $x_m \in FVIA_u(S \to T \uparrow \hat{v})$ such that $x_f \in Reach_{F_u}(\{x_m\})$, there exists a trajectory $x(\cdot) \in \mathcal{X}(x_m, F_u)$ that reaches x_f, triggering $\hat{v}$ before any other event.

Proposition 3 $x_f \in \mathcal{M}_u(S \to T \uparrow \hat{v}) \iff x_f \in FMAP_u(S \to T \uparrow \hat{v})$.

5.2 Backward Mapping

The backward mapping under a discrete state u given a source S, target T and, a target event $\hat{v}$ can be computed in a manner similar to the forward mapping.

Given event $\hat{v} \in V$. T unambiguous with respect to the threshold hyperplanes and a discrete state u, we define the backward mapping set $\mathcal{B}_u(T \uparrow \hat{v})$ as

$$\begin{aligned}\mathcal{B}_u(T \uparrow \hat{v}) = \{ \quad & x_0 \mid \exists\, x(\cdot) \in \mathcal{X}(x_0, F_u), t_f > 0 \text{ s.t. } x(t_f) \in T, \text{ and for} \\ & v(\cdot) = ZD(d - Cx(\cdot)), v(t_f) = \hat{v}, \text{ and } v(t) = 0, \forall 0 \le t < t_f\}\end{aligned}$$

The following steps can be used to compute $\mathcal{B}_u(S \to T \uparrow \hat{v})$.

1. Compute the set of via points in $R_{T\uparrow\hat{v}}$ from which T can be the first subset of threshold hyperplanes that is reached and thereby triggering $\hat{v}$.

$$BVIA_u(T \uparrow \hat{v}) = Reach_{-F_u}(T) \cap R_{T\uparrow\hat{v}} \bigcap_{i \text{ s.t. } T \subset M_i^+} M_i^{0+}$$

2. Compute the set of states from which $BVIA_u(T \uparrow \hat{v})$ can be reached

$$Reach_{-F_u}(BVIA_u(T \uparrow \hat{v}))$$

The above computation gives the set of candidate states from which T can be reached and $\hat{v}$ can be triggered.

3. Restrict the above set by some half spaces to yield the final result

$$BMAP_u(T \uparrow \hat{v}) = Reach_{-F_u}(BVIA_u(T \uparrow \hat{v})) \bigcap_{i, T \subset M_i, \hat{v}_i = 0} M_i^{0+} \bigcap_{i, T \subset M_i^+} M_i^{0+}$$

The propositions that follow lead to our main result in this subsection that $\mathcal{B}_u(T \uparrow \hat{v}) = BMAP_u(T \uparrow \hat{v})$, which is proved in Proposition 6. The backward mapping from $T \uparrow \hat{v}$ back to S under u is simply given by

$$\mathcal{B}_u(T \uparrow \hat{v} \to S) = \mathcal{B}_u(T \uparrow \hat{v}) \cap S$$

Proposition 4 Any trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$ from a state $x_0 \in S$ that reaches T and triggers $\hat{v}$ before any other event must pass through $BVIA_u(T \uparrow \hat{v})$.

Proposition 5 If $x_m \in BVIA_u(T \uparrow \hat{v})$, then there exists a trajectory $x(\cdot) \in \mathcal{X}(x_m, F_u)$ that reaches some $x_f \in T$, triggering $\hat{v}$ before any other event.

Proposition 6 $x_0 \in \mathcal{B}_u(T \uparrow \hat{v}) \iff x_0 \in BMAP_u(T \uparrow \hat{v})$

5.3 Null Event Set

The following proposition gives the fundamental representation of the the set of reachable states for our hybrid systems and for linear hybrid automata. This result provides the basic method for computing the set of null event states.

Proposition 7 Given a continuous state $x_0 \in R^{n_x}$ and a discrete state $u \in U$,

$$Reach_{F_u}(t, x_0) = x_0 + F_u t$$

where

$$x_0 + F_u t = \{x \in R^{n_x} | \exists w \in F_u \text{ s.t. } x = x_0 + wt\}$$

To compute $\mathcal{N}_u$ we define for a state x_0 the set of indices for the threshold hyperplanes for which x_0 lies in the negative halfspace. This set of indices, and the intersection of the associated halfspaces is given as follows.

Definition 3 Given a state x_0 we define I_{x_0} to be the set of indices i of the hyperplanes such that $x_0 \in M_i^-$. We also define S_{x_0} to be the intersection of all such M_i^-. Formally we define,

$$I_{x_0} = \{i | x_0 \in M_i^-\}$$

$$S_{x_0} = \bigcap_{i \in I_{x_0}} M_i^-$$

The significance of I_{x_0} is that threshold events will be *necessarily* generated from a trajectory starting from x_0 only for the threshold hyperplanes with indices in I_{x_0}. The reason is that the directional nature of the zero detector implies that if $i \notin I_{x_0}$, any trajectory crossing M_i from M_i^- must necessarily first cross M_i from M_i^{0+}. However, if such a trajectory exists, there is another trajectory that will cross into M_i and just keep going. That is, it will not return and cross the threshold hyperplane M_i. These observations lead to the following proposition which completely characterizes $\mathcal{N}_u$.

Proposition 8 $x_0 \in \mathcal{N}_u \iff I_{x_0} = \emptyset$ or $\exists w \in F_u$ s.t. $\forall i \in I_{x_0}$, $c_i^T w \leq 0$

To calculate $\mathcal{N}_u$, we separate R^{n_x} into 2^{n_v} regions as follows. Let $I \in \{0,1\}^{n_v}$, and define the set R_I to be

$$R_I = \bigcap_{j:I_j=0} M_j^- \bigcap_{j:I_j=1} M_j^{0+}.$$

For each $I \in \{0,1\}^{n_v}$, we note that for all $x_0 \in R_I$ the set of indices I_{x_0} is the same. Specifically, $I_{x_0} = \{j | I_j = 0\}$. To decide if $x_0 \in \mathcal{N}_u$ using Proposition 8, we check whether there is a feasible vector w satisfying the following set of constraints

$$\begin{bmatrix} C_{I_{x_0}} \\ A_u \end{bmatrix} w \leq \begin{bmatrix} 0 \\ b_u \end{bmatrix}, \qquad (5)$$

where for $I_{x_0} = \{i_1, \ldots, i_k\}$, $C_{I_{x_0}}$ is defined as

$$C_{I_{x_0}} = \begin{bmatrix} c_{i_1}^T \\ \vdots \\ c_{i_k}^T \end{bmatrix}$$

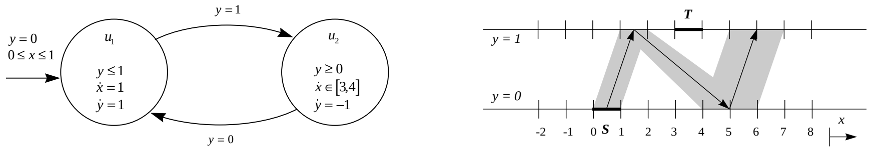
If (5) is feasible, then $x_0 \in \mathcal{N}_u$. Since the above check yields the same result for all $x_0 \in R_I$, we have that $R_I \subset \mathcal{N}_u$. After checking all possible I, we have that

$$N_u = \bigcup_{I \text{ s.t. } (5) \text{ is feasible}} R_I$$

Note that there is no need to check for the case $I = 1_{n_v \times 1}$. In this case $I_{x_0} = \emptyset$ and (5) is always feasible.

6 Example

In this section, we present a simple example to illustrate the construction of approximating automata for a verification problem, and to demonstrate the difference between the approximating automaton approach and the fixed-point reachability computation approach for linear hybrid automata. Consider the *linear hybrid automaton* shown in Figure 3a. We supply this automaton as an input to HyTech and compute the reachability region (for the continuous variables x and y) using the `post` operator, the result for the first three steps are depicted in Figure 3b.



(a) example linear hybrid automaton (b) reachable region

Figure 3:

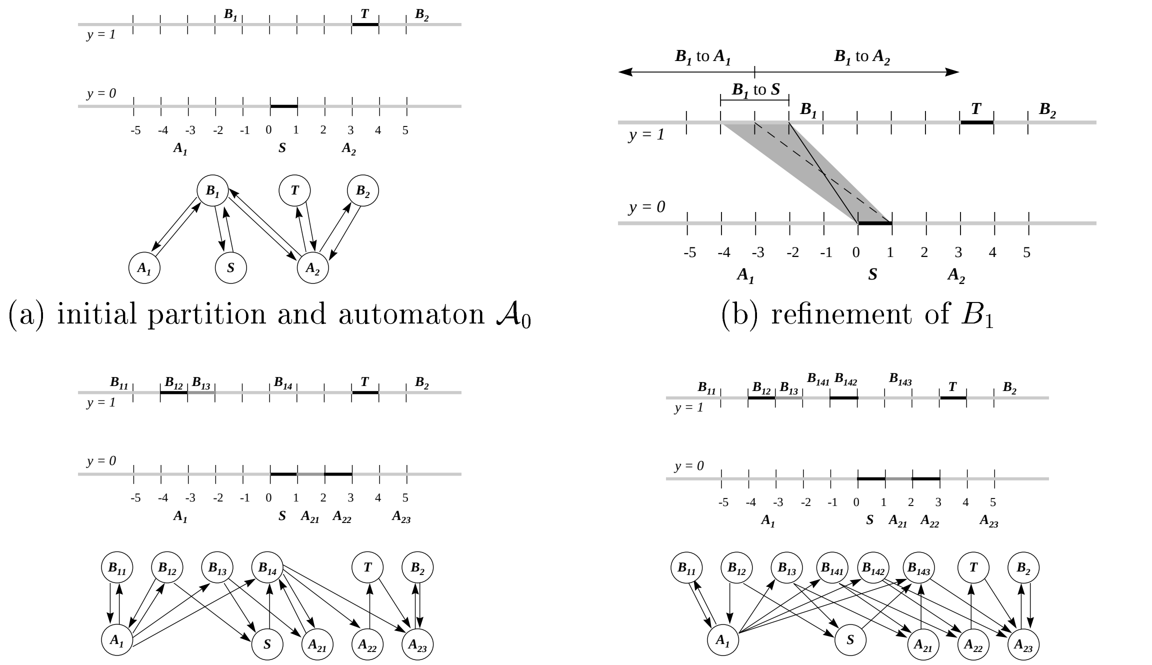
In this example, we would like to verify that starting from the set $S = \{0 \leq x \leq 1, y = 0\}$ in location u_1, the set $T = \{3 \leq x \leq 4, y = 1\}$ is never reached. It is clear from Figure 3 that T cannot be reached from S. However, by iteratively applying the `post` operator in HyTech, the computation never converges as the reachability region continues to expand in the positive x direction. We now turn to the approximating automata approach.

The above linear hybrid automaton can be easily converted into our model of hybrid systems introduced in Section 2 by defining the state variables $x_1 = x$ and $x_2 = y$ with the output matrix $C = [0\ -1;\ 0\ 1]$ and the threshold vector $d = [0\ 1]^T$. The threshold hyperplanes are $-y = 0$ and $y = 1$. The minus sign indicates that the hyperplane $-y = 0$ needs to be hit from above in order for an event to be generated. When the hyperplane $-y = 0$ and $y = 1$ are reached, the events $[1\ 0]^T$ and $[0\ 1]^T$ are generated, respectively. The transition function $h(\cdot, \cdot)$ simply switches the differential inclusion to $F_1 = \{\dot{x} = \dot{y} = 1\}$ on the event $[1\ 0]^T$ and to $F_2 = \{\dot{x} \in [3, 4], \dot{y} = -1\}$ on the event $[0\ 1]^T$.

To define the first approximating automaton, we select the partition of the threshold hyperplanes shown in Figure 4a. We choose to include S and T in the partition because we would like to analyze their transitional behaviors. The automaton $\mathcal{A}_0$ defined from this partition is also shown in Figure 4a. We drop the discrete state and threshold event from the automaton state labels as it is clear that any subset of $\{-y = 0\}$ is labeled with $u = 2$ and $v = [1\ 0]^T$ and any subset of $\{y = 1\}$ is labeled with $u = 1$ and $v = [0\ 1]^T$. The labels on the transitions are also dropped because each of them can be inferred directly from the successor state. From this automaton, we see that T is reachable from S. Thus, we proceed with the automaton refinement procedure.

To simplify our refinement in this example, we will only refine the sets in the partition corresponding to states that are reachable from S. As an example, consider the refinement of B_1 illustrated in Figure 4b. From the transition diagram of $\mathcal{A}_0$, we see that B_1 has multiple successors, namely A_1, S, and A_2. We find, through the use of backward mapping, the subsets of B_1 from which A_1, S, and A_2 can be reached. Using our notation, we compute $\mathcal{B}_2(A_1 \uparrow [1\ 0]^T \rightarrow B_1)$, $\mathcal{B}_2(S \uparrow [1\ 0]^T \rightarrow B_1)$, and $\mathcal{B}_2(A_2 \uparrow [1\ 0]^T \rightarrow B_1)$. These sets are indicated in Figure 4b. After making all of these sets disjoint, we break B_1 into $\{B_{11}, B_{12}, B_{13}, B_{14}\}$ as shown in Figure 4c. We proceed in the similar manner through all the subsets corresponding to the states that are reachable from S in automaton $\mathcal{A}_0$ and obtain the next partition and the corresponding automaton $\mathcal{A}_1$ shown in Figure 4c.

In the automaton $\mathcal{A}_1$, T is still reachable from S. We refine the automaton once more and obtain the next partition and the automaton $\mathcal{A}_2$ shown in Figure 4d. In $\mathcal{A}_2$, T is no longer reachable from S and we have our verification result. Therefore, the property that T is not reachable in the hybrid system is verified after two iterations of the approximating automata construction algorithm whereas the HyTech fixed-point computation would never



(a) initial partition and automaton $\mathcal{A}_0$ (b) refinement of B_1

(c) automaton $\mathcal{A}_1$ after first refinement (d) automaton $\mathcal{A}_2$ after second refinement

Figure 4:

terminate.

7 Discussion

This paper develops a complete set of computations for generating approximating automata for a particular class of hybrid systems characterized by linear constraints on the derivative of the continuous state vector. This demonstrates the possibility of carrying out the steps of the algorithm proposed in [2] (as well as the algorithm in [3]). The example in section 6 illustrates the use of approximating automata for verification and compares the approach to fixed-point computations used in verification tools for hybrid systems.

The hybrid systems considered herein are similar to the linear hybrid automata. One difference between the models is the use of arbitrary discrete dynamics in the hybrid system in Fig. 1. In linear hybrid automata the discrete dynamics are defined a priori as locations which determine the continuous dynamic modes. In the model described in section 2, the separation of the continuous and discrete dynamics allows for a more compact representation of problems in which the particular discrete dynamics is undetermined. Such situations arise, for example, when the objective is to synthesize a supervisory controller to select the continuous dynamics to satisfy a specification for the sequential behavior of the system as defined by the threshold events (see [3]).

A more rigorous comparison of the model in section 2 and linear hybrid automata should be made. It is evident an equivalent linear hybrid automaton can always be constructed when the discrete dynamics in Fig. 1 are deterministic. On the other hand, the invariants of linear hybrid automata and nondeterminism in firing times for events defined by guards can be represented by thresholds and freedom in the linear dynamics in the model in Fig. 1. The model in Fig. 1 does not admit jumps in the continuous state, which implies the linear hybrid automata are more general. (A more general model similar to Fig. 1 that admits jumps in the continuous state is described in [5].) In our opinion, the important difference in the models is the perspectives they represent. Linear hybrid automata emphasize the discrete state aspects of the system, whereas the model in Fig. 1 emphasizes the continuous dynamics and a modular, signal-flow view of the world. Each perspective is useful for particular applications.

Current research is focusing on algorithms for constructing approximating automata for hybrid systems with more general continuous dynamics. There are few applications where the continuous dynamics behave according to simple polytope constraints on the derivative of the continuous state vector. We are also investigating the application of approximating automata to the verification and synthesis of switching control laws in safety-critical systems. Results of this research will be reported in future papers.

Appendix: Proofs

Proof of Lemma 1

Suppose $\dot{x}_1(t)$, $0 \le t \le t_f$ is piecewise continuous with K pieces. Let $t_0 = 0$ and $0 < t_1 < \ldots < t_K$ be the time at the end point of each piece. Note that $t_K = t_f$. For the k^{th} piece, $1 \le k \le K$, we apply the Mean Value Theorem to obtain

$$\frac{x_1(t_k) - x_1(t_{k-1})}{t_k - t_{k-1}} = \dot{x}_1(t_k^*),\ t_{k-1} \le t_k^* \le t_k$$

Let $s_k = \dot{x}_1(t_k^*)$ and $x_k = x_1(t_k)$. Note that $x_K = x_1(t_K) = x_1(t_f) = x_f$. We rewrite the above expression as

$$x_k - x_{k-1} = s_k(t_k - t_{k-1})$$

It then follows that

$$\sum_{k=1}^{K}(x_k - x_{k-1}) = x_K - x_0 = \sum_{k=1}^{K} s_k(t_k - t_{k-1})$$

Dividing by t_K and letting $\alpha_k = (t_k - t_{k-1})/t_K$, we have that

$$\frac{x_K - x_0}{t_K} = \sum_{k=1}^{K} \alpha_k s_k$$

Since α_k is nonnegative and

$$\sum_{k=1}^{K} \alpha_k = \frac{1}{t_K}\sum_{k=1}^{K} t_k - t_{k-1} = \frac{t_K - t_0}{t_K} = 1,$$

the quantity $s_f = \sum_{k=1}^{K} \alpha_k s_k$ is a convex combination of s_k. And because $s_k \in F_u$ $\forall 1 \le k \le K$, we have that $s_f \in F_u$. Using the above results, we write

$$\frac{x_K - x_0}{t_K} = s_f$$

We construct the trajectory

$$x_2(t) = x_0 + s_f t$$

Recalling that $x_K = x_f$ and $t_K = t_f$, $x_2(t)$ is clearly a straight line connecting x_0 and x_f. It also reaches x_f at t_f. Finally, $x_2(\cdot) \in \mathcal{X}(x_0, F_u)$ because $x_2(0) = x_0$, $\dot{x}_2(t)$ is piecewise continuous (with only one continuous piece), and $\dot{x}_2(t) = s_f \in F_u$, $\forall t \ge 0$.

■

Proof of Lemma 2 (by contradiction)

Suppose that the trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$, where $x_0 \in S$, triggers $\hat{v}$ at $x_f \in T$ before any other event but $x_0 \notin CAN(S \to T \uparrow \hat{v})$. Then x_0 must fall into at least one of the following cases.

Case 1 $x_0 \notin S$. This contradicts the assumption that $x_0 \in S$.

Case 2 $x_0 \in \bigcup_{i \text{ s.t. } T \subset M_i^+} M_i^-$. Consider one threshold hyperplane M_i such that $x_0 \in M_i^-$ and $T \subset M_i^+$. Since T is not contained in M_i, $\hat{v}_i = 0$. Since $x_0 \in M_i^-$ and $T \subset M_i^+$, any trajectory from x_0 to T must cross M_i from the negative side and trigger $\hat{v}_i$ before reaching T, a contradiction.

Case 3 $x_0 \in \bigcup_{i \text{ s.t. } T \subset M_i, \hat{v}_i = 0} M_i^-$. Consider one threshold hyperplane M_i such that $x_0 \in M_i^-$ and $T \subset M_i, \hat{v}_i = 0$. In order to avoid triggering $\hat{v}_i$, $x(\cdot)$ must approach x_f from M_i^{0+}. However, since $x_0 \in M_i^-$, $x(\cdot)$ has to cross M_i from the negative side before it can enter M_i^{0+} and trigger $\hat{v}_i$, a contradiction.

■

Proof of Proposition 1 (by contradiction)

Suppose there is a trajectory $x(\cdot) \in \mathcal{X}(x_0, u)$ from a state $x_0 \in S$ which reaches T, triggers $\hat{v}$ before any other event but never enters $FVIA_u(S \to T \uparrow \hat{v})$. Let t_f be time when $\hat{v}$ is triggered, i.e. $x(t_f) \in T$. Then $x(t)$ must remain outside of $FVIA_u(S \to T \uparrow \hat{v})$ for all $0 \le t \le t_f$. It then follows that at any time $0 \le t_1 < t_f$, at least one of the following is true.

1. $x(t_1) \notin Reach_{F_u}(CAN(S \to T \uparrow \hat{v}))$
2. $x(t_1) \in \bigcup_{i \text{ s.t. } T \subset M_i^+} M_i^-$
3. $x(t_1) \notin R_{T \uparrow v}$

Case 1 $x(t_1) \notin \mathit{Reach}_{F_u}(CAN(S \to T \uparrow \hat{v}))$. This implies that $x_0 \notin CAN(S \to T \uparrow \hat{v})$. Then, by Lemma 2, we have that $x(\cdot)$ cannot reach T without triggering some other event before $\hat{v}$, a contradiction.

Case 2 $x(t_1) \in \bigcup_{i \text{ s.t. } T \subset M_i^+} M_i^-$. In this case, there are some indices i such that $T \subset M_i^+$ and $x(t_1) \in M_i^-$. Let $\mathcal{I}$ be the set of all such i. By assumption, $x(t)$ must eventually reach T at time $t_f > t_1$. Since T and $x(t_1)$ bracket M_i for all $i \in \mathcal{I}$, $x(t)$ must have crossed all M_i from the negative side *before* it reaches T. Therefore, $x(t)$ must have generated some other events before reaching T, a contradiction.

Case 3 $x(t_1) \notin R_{T\uparrow v}$. From Case 1 and 2, we have seen that $x(t)$ must remain in $\mathit{Reach}_{F_u}(S) \bigcap_{i \text{ s.t. } T \subset M_i^+} M_i^{0+}$ for all $0 \le t < t_f$ in order to have the possibility to satisfy our assumption. So we assume in this case that $x(t)$ remains in $\mathit{Reach}_{F_u}(S) \bigcap_{i \text{ s.t. } T \subset M_i^+} M_i^{0+}$ at all time before reaching T. However, the definition of the event signal $v(\cdot) = ZD(d - Cx(\cdot))$ requires that $x(t)$ enters $R_{T\uparrow\hat{v}}$ just before $\hat{v}$ is triggered. Thus, there exists a time $t_2 > t_1$ such that $x(t_2) \in R_{T\uparrow\hat{v}}$ and $x(t_2) \in \mathit{Reach}_{F_u}(S) \bigcap_{i \text{ s.t. } T \subset M_i^+} M_i^{0+}$. Clearly, $x(t_2) \in FVIA_u(S \to T \uparrow \hat{v})$, a contradiction.

■

Proof of Proposition 2

Suppose x_f and x_m satisfy the above assumption. By Lemma 1, we can construct a straight line trajectory $x(t) = x_m + st$ from x_m to x_f with $x(t_f) = x_f$ for some $t_f > 0$ and $s \in F_u$. We argue that this trajectory never generates any event before it reaches x_f. There are 4 types of event hyperplanes that can be encountered by $x(\cdot)$ during time $0 \le t < t_f$.

Case 1 M_i s.t. $T \subset M_i$ and $\hat{v}_i = 1$. From the definition of $R_{T\uparrow\hat{v}}$, $x_m \in M_i^-$. Since $x(t)$ is a line going from $x_m \in M_i^-$ to $x_f \in T \subset M_i$, all of these M_i are first reached at t_f from the negative side. Thus, none of these $\hat{v}_i$ are triggered during $0 \le t < t_f$.

Case 2 M_i s.t. $T \subset M_i$ and $v_i = 0$. From the definition of $R_{T\uparrow\hat{v}}$, $x_m \in M_i^{0+}$. Since $x(t)$ is a line going from $x_m \in M_i^{0+}$ to $x_f \in M_i$, $x(t)$ remains in M_i^{0+} for all $0 \le t \le t_f$. Thus, v_i cannot be triggered for all these M_i.

Case 3 M_i s.t. $T \subset M_i^+$. From the definition of $FVIA_u(S \to T \uparrow \hat{v})$, $x_m \in M_i^{0+}$. Since $x(t)$ is a line going from $x_0 \in M_i^{0+}$ to $x_f \in M_i^+$, $x(t)$ remain in M_i^{0+} for all $0 \le t \le t_f$. Thus, $\hat{v}_i$ cannot be triggered for all these M_i.

Case 4 M_i s.t. $T \subset M_i^-$. x_m may be on either side of M_i but $x_f \in M_i^-$. Even if $x(t)$ crosses M_i it does so from the positive side and, therefore, cannot trigger $\hat{v}_i$.

From Case 1, we see that all the events of interest in $\hat{v}$ is triggered for the first time at t_f. From Cases 2, 3, and 4, we see that all the other events are not triggered at all. Thus, $x(t)$ is one such trajectory that reaches T and triggers $\hat{v}$ before any other event.

■

Proof of Proposition 3

$\Rightarrow$ We prove this by contraposition. Suppose $x_f \notin FMAP_u(S \to T \uparrow \hat{v})$. Then, $x_f \notin Reach_{F_u}(FVIA_u(S \to T \uparrow \hat{v}))$ or $x_f \notin T$.

Case 1 $x_f \notin T$, then by definition $x_f \notin \mathcal{M}_u(S \to T \uparrow \hat{v})$.

Case 2 $x_f \notin Reach_{F_u}(FVIA_u(S \to T \uparrow \hat{v}))$. Assume also for this case that $x_f \in T$. By Proposition 1, any trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$ from $x_0 \in S$ which reaches T and triggers $\hat{v}$ before any other event must pass through $FVIA_u(S \to T \uparrow \hat{v})$. Thus, for x_f to belong to one such trajectory, it must be reachable from $FVIA_u(S \to T \uparrow \hat{v})$. Thus, there is no admissible trajectory from S that reaches x_f and triggers $\hat{v}$ before any other event and we have that $x_f \notin \mathcal{M}_u(S \to T \uparrow \hat{v})$.

$\Leftarrow$ Suppose $x_f \in FMAP_u(S \to T \uparrow \hat{v})$. Since $x_f \in Reach_{F_u}(FVIA_u(S \to T \uparrow \hat{v}))$ and $FVIA_u(S \to T \uparrow \hat{v}) \subset Reach_{F_u}(CAN(S \to T \uparrow \hat{v}))$, there exists a trajectory $x_a(\cdot) \in \mathcal{X}(x_0, u)$ for some $x_0 \in CAN(S \to T \uparrow \hat{v})$ to $x_f \in T$ via some $x_m \in FVIA_u(S \to T \uparrow \hat{v})$. Without loss of generality, we assume that x_m is in $\bigcap_{i \text{ s.t. } T \subset M_i^-} M_i^-$ because $x(t)$ must eventually enter $\bigcap_{i \text{ s.t. } T \subset M_i^-} M_i^-$ in order to reach x_f. Since there exists an admissible trajectory from x_0 to x_m, by Lemma 1, we can construct a trajectory $x_1(t) = x_0 + s_m t$ with $x_1(t_m) = x_0 + s_m t_m = x_m$, which is a straight line connecting x_0 and x_m. There are 4 types of threshold hyperplanes that might be encountered by $x_1(t)$.

Case 1 M_i such that $T \subset M_i^+$. For each of these hyperplanes, both x_0 and x_m are in M_i^{0+} by the definition of $CAN(S \to T \uparrow \hat{v})$ and $FVIA_u(S \to T \uparrow \hat{v})$. Thus, the whole line connecting them is also contained in M_i^{0+} for each M_i. Consequently, $x_1(t)$ cannot trigger the event $\hat{v}_i$ corresponding to any of these M_i for all $0 \le t \le t_m$.

Case 2 M_i such that $T \subset M_i^-$. As discussed above, $x_m \in M_i^-$ for each of these M_i. Therefore, even if $x_0 \in M_i^+$ for some M_i, $x_1(t)$ crosses all of such M_i from the positive side. Thus, $x_1(t)$ cannot trigger the event $\hat{v}_i$ corresponding to any of these M_i for all $0 \le t \le t_m$.

Case 3 M_i such that $T \subset M_i, \hat{v}_i = 0$. From the definition of $CAN(S \to T \uparrow \hat{v})$ and $FVIA_u(S \to T \uparrow \hat{v})$, for each of these M_i, both x_0 and x_m are in M_i^{0+} and so is the whole line connecting them. Thus, $x_1(t)$ cannot trigger the event $\hat{v}_i$ corresponding to any of these M_i for all $0 \le t \le t_m$.

Case 4 M_i such that $T \subset M_i, \hat{v}_i = 1$. From the definition of $FVIA_u(S \to T \uparrow \hat{v})$, $x_m \in M_i^-$ for all these hyperplanes. By the same argument as the one in Case 2, we have that $x_1(t)$ cannot trigger the event $\hat{v}_i$ corresponding to any of these M_i for all $0 \le t \le t_m$.

From Case 1 to 4, we have that the trajectory $x_1(t)$ cannot trigger any event up to time t_m.

Since $x_m \in FVIA_u(S \to T \uparrow \hat{v})$ and $x_f \in Reach_{F_u}(\{x_m\})$, there exists a trajectory $x_2(t) \in \mathcal{X}(x_m, u)$ which reaches $x_f \in T$ and triggers $\hat{v}$ before any other event by

Proposition 2. Thus, we can construct a single trajectory that triggers $\hat{v}$ before any other event from x_0 to x_f via x_m by combining $x_1(t)$ and $x_2(t)$ as shown below.

$$x(t) = \begin{cases} x_1(t) & 0 \le t < t_m \\ x_2(t - t_m, x_m) & t \ge t_m \end{cases}$$

The above result implies the left hand side of Proposition 3.

■

Proof of Proposition 4 (by contradiction)

Suppose there is a trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$ which reaches T, triggers $\hat{v}$ before any other event, but never enters $BVIA_u(T \uparrow \hat{v})$. Let t_f be the time when $\hat{v}$ is triggered. Then $x(t)$ must remain outside of $BVIA_u(T \uparrow \hat{v})$ for all $0 \le t < t_f$. In other words, at any time $0 \le t_1 < t_f$, at least one of the following is true.

1. $x(t_1) \notin Reach_{-F_u}(T)$
2. $x(t_1) \in \bigcup_{i, T \subset M_i^+} M_i^-$
3. $x(t_1) \notin R_{T\uparrow\hat{v}}$

Case 1 $x(t_1) \notin Reach_{-F_u}(T)$. In this case $x(t)$ cannot reach T for all $t \ge t_1$, a contradiction.

Case 2 $x(t_1) \in \bigcup_{i, T \subset M_i^+} M_i^-$. In this case, we have a contradiction by the same argument as discussed in Case 2 of the proof of Proposition 1.

Case 3 $x(t_1) \notin R_{T\uparrow\hat{v}}$. In this case, we have a contradiction by the same argument as discussed in Case 3 of the proof of Proposition 1.

■

Proof of Proposition 5

Suppose $x_m \in BVIA_u(T \uparrow \hat{v})$. Since x_m must be in $Reach_{-F_u}(T)$, Lemma 1 implies that there exists a state $x_f \in T$, a vector $-s \in -F_u$, and a trajectory $x_r(t) = x_f - st$ such that $x_r(t_f) = x_m$, which is a straight line connecting x_f and x_m. Since $-s \in -F_u \Rightarrow s \in F_u$, we can construct a forward straight line trajectory $x(t) = x_m + st$ from x_m to x_f with $x(t_f) = x_f$. This trajectory never generates any event before it reaches x_f by the same argument discussed in the proof of Proposition 2.

■

Proof of Proposition 6

$\Rightarrow$ We prove this by contraposition. Suppose $x_0 \notin BMAP_u(T \uparrow \hat{v})$, then x_0 must fall into at least one of the following cases.

Case 1 $x_0 \notin Reach_{-F_u}(BVIA_u(T \uparrow \hat{v}))$. In this case, any trajectory from x_0 can never reach a state in $BVIA_u(T \uparrow \hat{v})$. By Proposition 4, such trajectory cannot reach T and trigger $\hat{v}$ before any other event.

Case 2 $x_0 \in \bigcup_{i \text{ s.t. } T \subset M_i^+} M_i^-$. In this case, no trajectory from x_0 can reach T and trigger $\hat{v}$ before any other event (see Case 2 in the proof of Lemma 2).

Case 3 $x_0 \in \bigcup_{i \text{ s.t. } T \subset M_i, \hat{v}_i = 0} M_i^-$. In this case, no trajectory from x_0 can reach T and trigger $\hat{v}$ before any other event (see Case 3 in the proof of Lemma 2).

$\Leftarrow$ Suppose $x_0 \in BMAP_u(T \uparrow \hat{v})$.

If x_0 is also in $VIA_u(T \uparrow \hat{v})$, the implication follows from Proposition 5.

Now suppose $x_0 \notin VIA_u(T \uparrow \hat{v})$. Since $x_0 \in Reach_{-F_u}(VIA_u(T \uparrow \hat{v}))$ and $VIA_u(T \uparrow \hat{v}) \subset Reach_{-F_u}(T)$, there exists a trajectory $x_a(\cdot) \in \mathcal{X}(x_0, F_u)$ from x_0 to $x_f \in T$ via $x_m \in VIA_u(T \uparrow \hat{v})$. Without loss of generality, we assume that x_m is in $\bigcap_{i, T \subset M_i^-} M_i^-$. This is because $x_a(\cdot)$ must eventually enter $\bigcap_{i, T \subset M_i^-} M_i^-$ in order to reach T. We can construct a two-piece linear trajectory from x_0 to x_m and from x_m to x_f and show that this trajectory triggers $\hat{v}$ before any other event at x_f as discussed in the Proof of Proposition 3.

■

Proof of Proposition 7

We prove that $x \in Reach_{F_u}(t_1, x_0) \iff x \in x_0 + F_u t_1$

$\Rightarrow$ Suppose $x \in Reach_{F_u}(t_1, x_0)$. Then there exists a trajectory $x_1(\cdot) \in \mathcal{X}(x_0, u)$ such that $x = x_1(t_1)$. By Lemma 1, there exists another admissible trajectory $x_2(t) = x_0 + wt$ such that $x = x_0 + wt_1$ for some $w \in F_u$. Thus, we have that $x \in x_0 + F_u t_1$.

$\Leftarrow$ Suppose $x \in x_0 + F_u t_1$. Then $x = x_0 + wt_1$ for some $w \in F_u$. We construct a trajectory $x(t) = x_0 + wt$. Clearly, $x(\cdot) \in \mathcal{X}(x_0, u)$ and $x(t_1) = x$, so we have that $x \in Reach_{F_u}(t_1, x_0)$.

■

Proof of Proposition 8

$\Rightarrow$ We prove this implication by contradiction. Suppose that $I_{x_0} \neq \emptyset$ and $\forall w \in F_u, \exists i \in I_{x_0}$ s.t. $c_i^T w > 0$. For each w, we pick one such index and denote it by i_w. Since $c_{i_w}^T x_0 < d_{i_w}$ (by Definition 3) and $c_{i_w}^T w > 0$, there exists a time t_w such that for $t \geq t_w$, $c_{i_w}^T (x_0 + wt) \geq d_{i_w}$ or, equivalently, $x_0 + wt \notin M_{i_w}^-$. Specifically, define

$$t_w = \frac{d_{i_w} - c_{i_w}^T x_0}{c_{i_w}^T w}$$

Now, let

$$t^* = \max_{w \in F_u} t_w$$

It is clear that for all $w \in F_u$, $c_{i_w}^T(x_0 + wt^*) \geq d_{i_w}$. It follows that for all $w \in F_u$, there exists an index $i \in I_{x_0}$, namely i_w, such that $x_0 + wt^* \notin M_i^-$. This, in turns, implies that for all $w \in F_u$, $x_0 + wt^* \notin S_{x_0}$. By Proposition 7, we have that $x_0 + F_u t^* = Reach_{F_u}(t^*, x_0) \subset \overline{S_{x_0}}$. Thus, for any trajectory $x(\cdot) \in \mathcal{X}(x_0, u)$, $x(t^*) \in M_i^{0+}$ for some $i \in I_{x_0}$. But $x(0) = x_0 \in M_i^-$ by Definition 3, so $x(\cdot)$ must have crossed M_i from the negative side, triggering the event v_i, before or at t^*. Therefore, we conclude that $x_0 \notin \mathcal{N}_u$.

$\Leftarrow$

Case 1 If $I_{x_0} = \emptyset$, then $x_0 \in M_i^{0+}$ for all i. We choose any vector $w \in F_u$ and construct an admissible trajectory $x(t) = x_0 + wt$. This trajectory can never generate any event. This is because even if $x(\cdot)$ ever crosses any hyperplane M_i, it does so from the M_i^+ side and, therefore, cannot generate the event v_i.

Case 2 If $\exists w \in F_u$ s.t. $\forall i \in I_{x_0}, c_i^T w \leq 0$, we choose one such w and construct an admissible trajectory $x(t) = x_0 + wt$. We argue that this trajectory can never generate any event. For $i \notin I_{x_0}$, $x_0 \in M_i^{0+}$. The trajectory can never generate the event v_i by the same reasoning as in Case 1. For $i \in I_{x_0}$, $c_i^T x_0 < d_i$. Since $c_i^T \dot{x}(t) = c_i^T w \leq 0$, $c_i^T x(t) < d_i$ for all $t \geq 0$. Consequently, $x(t)$ can never generate the event v_i. Thus, for all i, $x(\cdot)$ can never generate v_i.

From Case 1 and Case 2, we conclude that $x_0 \in \mathcal{N}_u$.

■

Acknowledgment

This research was supported in part by DARPA contract F33615-97-C-1012.

References

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

[2] A. Chutinan and B.H. Krogh. Computing approximating automata for a class of hybrid systems. *Mathematical Modeling of Systems: Special Issue on Discrete Event Models of Continuous Systems*, submitted.

[3] J.E.R. Cury, B.H. Krogh, and T. Niinomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Trans. on Automatic Control*, to appear in the Special Issue on Hybrid Systems, 1997. a preliminary version of this paper also appeared in *Proc. 34th IEEE CDC, pp. 1461-1466*.

[4] P.W. Glynn. A gsmp formalism for discrete event systems. *Proceedings of the IEEE*, 77(1):14–23, Jan 1989.

[5] B.H. Krogh. Condition/event signal interfaces for block diagram modeling and analysis of hybrid systems. In *Proc. 1993 IEEE International Symposium on Intelligent Control*, pages 180–185, Aug 1993.

[6] R.P. Kurshan. *Computer-Aided Verfication of Coordinating Processes.* Princeton University Press, Princeton, NJ, 1994.

[7] K.L. McMillan. *Neuro-Dynamic Programming.* Klumer, 1993.

[8] T. Niinomi and B.K. Krogh. Modeling and analsyis of swithced-mode hybrid systems driven by threshold events. In H. Khalil, J. Chow, and P.A. Ioannou, editors, *Proc. of Workshop on Advances in Control Systems and its Applications*, pages 155–172. Springer-Verlag, 1995.

[9] T. Niinomi, B.K. Krogh, and J.E.R. Cury. Refinements of approximating automata for synthesis of supervisory controllers for hybrid systems. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, pages 475–484. Springer-Verlag, 1996.

[10] J. Raisch and S.D. O'Young. Discrete approximation and supervisory control of continuous systems. *IEEE Trans. on Automatic Control*, page to appear, 1998.

[11] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, Jan 1989.