Proceedings of the 38th
Conference on Decision & Control
Phoenix, Arizona USA • December 1999

Controller Synthesis for Hybrid Systems with Lower Bounds on Event Separation

Andrea Balluchi[†] Luca Benvenuti[†] Tiziano Villa[†] Howard Wong-Toi[‡]
Alberto L. Sangiovanni-Vincentelli[†§]

[†] PARADES, Via San Pantaleo, 66, 00186, Rome, Italy. {balluchi, lucab, villa, alberto}@parades.rm.cnr.it.
[‡] Cadence Berkeley Labs, 2001 Addison St., Third Floor, Berkeley, CA 94704, USA. howard@cadence.com.
[§] EECS Dept., University of California at Berkeley, CA 94720, USA. alberto@eecs.berkeley.edu.

Abstract

A systematic procedure for synthesizing all full-state feedback controllers for a hybrid system subject to a safety (state-invariance) specification has been proposed by Tomlin, Lygeros, and Sastry [6, 5]. The interaction between the controller and a nondeterministic hybrid plant is viewed as a two-person game. The controller wins if it keeps the state of the closed-loop system within a specified set of good states; its adversarial environment tries to force the system outside the good set. The synthesis procedure iteratively augments the set of states from which the environment wins via either one additional discrete step, or one additional continuous flow.

The key difficulty in carrying out the synthesis procedure lies in the computations for continuous flows. One must essentially solve a series of differential games in which the environment is trying to drive the system into its target set at the same time as avoiding the target set of the controller. In this paper, we study hybrid systems with lower bounds on the separation between occurrence times of consecutive discrete moves. These systems arise when modeling minimal delay times between events, either in the controller, or in the environment. For such systems, we provide techniques for solving the differential games in reduced state spaces. The main idea is to discretize information about whether discrete moves are enabled or not.

1 Introduction

A systematic procedure for synthesizing all full-state feedback controllers for a hybrid system subject to a *safety* specification has been proposed by Tomlin, Lygeros, and Sastry [6, 5]. A safety specification is a state-invariance property, and specifies a set of good states within which the closed-loop system must remain. The interaction between the controller and a nondeterministic hybrid plant is viewed as a two-person zero-sum game. Each player moves by setting both discrete and continuous control inputs. The controller wins if the state of the closed-loop system remains within the set of good states; the environment tries to force the system outside the good set. The synthesis procedure computes the set of states from which the enviroment wins, starting from the set of bad states, and iteratively adding the set of states from which it wins via either one additional discrete step, or one additional continuous flow.

The primary obstacle in executing the synthesis procedure lies in the computation for continuous flows. While the necessary calculations can be performed manually in some cases [6, 5], they quickly become complicated in even low-dimensional linear systems [2, 3]. One approach to this problem is to solve the dynamic game using numerical computation for solving partial differential equations [7].

In this paper, we advocate a different approach. We present techniques to exploit the structure of the closed-loop system during the computational step for continuous flows. Our method applies to systems where there is a lower bound on the delay time between each player's discrete moves. Such systems can model communication or computational delays that prevent a player from making distinct discrete actions in quick succession. The placement of such delay constraints is often used to prevent the synthesis of Zeno controllers which satisfy the safety property only by virtue of enforcing infinitely many events in finite time. Our technique uses information about whether discrete events are enabled or not to decompose complicated dynamic games into a series of simpler dynamic games played over lower dimensions.

2 Hybrid Automata

Intuitively, the hybrid automaton models a game board. This modeling formalism merges the game features (explicitly-defined independent moves) of [1] into the hybrid automata model (input structure and hybrid dynamics) found in [6, 4].

0-7803-5250-5/99/$10.00 © 1999 IEEE

A *hybrid automaton* is a tuple $H = ((Q, X), (U, \Sigma_c), (M_c^{cts}, M_c^{disc}), (D, \Sigma_e), (M_e^{cts}, M_e^{disc}), (f, \delta))$, where

- Q is the finite set of *modes*,
- $X \subseteq \mathbb{R}^n$ is the set of continuous *states*,
- $U \subseteq \mathbb{R}^m$ is the domain of *continuous control values*, $\mathcal{U} = \{u(.) \in PC^0 | u(t) \in U, \forall t \in \mathbb{R}\}$ is the class of control functions,
- Σ_c is the finite domain of *discrete control events*, ($\Sigma_c^\epsilon = \Sigma_c \cup \{\epsilon\}$ is the set of *discrete control moves*, with the special ϵ move being the *silent* move),
- $M_c^{disc} : Q \times X \to 2^{\Sigma_c^\epsilon} \setminus \{\}$ is the *discrete controller feasible move function*,
- $M_c^{cts} : Q \times X \to 2^U \setminus \{\}$ is the *continuous controller feasible move function*,
- $D \subseteq \mathbb{R}^p$ is the domain of *continuous disturbance values*, $\mathcal{D} = \{d(.) \in PC^0 | d(t) \in D, \forall t \in \mathbb{R}\}$ is the class of disturbance functions,
- Σ_e is the finite set of *discrete disturbance events*, ($\Sigma_e^\epsilon = \Sigma_e \cup \{\epsilon\}$ is the set of *discrete disturbance moves*),
- $M_e^{disc} : Q \times X \to 2^{\Sigma_e^\epsilon} \setminus \{\}$ is the *discrete disturbance feasible move function*,
- $M_e^{cts} : Q \times X \to 2^D \setminus \{\}$ is the *continuous disturbance feasible move function*,
- $f : Q \times X \times U \times D \to \mathbb{R}^n$ is the *continuous transition function* that models the time-invariant continuous dynamics, which depend on the mode. It is required that the function f is such that, for every control function $u \in \mathcal{U}$, for every disturbance function $d \in \mathcal{D}$, and for every $x_0 \in X$, there is a unique solution of the differential equation $\dot{x}(t) = f(q, x(t), u(t), d(t))$ with initial value $x(0) = x_0$, which we denote by $x(t) = \psi_q(u|_{[0,t)}, d|_{[0,t)}, x_0, t)$, $\forall t \geq 0$.
- $\delta : Q \times X \times \Sigma_c^\epsilon \times \Sigma_e^\epsilon \to 2^{Q \times X} \setminus \{\}$ is the *discrete transition function* that models the discrete dynamics. We require that for all $(q, x) \in Q \times X$, $\delta(q, x, \epsilon, \epsilon) = \{(q, x)\}$.

Both the controller and the environment make their moves simultaneously. The discrete transition function δ defines the transitions for the joint discrete moves of the controller and the disturbance. At the configuration (q, x), the controller chooses a pair $(\sigma_c, u) \in M_c^{disc}(q, x) \times M_c^{cts}(q, x)$. The environment does likewise, choosing a pair $(\sigma_e, d) \in M_e^{disc}(q, x) \times M_e^{cts}(q, x)$. If either of the players chooses a non-silent discrete move[1], then a non-trivial *discrete jump* takes place, with label (σ_c, σ_e). The discrete transition function δ determines the effect on the system. The resultant configuration is any configuration in $\delta(q, x, \sigma_c, \sigma_e)$. As long as both players choose ϵ as their discrete move, then time may progress and a *continuous evolution* takes place. In this case, the discrete mode remains fixed[2], and the continuous variables evolve according to the continuous dynamics specified by the function f under the continuous control u chosen by the controller and the continuous disturbance d chosen by the environment. We denote by *Wait* the set of configurations in which both players may choose not to play a discrete move, but instead wait for time to pass: $Wait = \{(q, x) \mid \epsilon \in M_c^{disc}(q, x) \text{ and } \epsilon \in M_e^{disc}(q, x)\}$. A *trajectory* of the hybrid automaton is a (finite or infinite) sequence of discrete jumps and continuous evolutions. One may think of the interaction between the players as a continuous game with occasional discrete interruptions where a discrete game takes place.

A *safety property* asserts that nothing bad happens along trajectories. It can be characterized by the set $Good \subseteq Q \times X$ of good configurations that do not violate the property. The hybrid automaton with initial configurations $(Q \times X)_0$ satisfies the safety property *Good* if all its trajectories that start in $(Q \times X)_0$ remain within *Good*.

3 Synthesis of maximal controllers

We review the synthesis methodology introduced in [6]. The design of a controller proceeds in two steps. In the first part of the procedure, the maximal safe set W is computed. By construction, from any configuration $(q, x) \in W$, the controller has a strategy to keep the system forever in W. In the second part of the procedure, the control strategy is explicitly extracted from W.

A controller watches the entire state of the system at all times, and decides whether to (1) take discrete control actions that may cause an instantaneous change in the configuration, or to (2) let time pass under a continuous input u. A feedback memory-less controller for a hybrid automaton is a pair $C = (T^{disc}, T^{cts})$, where $T^{disc} : Q \times X \to 2^{\Sigma_c^\epsilon} \setminus \{\}$ and $T^{cts} : Q \times X \to 2^U \setminus \{\}$ model the values allowed by the controller. The controller can only offer values permitted by the feasible move functions, and hence, for all $(q, x) \in Q \times X$, we require $T^{disc}(q, x) \subseteq M_c^{disc}(q, x)$ and $T^{cts}(q, x) \subseteq M_c^{cts}(q, x)$. The coupling of the hybrid automaton H with the controller $C = (T^{cts}, T^{disc})$ is the closed-loop hybrid automaton $H_C = ((Q, X), (U, \Sigma_c), (T^{cts}, T^{disc}), (D, \Sigma_e), (M_e^{cts}, M_e^{disc}), (f, \delta))$, obtained from H by replacing the discrete controller move function with T^{disc} and the continuous controller move function with T^{cts}.

[1] Different discrete move choices of the players can be modeled as follows. If $M_c^{disc}(q, x) = \{\epsilon\}$, then there is no non-trivial discrete move the controller can take; it can only let time pass. If $M_c^{disc}(q, x) = \{\sigma_c, \epsilon\}$, then it is possible either to let time pass, or to take the discrete move σ_c. If $M_c^{disc}(q, x) = \{\sigma_c\}$, then it is possible to make only the discrete move labeled σ_c, but it is not possible to let time pass (i.e., the move is forced to occur). The use of M_e^{disc} is similar.

[2] The requirement that for all $(q, x) \in Q \times X$, $\delta(q, x, \epsilon, \epsilon) = \{(q, x)\}$ means that if both players agree not to make a non-trivial discrete move, there is no discrete change in configuration.

A configuration (q, x) is *safe* for the automaton H and safety specification *Good* if there exists a controller such that the closed-loop system with initial configuration (q, x) satisfies the specification. A set W of configurations is a *safe set* for the automaton H and safety specification *Good* if $W \subseteq Good$ and every element of W is safe for H and *Good*.

3.1 Maximal Safe Set computation

The procedure to synthesize the maximal controller first computes the maximal controllable safe set [4]. This maximal set is obtained by first overapproximating it with all the safe configurations. Then one calculates all configurations from which the environment can drive the system into an unsafe configuration via either one discrete jump, or one continuous flow. These are the configurations from which the environment can win within one "step", and should be avoided by the controller. One iterates this computation, finding successively the configurations from which the environment can win within i steps. If the procedure terminates, we have determined the maximal controllable safe set. We first define the auxiliary predecessor operators Pre_e, Pre_c, and *Unav*, that are used to capture the winning configurations for the environment[3]. The *discrete uncontrollable predecessors* operator $Pre_e : 2^{(Q\times X)} \to 2^{(Q\times X)}$ is defined as follows:

$$Pre_e(K) = \{(q,x) \in Q \times X : \forall \sigma_c \in M_c^{disc}(q,x). \exists \sigma_e \in M_e^{disc}(q,x). (\sigma_c, \sigma_e) \neq (\epsilon, \epsilon) \wedge \delta(q,x,\sigma_c,\sigma_e) \not\subseteq K\}.$$

The escaping configurations [8, 4] are characterized by the *discrete controllable predecessors* operator $Pre_c : 2^{(Q\times X)} \to 2^{(Q\times X)}$ defined as follows:

$$Pre_c(K) = \{(q,x) \in Q \times X : \exists \sigma_c \in M_c^{disc}(q,x). \forall \sigma_e \in M_e^{disc}(q,x). (\sigma_c, \sigma_e) \neq (\epsilon, \epsilon) \wedge \delta(q,x,\sigma_c,\sigma_e) \subseteq K\}.$$

The configurations that the environment can force into a set in one continuous step are characterized by the *continuous uncontrollable predecessor operator Unav* : $2^{(Q\times X)} \times 2^{(Q\times X)} \to 2^{(Q\times X)}$, defined below. The operator takes two arguments. The first is the set of configurations the environment is trying to reach, and the second is a set it must avoid.

$$Unav(B, E) = \{(q,\hat{x}) \in Q \times X : \forall u \in \mathcal{U}\ \exists \bar{t} > 0\ \exists d \in \mathcal{D} \text{ such that for the trajectory } x(t) = \psi_q(u,d,\hat{x},t) \text{ we have } \forall \tau \in [0,\bar{t})\ (q, x(\tau)) \in Wait \cap \overline{E} \wedge (q, x(\bar{t})) \in B\}.$$

Figure 1 shows the fixed-point computation to obtain the maximal safe set. The procedure successively prunes away configurations that are found to be losing upon one additional discrete step ($Pre_e(W^i)$), or a continuous step to a losing configuration ($Unav(Pre_e(W^i) \cup \overline{W^i}, Pre_c(W^i))$). It is not guaranteed to stop within a finite number of steps.

[3]The operators Pre_e, Pre_c, and *Unav* are dependent on the underlying automaton H.

$W^0 := Good$
$i := -1$
repeat {
  $i := i + 1$
  $W' := W^i \setminus Pre_e(W^i)$
  $W^{i+1} := W' \setminus Unav(Pre_e(W^i) \cup \overline{W^i}, Pre_c(W^i))$
} **until** $(W^{i+1} = W^i)$
$Safe := W^i$

Figure 1: Computation of Maximal Safe Set [6].

3.2 Controller extraction

Extracting the maximal control strategy from the maximal safe set W amounts to determining for every configuration in W, which control choices will keep the system in W. The available control choices either (1) force a discrete control action, or (2) allow time to pass. In case (1), the value of u is irrelevant, since a discrete jump will occur. In case (2), we must choose the input control vector u so that, in the event that the environment also lets time pass, the ensuing continuous flow remains in W. The control strategy (T^{disc}, T^{cts}) derived from W is defined on W by the following three rules:

1. $\sigma_0 \in T^{disc}(q,x)$, if $\sigma_0 \in M_c^{disc}(q,x)\backslash\{\epsilon\}$ and for all $\sigma_1 \in M_e^{disc}(q,x)$, $\delta(q,x,\sigma_0,\sigma_1) \subseteq W$.
2. $\epsilon \in T^{disc}(q,x)$, if $\epsilon \in M_c^{disc}(q,x)$ and
   - for all $\sigma_1 \in M_e^{disc}(q,x)$, $\delta(q,x,\epsilon,\sigma_1) \subseteq W$, and
   - if $\epsilon \in M_e^{disc}(q,x)$, then there must exist a $u \in M_c^{cts}(q,x)$ such that for all $d \in M_e^{cts}(q,x)$, the vector $f(q,x,u,d)$ is in the inward tangent space of W at (q,x);
3. $T^{cts}(q,x) = M_c^{cts}(q,x)$, if $\epsilon \notin T^{disc}(q,x)$, and otherwise for all $u \in U$, we have $u \in T^{cts}(q,x)$ if $u \in M_c^{cts}(q,x)$ and for all disturbances $d \in M_e^{cts}(q,x)$, the vector $f(q,x,u,d)$ is in the inward tangent space of W at (q,x).

4 Controller synthesis with a lower bound on event separation

Many hybrid control systems involve a delay of at least Δ time units between pairs of consecutive discrete events. For example, certain sampling actions may require a minimal set-up time, or there may be a minimal delay between successive control applications. Lower bounds on the time between discrete events may also be added to ensure nonZenoness. Lower bounds on event separations can be enforced by introducing a timer t_c (a timer is a continuous variable with rate of

increment $\dot{t}_c = 1$). Thus the original continuous state space is extended from X to $X \times \mathbb{R}$. Discrete events are enabled when $t_c \geq 0$ and every jump resets the timer t_c to $-\Delta$, so that no discrete event is allowed in the interval $-\Delta \leq t_c < 0$. We could apply the synthesis procedure of Fig. 1 to the hybrid system augmented with variable t_c. However, our previous experience with a heating system shows that the addition of a variable can complicate reasoning about the dynamics of the system substantially [2, 3]. It would be convenient to apply the synthesis procedure to the hybrid system without variable t_c. In this section we develop a revised synthesis procedure that operates over the original continuous state space X, instead of the extended space $\widetilde{X} = (X, t_c)$.

The intuitive idea is that since there is only *one* timer t_c, information about its value can be discretized into the two parts: $t_c = -\Delta$ and $t_c \geq 0$. The continuous computations over the extended $(n+1)$-dimensional state space $\widetilde{X}$ can be replaced with time-bounded computations over the reduced n-dimensional space X. During the main loop of the synthesis procedure, we need only to store two kinds of n-dimensional sets — those representing states for which discrete events are enabled w.r.t. the timer, and those representing states immediately after a jump. In either of these two types of sets, it does not matter what the specific timer value is, because: (1) if $t_c \geq 0$, then the specific value of t_c does not matter, and it suffices to know that a discrete jump is enabled; (2) immediately after a jump, the timer has value $-\Delta$ and need not be stored. For states at which $-\Delta < t_c < 0$, we need to know the value of t_c, but since t_c after a jump is always reset to $-\Delta$, the value of t_c can be determined by knowing the integration time. Thus we can move between the two separated parts for $t_c = -\Delta$ and $t_c \geq 0$ by integrating between them for a fixed time Δ. The proofs of the Lemmas contained in this Section can be found in [3].

4.1 Extension of hybrid automaton with one timer

Let $\widetilde{H} = ((Q, \widetilde{X}), (U, \Sigma_c), (\tilde{M}_c^{cts}, \tilde{M}_c^{disc}), (D, \Sigma_e), (\tilde{M}_e^{cts}, \tilde{M}_e^{disc}), (\tilde{f}, \tilde{\delta}))$, be the hybrid automaton obtained extending the hybrid automation H with one timer t_c:

$\widetilde{X} = X \times \mathbb{R}$,

$$\tilde{M}_c^{disc}(q,(x,t_c)) = \begin{cases} \{\epsilon\} & -\Delta \leq t_c < 0 \\ M_c^{disc}(q,x) & t_c \geq 0 \end{cases},$$

$\tilde{M}_c^{cts}(q,(x,t_c)) = M_c^{cts}(q,x) \quad \forall t_c$

$$\tilde{M}_e^{disc}(q,(x,t_c)) = \begin{cases} \{\epsilon\} & -\Delta \leq t_c < 0 \\ M_e^{disc}(q,x) & t_c \geq 0 \end{cases},$$

$\tilde{M}_e^{cts}(q,(x,t_c)) = M_e^{cts}(q,x) \quad \forall t_c$

$\tilde{f} : Q \times \widetilde{X} \times U \times D \to \mathbb{R}^{n+1}$ are such that at each mode the same flows as in f apply, together with the flow $\dot{t}_c = 1$, i.e. $\tilde{f} = (f, 1)$,

$\tilde{\delta} : Q \times \widetilde{X} \times \Sigma_c^\epsilon \times \Sigma_e^\epsilon \to 2^{Q \times \widetilde{X}} \setminus \{\}$ is defined as

(a) $\tilde{\delta}(q,(x,t_c),\sigma_c,\sigma_e) = \{(q,(x,t_c))\}$, if $-\Delta \leq t_c < 0$ or $t_c \geq 0 \quad \wedge \quad (\sigma_c,\sigma_e) = (\epsilon,\epsilon)$ and

(b) $\tilde{\delta}(q,(x,t_c),\sigma_c,\sigma_e) = \delta(q,x,\sigma_c,\sigma_e) \times \{-\Delta\}$, if $t_c \geq 0 \ \wedge \ (\sigma_c,\sigma_e) \neq (\epsilon,\epsilon)$.

4.2 Timer-reduced sets

The basis of our simplified view of the calculations for continuous flows is that the value of the timer is irrelevant once it has exceeded 0. For instance, when $t_c \geq 0$, it suffices to know the value of X in order to determine the future evolution of the system: the timer value is not relevant, since the lower bound on event separations has passed. In order to capture this notion we introduce the following:

Definition 1 *A set $G \subseteq \widetilde{X}$ is timer-reduced if the set G restricted to the domain where $t_c \geq 0$ is independent[4] of t_c. A set $W \subseteq Q \times \widetilde{X}$ of configurations is timer-reduced if for every mode $q \in Q$, the set $\{\tilde{x} \in \widetilde{X} \mid (q,\tilde{x}) \in W\}$ is timer-reduced.*

Throughout, we use the operators $\widetilde{Pre_c}$, $\widetilde{Pre_e}$, and $\widetilde{Unav}$ decorated with a tilde to indicate that they operate over the extended automaton $\widetilde{H}$ and the operators Pre_c, Pre_e, and $Unav$ which operate over the automaton H. The following lemmas can be proved:

Lemma 1 *For any set W, the sets $\widetilde{Pre_c}(W)$ and $\widetilde{Pre_e}(W)$ are timer-reduced. Moreover, if B and E are timer-reduced sets, then the set $\widetilde{Unav}(B,E)$ is timer-reduced.* ◁

Lemma 2 *If the specification Good is timer-reduced, then the set Safe and also every set W^i computed in the synthesis procedure of Figure 1 is timer-reduced.* ◁

4.3 Projections of maximal safe set computations

In the previous Section, we stated that when $t_c \geq 0$ the operators $\widetilde{Pre_c}$, $\widetilde{Pre_e}$, and $\widetilde{Unav}$ preserve independence from t_c. For easier algebra, it is convenient to introduce the following projections operators, where $W \subseteq Q \times \widetilde{X}$ is a set of configurations: $\pi_{-\Delta} : Q \times \widetilde{X} \to Q \times X$, and $\pi_0 : Q \times \widetilde{X} \to Q \times X$, are such that

$$\pi_{-\Delta}(W) = \{(q,x) \in Q \times X | (q,x,-\Delta) \in W\}$$
$$\pi_0(W) = \{(q,x) \in Q \times X | (q,x,0) \in W\}$$

If W is timer-reduced, then

$$\pi_0(W) \times [0,\infty) = W \cap (Q \times X \times [0,\infty))$$

and the set $\pi_0(W) \subseteq \mathbb{R}^n$ captures exactly the part of $W \subseteq \mathbb{R}^{n+1}$ for which $t_c \geq 0$. We will show that the computation of the safe set can be carried out using only the projections of the sets W for $t_c = -\Delta$ and

[4] A set $G \subseteq \Pi_{j=1..k} Y_j$ for domains Y_j is *independent* of the variable y_i having domain Y_i if the following holds: $\forall_{j \neq i}\ y_j \in Y_j$. $\forall\ y', y'' \in Y_i$. $(y_1,\ldots,y_{i-1},y',y_{i+1},\ldots,y_k) \in G$ iff $(y_1,\ldots,y_{i-1},y'',y_{i+1},\ldots,y_k) \in G$. In other words, membership of a point y in G can be determined independently of the value of y_i.

$t_c \geq 0$. We study first the discrete controllable and uncontrollable predecessor operators.

Lemma 3

(1) $\pi_{\text{-}\Delta}(\widetilde{Pre_c}(W)) = \emptyset$
(2) $\pi_{\text{-}\Delta}(\widetilde{Pre_e}(W)) = \emptyset$
(3) $\pi_0(\widetilde{Pre_c}(W)) = Pre_c(\pi_{\text{-}\Delta}(W))$
(4) $\pi_0(\widetilde{Pre_e}(W) = Pre_e(\pi_{\text{-}\Delta}(W))$ ◁

The continuous uncontrollable predecessors we are interested in consist of

1. the configurations that start from $t_c \geq 0$ and unavoidably lose at $t_c \geq 0$,
2. the configurations that start from $t_c = -\Delta$ and
   (a) unavoidably lose at $-\Delta < t_c < 0$,
   (b) unavoidably lose at $t_c = 0$
   (c) unavoidably lose at $t_c > 0$.

We show how the set can be computed using projections onto the $t_c \geq 0$ subspace and the $t_c = -\Delta$ subspace. The configurations defined by case 1 are handled by the following result.

Lemma 4 $\pi_0(\widetilde{Unav}(\widetilde{Pre_e}(W) \cup \overline{W}, \widetilde{Pre_c}(W))) =$

$Unav(Pre_e(\pi_{\text{-}\Delta}(W)) \cup \pi_0(\overline{W}), Pre_c(\pi_{\text{-}\Delta}(W)))$ ◁

The configurations defined by case 2 are handled by the following lemma.

Lemma 5 If W is timer–reduced, then

$\pi_{\text{-}\Delta}(\widetilde{Unav}(\widetilde{Pre_e}(W) \cup \overline{W}, \widetilde{Pre_c}(W))) =$
$\{(q, \hat{x}) \in Q \times X \mid \forall u \in \mathcal{U}\ \exists\, \bar{t} \in (0, \Delta]\ \exists d \in \mathcal{D}$ *such that for the trajectory* $x(t) = \psi_q(u, d, \hat{x}, t)$ *we have*
$(q, x(\bar{t}), -\Delta + \bar{t}) \in \overline{W} \wedge \bar{t} < \Delta \vee$
$(q, x(\Delta)) \in \pi_0(\widetilde{Pre_e}(W) \cup \overline{W} \cup$
$\widetilde{Unav}(\widetilde{Pre_e}(W) \cup \overline{W}, \widetilde{Pre_c}(W))) \wedge \bar{t} = \Delta\}$ ◁

Condition $(q, x(\bar{t}), -\Delta + \bar{t}) \in \overline{W}$ on the right hand side of the previous equation collects the configurations defined by case 2.a, while condition $(q, x(\Delta)) \in \pi_0(\widetilde{Pre_e}(W) \cup \overline{W} \cup \widetilde{Unav}(\widetilde{Pre_e}(W) \cup \overline{W}, \widetilde{Pre_c}(W)))$ collects those defined by case 2.b and case 2.c. In particular the term $\widetilde{Pre_e}(W) \cup \overline{W}$ in $\pi_0(\cdot)$ is related to case 2.b, and the term $\widetilde{Unav}(\widetilde{Pre_e}(W) \cup \overline{W}, \widetilde{Pre_c}(W))$ to case 2.c.

From Lemmas 4 and 5, we almost obtain a means to compute the continuous uncontrollable predecessors using only sets in n-dimensional space. The remaining obstacle is the use of $W \subseteq \widetilde{X}$ in the condition for case 2.a above. It appears as if one needs to know the record the states of W for which $-\Delta < t_c < 0$.

However, the following observation solves our problem.

Observation 1 In the synthesis procedure of Fig. 1, the target set $Pre_e(W^i) \cup \overline{W^i}$ of the environment in the continuous game can be replaced by the set

```
// Good ⊆ Q × X is independent of t_c
W_0^0 := Good;   W_-Δ^0 := Good
i := −1
repeat {
  i := i + 1
  W' := W_0^i \ Pre_e(W_-Δ^i)
  W_0^{i+1} := W' \ Unav(Pre_e(W_-Δ^i) ∪ overline(W_0^i), Pre_c(W_-Δ^i))
  W_-Δ^{i+1} := W_-Δ^i \ Unav_(-Δ,0](overline(Good), W_0^{i+1})
} until (W_0^{i+1} = W_0^i and W_-Δ^{i+1} = W_-Δ^i)
Safe_0 := W_0^i;   Safe_-Δ := W_-Δ^i
```

Figure 2: Computation of Maximal Safe Set with Projection of 1 Timer.

$Pre_e(W^i) \cup \overline{Good}$. This reduction in target set yields the same set of continuous uncontrollable predecessors $Unav(Pre_e(W^i) \cup \overline{Good}, Pre_c(W^i)) = Unav(Pre_e(W^i) \cup \overline{W^i}, Pre_c(W^i))$ because every trajectory that arrives at a state in $Pre_e(W^i) \cup \overline{W^i}$ is also winning for the environment in the game with target $Pre_e(W^i) \cup \overline{Good}$: the game either continues to a state from which there is a "bad" jump (out of $Pre_e(W^i)$ which increases monnotonically with i), or to a state which violates the specification (i.e., a state in $\overline{Good}$).

Lemmas 4 and 5, together with the above observation, motivate the definition of $Unav_{(-\Delta,0]} : 2^{(Q\times X)} \times 2^{(Q\times X)} \to 2^{(Q\times X)}$ as

$Unav_{(-\Delta,0]}(B_G, B) =$
$\{(q, \hat{x}) \in Q \times X \mid \forall u \in \mathcal{U}\ \exists\, \bar{t} \in (0, \Delta]\ \exists d \in \mathcal{D}$
such that for the trajectory $x(t) = \psi_q(u, d, \hat{x}, t)$ *either*
$\bar{t} < \Delta \wedge (q, x(\bar{t})) \in B_G$ *or*
$\bar{t} = \Delta \wedge (q, x(\Delta)) \in B\}$.

The intention is that the argument B_G will represent the states that violate the specification for some $t_c < 0$, and the argument B will represent the states for which the game is won by the environment for $t_c \geq 0$.

Figure 2 shows the fixed-point computation with timer projection to obtain the maximal safe set. The procedure computes the projections $Safe_{\text{-}\Delta} \subset Q \times X$, for $t_c = -\Delta$, and $Safe_0 \subset Q \times X$, for $t_c = 0$, of the maximal safe set $Safe \subset Q \times \widetilde{X}$ for the hybrid automaton $\widetilde{H}$. The procedure makes use of the definitions of the original hybrid automaton H and proceeds by computing intermediate sets $W_0^i \subset Q \times X$, $W_{\text{-}\Delta}^i \subset Q \times X$ related respectively to $t_c \geq 0$ and $t_c = -\Delta$.

Theorem 1 *The procedure in Fig. 2 applied to the automaton H converges if and only if the procedure in Fig. 1 applied to the automaton $\widetilde{H}$, with $W^0 = Good \times \mathbb{R}$, does*[5]. *If so, $Safe_0 = \pi_0(Safe)$ and*

[5]Since the procedure is applied to the automaton $\widetilde{H}$, then the operators Pre_e, Pre_c and $Unav$ in the procedure in Fig. 1 must be replaced with the operators $\widetilde{Pre_e}$, $\widetilde{Pre_c}$ and $\widetilde{Unav}$, respectively.

$Safe_{-\Delta} = \pi_{-\Delta}(Safe)$.

Sketch of the proof The proof is by induction on the index of iteration i. For our inductive hypothesis, we assert that

$$\pi_0(W^i) = W_0^i \quad \text{and} \quad \pi_{-\Delta}(W^i) = W_{-\Delta}^i.$$

The base case of $i = 0$ is easily established. We need to show that, given the inductive hypothesis above,

$$1)\ \pi_0(W^{i+1}) = W_0^{i+1}$$
$$2)\ \pi_{-\Delta}(W^{i+1}) = W_{-\Delta}^{i+1}.$$

Consider first property (1). According to Fig. 2, we have

$$W_0^{i+1} = W_0^i \setminus [Pre_e(W_{-\Delta}^i) \cup Unav(Pre_e(W_{-\Delta}^i) \cup \overline{W_0^i}, Pre_c(W_{-\Delta}^i))].$$

Applying the induction hypothesis, using the property $\overline{\pi_0(W^i)} = \pi_0(\overline{W^i})$ and by Lemmas 3 and 4, it follows that

$$W_0^{i+1} = \pi_0(W^i) \setminus [\pi_0(\widetilde{Pre}_e(W^i)) \cup \pi_0(\widetilde{Unav}(\widetilde{Pre}_e(W^i) \cup \overline{W^i}, \widetilde{Pre}_c(W^i)))].$$

The result then follows by distributivity of the projection π_0 over the set operations $\setminus$ and $\cup$ applied to timer-reduced sets.

We now establish property (2). According to the procedure in Fig. 2, it is

$$W_{-\Delta}^{i+1} = W_{-\Delta}^i \setminus Unav_{(-\Delta,0]}(\overline{Good}, \overline{W_0^{i+1}}).$$

Applying the induction hypothesis and since we already proved that $\pi_0(W^{i+1}) = W_0^{i+1}$, one has $\overline{W_0^{i+1}} = \overline{\pi_0(W^{i+1})} = \pi_0(\overline{W^{i+1}})$ from which it follows that

$$W_{-\Delta}^{i+1} = \pi_{-\Delta}(W^i) \setminus Unav_{(-\Delta,0]}(\overline{Good}, \pi_0(\overline{W^{i+1}})).$$

Then, using Lemma 5, Observation 1 and Lemma 3, according to which $\pi_{-\Delta}(Pre_e(W^i)) = \emptyset$, one gets

$$W_{-\Delta}^{i+1} = \pi_{-\Delta}(W^i) \setminus [\pi_{-\Delta}(\widetilde{Pre}_e(W^i)) \cup \pi_{-\Delta}(\widetilde{Unav}(\widetilde{Pre}_e(W^i) \cup \overline{W^i}, \widetilde{Pre}_c(W^i)))],$$

The result then follows by distributivity of the projection $\pi_{-\Delta}$ over the set operations $\setminus$ and $\cup$ applied to timer–reduced sets.

5 Conclusion

The modified controller synthesis procedure presented here applies to systems that are subject to lower bounds on the separation times between all discrete events. When both the controller and the environment make their discrete actions independently, this modeling may introduce a false coupling between the enablement of events. We have extended our results to handle systems that have separation bounds on the events of the controller and independent separation bounds on the events of the plant [3]. Furthermore, we have demonstrated the applicability of these reduction methods by synthesizing maximal controllers for the heating system first introduced in [2].

This work is part of a larger project on developing techniques for the synthesis of controllers for various modes of an automotive engine. In particular, we are exploring the idle regime mode of operation, for reduction and approximation methods that, similar to those in this paper, exploit the particular structure of the hybrid systems under control. The goal is to design a feedback controller which keeps the speed of the crankshaft in a specified range (safety specification) robustly with respect to disturbances due to load torques and the inertial load changes due to the driver pushing or releasing the clutch pedal. The available controls are the spark ignition time and the input voltage of the DC motor which drives the throttle valve.

References

[1] E. Asarin, O. Maler, A. Pnueli, J. Sifakis, "Controller Synthesis for Timed Automata", *Proc. System Structure and Control*, IFAC, Elsevier, 1998.

[2] A. Balluchi, L. Benvenuti, T. Villa, H. Wong-Toi, A. L. Sangiovanni-Vincentelli, "Synthesis of Maximal Controllers for Hybrid Systems: Case Study of a Heating System", *Proc. European Control Conference*, Karlsruhe, Germany, 1999, to appear.

[3] A. Balluchi, L. Benvenuti, T. Villa, H. Wong-Toi, A. L. Sangiovanni-Vincentelli, "Nonzeno controller synthesis for a heating system", submitted, 1999.

[4] J. Lygeros, C. Tomlin, S. Sastry, "On Controller Synthesis for Nonlinear Hybrid Systems", *Proc. 37th IEEE Conference on Decision and Control*, pp. 2101-2106, Tampa, FL, USA, 1998.

[5] J. Lygeros, C. Tomlin, S. Sastry, "Controllers for Reachability Specifications for Hybrid Systems", *Automatica*, **35**, no. 3, pp. 349-370, 1999.

[6] C. Tomlin, J. Lygeros, S. Sastry, "Synthesizing Controllers for Nonlinear Hybrid Systems", *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science 1386, pp. 360-373, T. Henzinger and S. Sastry Eds., Springer-Verlag, 1998.

[7] C. Tomlin, J. Lygeros, S. Sastry, "Computing Controllers for Nonlinear Hybrid Systems", *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science 1569, pp. 238-255, F. W. Vaandrager and J. H. van Schuppen Eds., Springer-Verlag, 1999.

[8] H. Wong-Toi, "The Synthesis of Controllers for Linear Hybrid Automata", *Proc. 36th Conference on Decision and Control*, pp. 4607-4612, San Diego, CA, USA, 1997.