Proceedings of the 1993 International Symposium on Intelligent Control Chicago, Illinois, USA - August 1993

Extracting Discrete Event System Models from Hybrid Control Systems

James A. Stiver and Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame, Notre Dame, IN 46556

Abstract

In this paper, the state space partitioning in the interface of the hybrid control system is modeled using covering halfspaces; this formulation extends the model introduced in [7]. The covering halfspaces are used to define a set of plant events and a discrete event system model is generated which captures the behavior of the plant and interface of the hybrid control system.

1 Introduction

A hybrid control system consists of a continuous-time system which is being controlled by a discrete event system. The continuous-time system is referred to as the plant because it is the system under control. The plant is usually termed a "conventional" system as it has a continuous state space and it is described by a set of differential (or difference) equations. The discrete event system, called the controller, has a discrete state space and symbolic input and output. In addition to the plant and controller, there is an interface which provides communication between them. The interface generates symbols for the controller as the state of the plant moves through a partitioned state space. It also converts symbols from the controller into plant inputs. When the plant and interface of a hybrid control system are treated as a single system, they behave like a discrete event system. This is useful because it allows the use of discrete event control theory for hybrid systems. The discrete event system which models the plant and interface is referred to as the *discrete event plant model.*

In this work, using the model for hybrid control systems originally described in [1] and [7], we develop a systematic way of obtaining the discrete event plant model from the description of the hybrid control system. This model can be used to apply discrete event controller design techniques.

2 Hybrid Control System Model

A hybrid control system consists of three parts. The modeling and interactions of these parts are now described.

2.1 Plant

The plant is modeled as a time-invariant, continuous-time system, represented by the familiar equations,

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{r}) \quad (1)$$

$$\mathbf{z} = g(\mathbf{x}) \quad (2)$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{r} \in \mathbb{R}^m$, and $\mathbf{z} \in \mathbb{R}^p$ are the state, input, and output vectors respectively. For the purposes of this work we assume that $\mathbf{z} = \mathbf{x}$.

2.2 Controller

The controller is a discrete event system which is modeled as a deterministic automaton specified by the quintuple, $\{\tilde{S}, \tilde{Z}, \tilde{R}, \delta, \phi\}$, where $\tilde{S}$ is the (possibly infinite) set of states, $\tilde{Z}$ is the set of plant symbols which are generated by events in the plant and make up the controller input, $\tilde{R}$ is the set of controller symbols which constitute the controller's output set, $\delta : \tilde{S} \times \tilde{Z} \to \tilde{S}$ is the state transition function, and $\phi : \tilde{S} \to \tilde{R}$ is the output function. The behavior of the controller is described by

$$\tilde{s}[n] = \delta(\tilde{s}[n-1], \tilde{z}[n]) \quad (3)$$

$$\tilde{r}[n] = \phi(\tilde{s}[n]) \quad (4)$$

where $\tilde{s}[n] \in \tilde{S}, \tilde{z}[n] \in \tilde{Z}$, and $\tilde{r}[n] \in \tilde{R}$. The index n indicates the order of the symbols.

2.3 Interface

The controller and plant cannot communicate directly in a hybrid control system because each utilizes a different type of signal. Thus an interface is required which can convert continuous-time signals to sequences of symbols and vice versa. The interface consists of two memoryless maps, γ and α. The first map, called the actuating function or actuator, $\gamma : \tilde{R} \to \mathbb{R}^m$, converts a sequence of controller symbols to a piecewise constant plant input as follows

$$\mathbf{r} = \gamma(\tilde{r}) \tag{5}$$

The plant input, $\mathbf{r}$, can only take on certain constant values, where each value is associated with a particular controller symbol. Thus the plant input is a piecewise constant signal which may change only when a controller symbol occurs. The second map, the plant symbol generating function or generator, $\alpha : \mathbb{R}^n \to \tilde{Z}$, is a function which maps the state space of the plant to the set of plant symbols as follows

$$\tilde{z} = \alpha(\mathbf{x}) \tag{6}$$

It would appear from Equation 6 that, as $\mathbf{x}$ changes, $\tilde{z}$ also continuously changes. That is, there is a continuous generation of plant symbols by the interface because each state is mapped to a symbol. This is not the case due to the way α is defined as will now be explained.

The plant symbol generating function, α, is designed based on an open covering of the state space of the plant. Consider a collection of I open subsets in $\mathbb{R}^n$ which form an open cover for the plant state space. Let this collection be represented as

$$C = \{c_1 c_2 ... c_I\} \tag{7}$$

The collection consists of open subsets, where each subset is called a *covering event*. The set of covering events is formed by a set of $(n-1)$ dimensional hypersurfaces, which are described by a set of functions, $h_i : \mathbb{R}^n \to \mathbb{R}$. The ith covering event is defined as

$$c_i = \{\mathbf{x} : h_i(\mathbf{x}) < 0\} \tag{8}$$

Let the ith covering event, c_i, be associated with a unique *covering symbol*, $\tilde{c}_i$. The "alphabet" of covering symbols can therefore be represented as

$$\tilde{C} = \{\tilde{c}_1 \tilde{c}_2 ... \tilde{c}_I\} \tag{9}$$

These covering symbols are used to define the *plant symbols* as follows

$$\tilde{z} = \alpha(\mathbf{x}) = \{\tilde{c}_i : \mathbf{x} \in c_i\} \tag{10}$$

As shown in Equation 10, a plant symbol is a collection of covering symbols which defines a region in the state space. It is convenient to treat this collection as a symbol. It is also convenient to index the set of plant symbols with a binary vector $b \in \mathbf{B}^I$. Then $\tilde{c}_i \in \tilde{z}_b$ only if $b_i = 1$.

A set of *plant events* can also be defined based on the covering events.

$$z_b = \{\mathbf{x} : \alpha(\mathbf{x}) = \tilde{z}_b\} \tag{11}$$

A plant symbol is generated only when a new event first occurs. The overall effect is that the state space of the plant is partitioned into a number of regions and each is associated with a unique plant symbol which is generated whenever the state enters that region. Thus, these regions (i.e. plant events) form the equivalence classes of α.

2.4 DES Plant Model

If the plant and interface of a hybrid control system are viewed as a single component, this component behaves like a discrete event system. It is advantageous to view a hybrid control system this way because it allows it to be modeled as two interacting discrete event systems which are more easily analyzed than the system in its original form. The discrete event system which models the plant and interface is called the *DES Plant Model* and is modeled as an automaton similar to the controller. The automaton is specified by a quintuple, $\{\tilde{P}, \tilde{Z}, \tilde{R}, \psi, \xi\}$, where $\tilde{P}$ is the set of states, $\tilde{Z}$ and $\tilde{R}$ are the sets of plant symbols and controller symbols, $\psi : \tilde{P} \times \tilde{Z} \to \tilde{P}$ is the state transition function, and $\xi : \tilde{P} \times \tilde{R} \to \mathbf{P}(\tilde{Z})$ is the enabling function. The enabling function defines which events are enabled for a given state and input. Since there is generally more than one event enabled for a given state and input, the DES plant model is nondeterministic. The state transition function defines the state which results following the occurrence of an event. The state transition function, ψ, is a partial function because some events are never enabled from a given state. This model for the DES plant differs in notation, though not in essence, from that used in [1] and [7]. The change is to facilitate the use of existing DES methods.

The behavior of the DES plant model is as follows

$$\begin{aligned} \tilde{z}[n+1] &\in \xi(\tilde{p}[n], \tilde{r}[n]) && (12) \\ \tilde{p}[n] &= \psi(\tilde{p}[n-1], \tilde{z}[n]) && (13) \end{aligned}$$

where $\tilde{p}[n] \in \tilde{P}, \tilde{r}[n] \in \tilde{R}$, and $\tilde{z}[n] \in \tilde{Z}$. After an input from the controller, one of the enabled events occurs and the state of the DES plant changes according to the state transition function.

3 Obtaining the DES Plant Model

As described in the previous section, the DES plant model is an automaton described by the quintuple, $\{\tilde{P}, \tilde{Z}, \tilde{R}, \psi, \xi\}$. To obtain the DES plant model it is necessary to find each of these. First, $\tilde{Z}$ and $\tilde{R}$ are already specified in the hybrid system model. The set of states, $\tilde{P}$, is determined by the set of plant events. Specifically, for each plant event, z_b, there is a DES plant state, $\tilde{p}_b$, such that whenever $\mathbf{x} \in z_b$, the state of the DES plant will be $\tilde{p}_b$.

The state transition function, ψ is defined as follows

$$\psi(\tilde{p}_a, \tilde{z}_b) = \begin{cases} \tilde{p}_b & \text{if } \exists \tilde{c}_k \ni \tilde{z}_b \in \xi(\tilde{p}_a, \tilde{c}_k) \\ \text{undefined} & \text{if } \quad \text{otherwise} \end{cases}$$

This leaves the enabling function, ξ. The enabling function maps a state and an input to a set of states. We can find ξ by a test which determines whether a given event is in the set of events which are enabled for a certain state and input. $\tilde{z}_b$ is enabled from state $\tilde{p}_a$ by input $\tilde{c}_k$ (i.e. $\tilde{z}_b \in \xi(\tilde{p}_a, \tilde{c}_k)$) if $a \neq b$ and there exists $\mathbf{x}$ such that

$$\forall i \begin{cases} a_i = b_i = 0 & \rightarrow & h_i(\mathbf{x}) \geq 0 \\ a_i = b_i = 1 & \rightarrow & h_i(\mathbf{x}) < 0 \\ a_i = 0, b_i = 1 & \rightarrow & h_i(\mathbf{x}) = 0, \nabla h_i(\mathbf{x}) \cdot f_k(\mathbf{x}) < 0 \\ a_i = 1, b_i = 0 & \rightarrow & h_i(\mathbf{x}) = 0, \nabla h_i(\mathbf{x}) \cdot f_k(\mathbf{x}) > 0 \end{cases} \quad (14)$$

where $f_k(\mathbf{x}) = f(\mathbf{x}, \gamma(\tilde{c}_k))$.

4 Example

The plant is a double integrator

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{r} \quad (14)$$

where $\mathbf{r} \in \{-1, 0, 1\}$ which yields

$$\begin{aligned} f_1(\mathbf{x}) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} && (15) \\ f_2(\mathbf{x}) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} && (16) \\ f_3(\mathbf{x}) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} && (17) \end{aligned}$$

The events are formed by the following two hypersurfaces

$$\begin{aligned} h_1(\mathbf{x}) &= x_1 && (18) \\ h_2(\mathbf{x}) &= x_2 && (19) \end{aligned}$$

Thus, there are two covering events

$$\begin{aligned} c_1 &= \{\mathbf{x} : x_1 < 0\} && (20) \\ c_2 &= \{\mathbf{x} : x_2 < 0\} && (21) \end{aligned}$$

and 2^2 plant events

$$\begin{aligned} z_{00} &= \{\mathbf{x} : x_1 \geq 0, x_2 \geq 0\} && (22) \\ z_{01} &= \{\mathbf{x} : x_1 \geq 0, x_2 < 0\} && (23) \\ z_{10} &= \{\mathbf{x} : x_1 < 0, x_2 \geq 0\} && (24) \\ z_{11} &= \{\mathbf{x} : x_1 < 0, x_2 < 0\} && (25) \end{aligned}$$

Now that the hybrid system has been described, the DES plant model can be obtained. There are four plant symbols which represent the four plant events.

$$\tilde{Z} = \{\tilde{z}_{00}, \tilde{z}_{01}, \tilde{z}_{10}, \tilde{z}_{11}\} \quad (26)$$

There are three controller symbols,

$$\tilde{R} = \{\tilde{r}_1, \tilde{r}_2, \tilde{r}_3\} \quad (27)$$

which provide the three possible plant inputs described above. There are four DES plant states,

$$\tilde{P} = \{\tilde{p}_{00}, \tilde{p}_{01}, \tilde{p}_{10}, \tilde{p}_{11}\} \quad (28)$$

which correspond to the four events.

To find the enabling function, we must look at each state and input. For example, consider

$\xi(\tilde{p}_{10}, \tilde{r}_1)$. $\tilde{z}_{00}$ is enabled because there exists $\mathbf{x} = [0, 1]'$ which satisfies the conditions of equation 14. Also, $\tilde{z}_{11}$ is enabled because there exists $\mathbf{x} = [-1, 0]'$ which satisfies equation 14. $\tilde{z}_{10}$ is not enabled because it has the same index, 10, as the current state and $\tilde{z}_{01}$ is not enabled either because there is no $\mathbf{x}$ which satisfies equation 14. Thus we have:

$$\xi(\tilde{p}_{10}, \tilde{r}_1) = \{\tilde{z}_{00}, \tilde{z}_{11}\} \tag{29}$$

Once the enabling function has been derived, the state transition function is obvious from equation 14. For example,

$$\psi(\tilde{p}_{10}, \tilde{z}_{00}) = \tilde{p}_{00} \tag{30}$$

$$\psi(\tilde{p}_{10}, \tilde{z}_{11}) = \tilde{p}_{11} \tag{31}$$

5 Conclusion

The technique described here to extract the DES plant model is very similar to the test used in variable structure control to determine if a surface is invariant. In our case we are interested in whether or not a trajectory will actually cross a surface, thus representing a state transition in the DES plant model.

When applied to the general case as in this paper, the technique can become cumbersome for systems with many inputs and events. A more streamlined procedure should be available for systems with linear plants and linear hypersurfaces.

References

[1] P. J. Antsaklis, M. D. Lemmon, J. A. Stiver, "Hybrid System Modeling and Event Identification", *Technical Report of the ISIS Group*, ISIS-93-002, University of Notre Dame, Notre Dame IN, January 1993.

[2] R. Grossman, R. Larson, "Viewing Hybrid Systems as Products of Control Systems and Automata", *Proceedings of the 31st Conference on Decision and Control*, pp. 2953-2955, Tucson AZ, December 1992.

[3] L. Holloway, B. Krogh, "Properties of Behavioral Models for a Class of Hybrid Dynamical Systems", *Proceedings of the 31st Conference on Decision and Control*, pp. 3752-3757, Tucson AZ, December 1992.

[4] W. Kohn, A. Nerode, "Multiple Agent Autonomous Hybrid Control Systems", *Proceedings of the 31st Conference on Decision and Control*, pp. 2956-2966, Tucson AZ, December 1992.

[5] P. J. Ramadge, W. M. Wonham, "Supervisory Control of a Class of Discrete Event Processes", *Systems Control Group Report #8515*, University of Toronto, Toronto, Canada, November 1985.

[6] P. J. Ramadge, W. M. Wonham, "The Control of Discrete Event Systems", *Proceedings of the IEEE*, Vol. 71, No. 1, pp. 81-98, January 1989.

[7] J. A. Stiver, P. J. Antsaklis, "Modeling and Analysis of Hybrid Control Systems", *Proceedings of the 31st Conference on Decision and Control*, pp. 3748-3751, Tucson AZ, December 1992.

[8] J. A. Stiver, P. J. Antsaklis, "State Space Partitioning for Hybrid Control Systems", *Proceedings of the American Control Conference*, San Francisco, California, June 2-4, 1993.

[9] W. M. Wonham, P. J. Ramadge, "On the Supremal Controllable Sublanguage of a Given Language", *Systems Control Group Report #8312*, University of Toronto, Toronto, Canada, November 1983.