Pergamon

Mathl. Comput. Modelling Vol. 23, No. 11/12, pp. 55–76, 1996
Copyright©1996 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0895-7177/96 $15.00 + 0.00

S0895-7177(96)00064-7

A Logical DES Approach to the Design of Hybrid Control Systems

J. A. STIVER, P. J. ANTSAKLIS AND M. D. LEMMON
Department of Electrical Engineering
University of Notre Dame, Notre Dame, IN 46556, U.S.A.

Abstract—Hybrid control systems, that is, systems which contain both continuous dynamics and discrete event dynamics are studied in this paper. First, a model is introduced that describes the continuous plant and discrete event controller along with an interface which connects them. A Discrete Event System (DES) automaton description is employed to describe the plant together with the interface and it is used to analyze the hybrid control system. Controllability is defined for hybrid control systems, enhancing existing DES control concepts. It is then used to obtain a controller design method for hybrid control systems.

Keywords—Discrete event, Hybrid, Control, Automata, Feedback.

1. INTRODUCTION

Hybrid systems contain two distinct types of systems, systems with continuous dynamics and systems with discrete event dynamics, that interact with each other. Hybrid control systems typically arise when continuous processes interact with, or are supervised by, sequential machines. Since the continuous and discrete dynamics coexist and interact with each other, it is important to develop models that accurately describe the dynamic behavior of such hybrid systems. In this way it is possible to develop control strategies that take fully into consideration the relation and interaction of the continuous and discrete parts of the system. In the past, models for the continuous and discrete event subsystems were developed separately; the control law was then derived in a rather empirical fashion, except in special cases such as the case of digital controllers for linear time-invariant systems. The study of hybrid control systems is essential in designing sequential supervisory controllers for continuous systems, and it is central in designing intelligent control systems with a high degree of autonomy [1,2]. Examples of hybrid control systems are common in practice and are found in such applications as flexible manufacturing, chemical process control, electric power distribution, and computer communication networks. A simple example of a hybrid control system is the heating and cooling system of a typical home. The furnace and air conditioner, along with the heat flow characteristics of the home, form a continuous system which is to be controlled. The thermostat is a simple discrete event system which basically handles the symbols *too hot*, *too cold*, and *normal*. The temperature of the room is translated into these representations in the thermostat and the thermostat's response is translated back to electrical currents which control the furnace, air conditioner, blower, etc. That is, the thermostat (controller) only sees a quantized, or symbolic, version of the output (temperature) of the home (plant). It generates a finite number of piecewise constant signals that translate into commands to turn on or off the furnace or the air conditioner.

This work was partially supported by NSF Grants MSS-9216559 and IRI91-09298.

The hybrid control systems of interest here consist of a continuous (state, variable) system to be controlled, also called the plant, and a discrete event controller connected to the plant via an interface. It is generally assumed that the dynamic behavior of the plant is governed by a set of known nonlinear ordinary differential or difference equations, however our development below is based on the state trajectories of the plant rather than the particular mechanism in force that generated those trajectories; that is, our results apply, under certain assumptions, to systems where a differential or difference equation description is not known or may not even exist. In the approach described below, the plant contains all continuous subsystems of the hybrid control system, such as any conventional continuous controller that may have been developed, a clock if time and synchronous operations are to be modeled, etc. The controller is an event driven, asynchronous discrete event system (DES), described here by a finite state automaton. The hybrid control system also contains an interface that provides the means for communication between the continuous plant and the DES controller; see Figure 1. The interface receives information from the plant in the form of measurements of a continuous variable, such as the continuous state, and issues a sequence of symbols to the DES controller. It also receives a sequence of control symbols from the controller and issues (piecewise) continuous input commands to the plant.

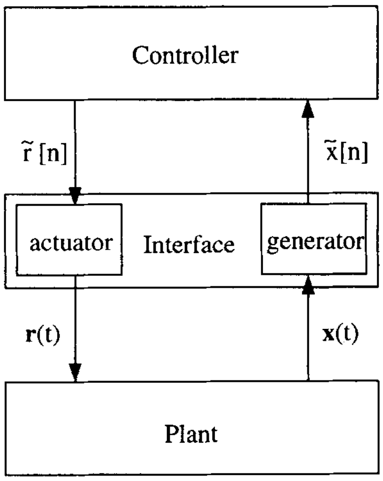

Figure 1. Hybrid control system.

The interface plays a key role in determining the dynamic behavior of the hybrid control system. Understanding how the interface affects the properties of the hybrid system is one of the fundamental issues in the theory of hybrid control systems. In our model, the interface has been chosen to be simply a partitioning of the state space and this is done without loss of generality. If memory is necessary to derive an effective control law, it is included in the DES controller and not in the interface. Also, the piecewise continuous command signal issued by the interface is simply a staircase signal, not unlike the output of a zero-order hold in a digital control system. Including an appropriate continuous system at (the input of) the plant, signals such as ramps, sinusoids, etc., can be generated if desired. The simple interface used in our model allows analysis of the hybrid control system, and in particular development of properties such as controllability [3], stability [4], and determinism, in addition to synthesis results and the development of controller design methodologies [5,6]. The simplicity of our interface with the resulting benefits in identifying central issues and concepts in hybrid control systems is perhaps the main characteristic of our approach. It is also what has been distinguishing our approach from other approaches (with more complex interfaces, or with restrictions on the class of systems studied) since early versions of our model first appeared in 1991 [7,8].

In general, the design of the interface depends not only on the plant to be controlled, but also on the control policies available, as well as on the control goals to be attained. Certain control goals may require, for example, detailed feedback information, while for others coarser quantization levels of the signals may be sufficient. The former case corresponds to finer partitioning of the feedback signal space, while the latter corresponds to coarser partitioning. The fact that different control goals may require different types of information about the plant is not surprising, as it is rather well known that to stabilize a system, for example, requires less detailed information about the system's dynamic behavior than to do tracking. Note that in general, the fewer the distinct regions in the partitioned signal space, the simpler (fewer states) the resulting DES plant model will be, and this will result in a simpler DES controller design. Since the systems to be controlled via hybrid controllers are typically complex, it is important to make every effort to use only the necessary information to attain the control goals, as this leads to simpler interfaces that issue only the necessary number of distinct symbols, and to simpler DES plant models and controllers. The question of systematically determining the minimum amount of information needed from the plant in order to achieve particular control goals using a finite number of distinct control policies is an important and still open question; our work in [6] partially resolves this question.

The hybrid control system model presented in this paper is close to the model of [9], which was also developed for control purposes (in particular control design; our model is for analysis and design). Other models include [10]; see also the earlier references therein. The models in [11,12] are more general but they are developed primarily for simulation purposes. The model of [13] is developed for control, but the interface is rather complex. Other approaches include [14–21]. The paper by Branicky *et al.* [22] presents a rather detailed comparison of some of the models. Early work also included [23,24]. For recent developments in hybrid systems research, see [25].

In this paper, a model is presented in Section 2 that is developed for hybrid control system analysis and design. This model is general and it is characterized by a simple interface. This allows the development of a DES plant model in Section 3, which includes the continuous plant together with the interface. In particular, DES plant properties such as determinism and observability are introduced and discussed, as well as the notion of controllability, which is an extension of the corresponding controllability notion in logical DESs. Given control goals, such as forbidden states, a design methodology is presented that leads to a DES controller. This design method is an extension of the corresponding method in logical DES. An alternative approach in modeling is to consider a discrete-time plant instead of a continuous-time plant [7,26]. This approach leads to simpler mathematical analysis, as it avoids the issue of continuous trajectories crossing hypersurfaces—this approach is outlined in Section 5.

Note that early versions of the results discussed in this paper have appeared in [8,27–30].

2. HYBRID CONTROL SYSTEM MODELING

A hybrid control system can be divided into three parts, the plant, interface, and controller, as shown in Figure 1. In this model, the plant represents the continuous-time components of the system, while the controller represents the discrete-event portions. The interface is the necessary mechanism by which the former two communicate. The models used for each of these three parts, as well as the way they interact are now described.

2.1. Plant

The plant is the part of the model which represents the entire continuous portion of the hybrid control system. The distinguishing feature of the plant is that it has a continuous state space, where the states take on values that are real numbers, and evolve with time according to a set of differential equations. Motivated by tradition, this part of the model is referred to as the plant but since it contains all the continuous dynamics, it can also contain a conventional, i.e., continuous-time, controller.

The plant is a nonlinear, time-invariant system represented by a set of ordinary differential equations.

$$\dot{\mathbf{x}}(t) = f\left(\mathbf{x}(t), \mathbf{r}(t)\right), \tag{1}$$

where $\mathbf{x}(t) \in \mathbf{X}$ and $\mathbf{r}(t) \in \mathbf{R}$ are the state and input vectors, respectively, and $\mathbf{X} \subset \Re^n$, $\mathbf{R} \subset \Re^m$, with $t \in (a, b)$ some time interval. For any $\mathbf{r}(t) \in \mathbf{R}$, the function $f : \mathbf{X} \times \mathbf{R} \to \mathbf{X}$ is continuous in $\mathbf{x}$ and meets the conditions for existence and uniqueness of solutions for initial states, $\mathbf{x}_0 \in \mathbf{X}$. Note that the plant input and state are continuous-time vector valued signals. Boldface letters are used here to denote vectors and vector valued signals.

2.2. Controller

The controller is a discrete event system which is modeled as a deterministic automaton. This automaton is specified by a quintuple, $(\tilde{S}, \tilde{X}, \tilde{R}, \delta, \phi)$, where $\tilde{S}$ is the set of states, $\tilde{X}$ is the set of *plant symbols*, $\tilde{R}$ is the set of *controller symbols*, $\delta : \tilde{S} \times \tilde{X} \to \tilde{S}$ is the state transition function, and $\phi : \tilde{S} \to \tilde{R}$ is the output function. The symbols in set $\tilde{R}$ are called controller symbols because they are generated by the controller. Likewise, the symbols in set $\tilde{X}$ are called plant symbols and are generated based on events in the plant. The action of the controller is described by the equations

$$\tilde{s}[n] = \delta\left(\tilde{s}[n-1], \tilde{x}[n]\right), \tag{2}$$

$$\tilde{r}[n] = \phi\left(\tilde{s}[n]\right), \tag{3}$$

where $\tilde{s}[n] \in \tilde{S}$, $\tilde{x}[n] \in \tilde{X}$, and $\tilde{r}[n] \in \tilde{R}$. The index n is analogous to a time index in that it specifies the order of the symbols in the sequence. The input and output signals associated with the controller are sequences of symbols.

Tildes are used to indicate a symbol valued set or sequence. For example, $\tilde{X}$ is the set of plant symbols and $\tilde{x}[n]$ is the n^{th} symbol of a sequence of plant symbols. Subscripts are also used, e.g., $\tilde{x}_i$ which denotes the i^{th} member of the symbol alphabet $\tilde{X}$.

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 way that this conversion is accomplished determines, to a great extent, the nature of the overall hybrid control system. The interface consists of two simple subsystems, the generator and actuator.

The generator issues symbols to the controller and plays the role of a quantizer of the signals analogous to an A/D converter (sampler) in a digital control system. The actuator injects the appropriate control signal into the plant and it is analogous to a D/A converter (typically a zero-order hold) in a digital control system. The generator and the actuator perform, however, more general functions than their counterparts in a typical digital control system.

2.3.1. Plant events and the generator

The *generator* is the subsystem of the interface which converts the continuous-time output (state) of the plant to an asynchronous, symbolic input for the controller. To perform this task, two processes must be in place. First, a triggering mechanism is required which will determine when a plant symbol should be generated. Second, a process to determine which particular plant symbol should be generated is required.

In the generator, the triggering mechanism is based on the idea of *plant events*. A plant event is simply an occurrence in the plant, an idea borrowed from the field of discrete event systems. In the case of hybrid control, a plant event is defined by specifying a hypersurface which separates the plant's state space. The plant event occurs whenever the plant state trajectory crosses this hypersurface. The basis for this definition of a plant event is that an event is considered to be the realization of a specified condition. This condition can be given as an open region of the state space, separated from the remainder of the state space by a hypersurface. If the state crosses the hypersurface into the given open region, the event has occurred. Mathematically, the set of plant events recognized by the generator is determined by a set of smooth functionals, $\{h_i : \Re^n \to \Re, i \in I\}$, defined on the state space of the plant. Each functional must satisfy the condition,

$$\nabla_x h_i(\xi) \neq 0, \qquad \forall \xi \in \mathcal{N}(h_i), \tag{4}$$

which ensures that the null space of the functional, $\mathcal{N}(h_i) = \{\xi \in \Re^n : h_i(\xi) = 0\}$, forms an $n-1$ dimensional smooth hypersurface separating the state space.

Let the sequence of plant events be denoted e, where $e[n] = i$ means that the n^{th} plant event was triggered by crossing the hypersurface defined by h_i. Let the sequence of plant event instants be given by τ_e, where $\tau_e[n]$ is the time of the n^{th} plant event and $\tau_e[0] = 0$. By definition, these sequences satisfy the following conditions.

$$e[n] = i \Rightarrow \begin{cases} h_i(\mathbf{x}(\tau_e[n])) = 0, \\ \exists\ \delta_1 > 0 \ \text{ s.t. } \forall \epsilon, 0 < \epsilon < \delta_1, \ \ \pm h_i(\mathbf{x}(\tau_e[n] + \epsilon)) < 0, \\ \exists\ \delta_2 > 0 \ \text{ s.t. } \forall \epsilon, 0 < \epsilon < \delta_2, \ \ \pm h_i(\mathbf{x}(\tau_e[n] - \delta_2)) > 0, \ \ \pm h_i(\mathbf{x}(\tau_e[n] - \epsilon)) \geq 0, \end{cases} \tag{5}$$

and

$$\forall n, \tau_e[n] < \tau_e[n+1] \vee (\tau_e[n] = \tau_e[n+1] \wedge e[n] < e[n+1]). \tag{6}$$

The first group, equation (5), contains three conditions:

(i) at the time of the plant event, the plant state lies on the triggering hypersurface,
(ii) immediately after the event, the plant state lies on the negative (positive) side of the triggering hypersurface, and
(iii) prior to reaching the triggering hypersurface, the plant state lied on the negative (positive) side.

The fourth condition, equation (6), concerns the ordering of the sequences. It requires that plant events be ordered chronologically and simultaneous plant events be ordered according to their number, that is, the value of i.

An alternative, and perhaps simpler way of expressing the conditions of (5) is by the condition $h_i(\mathbf{x}(t)) = 0$, $\frac{d}{dt}h_i(\mathbf{x}(t)) \neq 0$. In this case, the assumption is made that the derivative is nonzero; that is, $\frac{d}{dt}h_i(\mathbf{x}(t)) \neq 0$ at the crossing. Note however, that these conditions do not take into account the case where the crossing occurs exactly at an inflection point. When $\frac{d}{dt}h_i(\mathbf{x}(t)) = 0$, one must use (5).

A plant event will only cause a plant symbol to be generated if the hypersurface is crossed in the negative direction. The reason for this is that in many applications, sensors only detect when a threshold is crossed in one direction, e.g., a thermostat. When the hypersurface is crossed in the opposite direction the event is silent, and for convenience, assume that a null symbol, ϵ, is generated. At each time in the sequence $\tau_e[n]$, a plant symbol is generated according to the function $\alpha_i : \mathcal{N}(h_i) \to \tilde{X}$. The sequence of plant symbols can now be defined as

$$\tilde{x}[n] = \begin{cases} \alpha_i\left(\mathbf{x}(\tau_e[n])\right), & \text{nonsilent event}, \\ \epsilon, & \text{silent event}, \end{cases} \tag{7}$$

where i identifies the hypersurface which was crossed. Alternatively, one could select the interface to generate information bearing symbols when crossed in either direction. This is not as flexible and leads to some difficulties in the analysis of the hybrid control system. Our definition in (7) is more general and can easily model this case. See the thermostat example in 3.3 below.

2.3.2. The actuator

The actuator converts the sequence of controller symbols to a plant input signal, using the function $\gamma : \tilde{R} \to \Re^m$, as follows.

$$\mathbf{r}(t) = \sum_{n=0}^{\infty} \gamma\left(\tilde{r}[n]\right) I\left(t, \tau_c[n], \tau_c[n+1]\right), \tag{8}$$

where $I(t, \tau_1, \tau_2)$ is a characteristic function taking on the value of unity over the time interval $[\tau_1, \tau_2)$ and zero elsewhere. The time of the n^{th} control symbol, $\tau_c[n]$, which is based on the sequence of plant symbol instants, defined in (5), is according to

$$\tau_c[n] = \tau_e[n] + \tau_d, \tag{9}$$

where τ_d is the total delay associated with the interface and controller. Following the occurrence of a plant event, it takes a time of τ_d for a new control policy to be used by the plant. It will be assumed that $\tau_e[n] < \tau_c[n] < \tau_e[n+1]$.

The plant input, $\mathbf{r}(t)$, 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.

Remark. The model presented uses the plant state, $\mathbf{x}(t)$, as the feedback signal. When the state is not available for measurement, a plant output signal, $\mathbf{y}(t)$, can also be used. This case is not treated in this paper.

Remark. In the interface a delay, τ_d was introduced. The presence of the delay is necessary for two reasons. First, from a practical point of view, the generator will not be able to detect an event until after the state has actually crossed the hypersurface. Second, if a chattering control strategy is used, the delay partially determines the chattering period. It is, of course, possible for two plant events to occur within the period of a single delay. In such a case, each event will be acted upon, in turn, τ_d after it occurs. In this way, the delay can pose a problem for the controller, but it is unavoidable as real systems cannot react instantaneously.

3. DES PLANT MODEL

In a hybrid control system, the plant taken together with the actuator and generator, behaves like a discrete event system; it accepts symbolic inputs via the actuator and produces symbolic outputs via the generator. This situation is somewhat analogous to the way a continuous-time plant, equipped with a zero order hold and a sampler, "looks" like a discrete-time plant. In a hybrid control system, the DES which models the plant, actuator, and generator is called the *DES plant model*. From the DES controller's point of view, it is the DES plant model which is controlled.

It must be pointed out that the DES plant model is an approximation of the actual plant-actuator-generator combination. Since the DES plant model has a discrete state space, it cannot model the exact behavior of a system which has a continuous state space. The exact relationship between the two will be discussed after the description of the DES plant model.

3.1. Extraction of the DES Plant Model

The DES plant model is a nondeterministic automaton, represented mathematically by a quintuple, $(\tilde{P}, \tilde{X}, \tilde{R}, \psi, \lambda)$. $\tilde{P}$ is the set of states, $\tilde{X}$ is the set of plant symbols, and $\tilde{R}$ is the set of control symbols. $\psi : \tilde{P} \times \tilde{R} \to 2^{\tilde{P}}$ is the state transition function; for a given DES plant state and a given control symbol, it specifies which DES plant states are enabled. The output function, $\lambda : \tilde{P} \times \tilde{P} \to 2^{\tilde{X}}$, maps the previous and current state to a set of plant symbols.

The set of DES plant model states, $\tilde{P}$, is based upon the set of hypersurfaces realized in the generator. Each open region in the state space of the plant, bounded by hypersurfaces, is associated with a state of the DES plant. Whenever a plant event occurs, there is a state transition in the DES plant. Stating this more rigorously, an equivalence relation, $\equiv_p$, can be defined on the set $\{\xi \in \Re^n : h_i(\xi) \neq 0, i \in I\}$ as follows:

$$\xi_1 \equiv_p \xi_2 \iff h_i(\xi_1)h_i(\xi_2) > 0, \qquad \forall i \in I. \tag{10}$$

Each of the equivalence classes of this relation is associated with a unique DES plant state. Thus it is convenient to index the set of states, $\tilde{P}$, with a binary vector, $b \in \{0,1\}^I$, such that b_i is the i^{th} element of b and $\tilde{p}_b$ is associated with the set $\{\xi \in \Re^n : b_i = 1 \Leftrightarrow h_i(\xi) < 0\}$. The equivalence relation is not defined for states which lie on the hypersurfaces. When the continuous state touches a hypersurface, the DES plant model remains in its previous state until the hypersurface is crossed.

Formally, the set of DES plant states is defined as a set of equivalence classes on the state space of the plant.

DEFINITION 1. *The set of DES plant states, $\tilde{P}$, is defined as follows:*

$$\tilde{P} = \{\xi \in \Re^n : h_i(\xi) \neq 0, i \in I\} / \equiv_p . \tag{11}$$

So, for example, the state $\tilde{p}_b$ is defined as

$$\tilde{p}_b = \{\xi \in \Re^n : b_i = 0 \Rightarrow h_i(\xi) > 0 \text{ and } b_i = 1 \Rightarrow h_i(\xi) < 0\} \tag{12}$$

Now the DES plant state can be defined for a system.

DEFINITION 2. *The DES plant state, $\tilde{p}[n]$, is defined as follows.*

$$\tilde{p}[n] = \tilde{p}_b, \tag{13}$$

where

$$\lim_{\epsilon \to 0^+} \mathbf{x}(\tau_e[n] + \epsilon) \in \tilde{p}_b. \tag{14}$$

So the state of the DES corresponds to the most recently entered region of the plant state space. The limit must be used because at exactly $\tau_e[n]$ the continuous state will be on a boundary.

The reason for this definition of state for the DES plant model is that it represents how much can be known about the system by observing the plant symbols without actually calculating the trajectories. So after a plant symbol is generated nothing can be ascertained beyond the resulting region.

Now we are in a position to determine the state transition function, ψ, and the output function, λ. First we define adjacency for DES plant states.

DEFINITION 3. *Two DES plant states, $\tilde{p}_b$, $\tilde{p}_c$, are adjacent at $(i \in I, \xi \in \mathcal{N}(h_i))$, if for all $j \in I$,*

$$\mathcal{N}(h_j) = \mathcal{N}(h_i) \Rightarrow b_i \neq c_i,$$
$$\mathcal{N}(h_j) \neq \mathcal{N}(h_i) \Rightarrow b_i = c_i,$$
$$\xi \in \overline{\tilde{p}_b} \cap \overline{\tilde{p}_c},$$

where $\overline{\tilde{p}_b}$ represents the closure of $\tilde{p}_b$.

When two DES plant states are adjacent at (i, ξ), it means that the regions corresponding to these states are separated by the hypersurface $\mathcal{N}(h_i)$, and the point ξ lies on this hypersurface on the boundary of both regions. Thus, ξ identifies a possible transition point between the regions.

The following proposition states that for a given DES plant state, $\tilde{p}_b$, and control symbol, $\tilde{r}_k$, a possible successor state is $\tilde{p}_c$ if the stated conditions are met. Assume that the hypersurfaces defined by h_i do not have inflection points; see comments following (5).

PROPOSITION 1. *Given a hybrid control system described by (1)–(9), with f and h_i smooth, if $\exists\, i \in I$ and $\xi \in \mathcal{N}(h_i)$ such that the following conditions are satisfied:*

- *$\tilde{p}_b$ and $\tilde{p}_c$ are adjacent at (i, ξ),*
- *$b_i = 0 \Rightarrow \nabla_x h_i(\xi) \cdot f(\xi, \gamma(\tilde{r}_k)) < 0$, and*
- *$b_i = 1 \Rightarrow \nabla_x h_i(\xi) \cdot f(\xi, \gamma(\tilde{r}_k)) > 0$,*

then $\tilde{p}_c \in \psi(\tilde{p}_b, \tilde{r}_k)$.

PROOF. Assume there exists ($i \in I$, $\xi \in \mathcal{N}(h_i)$) which satisfy the proposition for some $\tilde{p}_b$, $\tilde{p}_c$, and $\tilde{r}_k$. Consider a trajectory, $\mathbf{x}$, such that at time t, $\mathbf{x}(t) = \xi$ and $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \gamma(\tilde{r}_k))$. By the adjacency assumption, we know that $\mathbf{x}(t) \in \overline{\tilde{p}_b}$, and along with the other two conditions of the proposition, we know that $\mathbf{x}(t^-) \in \tilde{p}_b$. The adjacency assumption also means that $\mathbf{x}(t) \in \overline{\tilde{p}_c}$, and along with the other two conditions of the proposition, we know that $\mathbf{x}(t^+) \in \tilde{p}_c$. So therefore, there is a state transition at time t from $\tilde{p}_b$ to $\tilde{p}_c$ with the control symbol $\tilde{r}_k$. ∎

The usefulness of this proposition is that it allows the extraction of a DES automaton model of the continuous plant and interface. Note that in certain cases this is a rather straightforward task. For instance, it is known that if a particular region boundary is only crossed in one direction under a given command, then the conditions of the proposition need only be tested at a single point on the boundary. This condition is true for the double integrator example, which follows. In general, this may not be the case, but one can restrict the area of interest to an operating region of the plant state space, thus reducing the computations required. The general conditions under which a DES plant model can be easily extracted have yet to be derived. A special case is the case of timed automata, where the continuous plant is of the form $\dot{\mathbf{x}}$ constant, treated in [15] among others.

The output function, λ, can be found by a similar procedure described in the next proposition.

PROPOSITION 2. *Given a hybrid control system described by (1)–(9), with f and h_i smooth, $\tilde{x}_\ell \in \lambda(\tilde{p}_b, \tilde{p}_c)$ if and only if $\exists\,(i, \xi)$, which satisfies Proposition 1 for some $\tilde{r}_k$, and such that $\alpha_i(\xi) = \tilde{x}_\ell$.*

PROOF. This proposition follows immediately from the definition of the generator. In particular, the plant symbol generated by a plant event is defined as $\alpha_i(\xi)$ where ξ is the continuous-time plant state at the time of the plant event. ∎

3.2. The DES Plant Model as an Approximation

As stated above, the DES plant model is an approximation of the actual hybrid system. Specifically, the state of the DES plant model is an approximation of the state of the continuous plant. As a result, the future behavior cannot be determined uniquely, in general, from knowledge of the DES plant state. The approach taken here is to incorporate all the possible future behaviors into the DES plant model. From a control point of view this means that if undesirable behaviors can be eliminated from the DES plant (through appropriate control policies) then these behaviors can likewise be eliminated from the actual system. On the other hand, just because a control policy permits a given behavior in the DES plant, is no guarantee that that behavior will occur in the actual system; this phenomenon is due to the nondeterminism in the DES plant model. The issues of determinism and nondeterminism are further discussed at the end of this section.

3.3. Example—Furnace and Thermostat

Consider a system made up of a thermostat, room, and heater. If the thermostat is set at 70°F, and assuming it is colder outside, the system behaves as follows. If the room temperature falls below 70 degrees the heater starts and remains on until the room temperature exceeds 75 degrees at which point the heater shuts off. For simplicity, we will assume that when the heater is on it produces heat at a constant rate.

The plant in this hybrid control system is made up of the heater and room, and it can be modeled with the following differential equation:

$$\dot{\mathbf{x}}(t) = .0025\,(T_O - \mathbf{x}(t)) + .02\mathbf{r}(t). \tag{15}$$

Here, $\mathbf{x}(t)$ is the room temperature, T_O is the outside temperature, and $\mathbf{r}(t)$ is the voltage into the heater. Temperatures are in degrees Fahrenheit and time is in minutes.

The generator and controller are found in the thermostat. The generator partitions the state space with two hypersurfaces.

$$h_1(\mathbf{x}) = \mathbf{x} - 70, \tag{16}$$
$$h_2(\mathbf{x}) = -\mathbf{x} + 75. \tag{17}$$

The first hypersurface detects when the temperature falls below 70°F and the second detects when the temperature rises above 75°F. The events are represented symbolically to the controller

$$\alpha_1(\xi) = \text{cold}, \tag{18}$$
$$\alpha_2(\xi) = \text{hot}. \tag{19}$$

It is common to see bimetallic strips performing this function in an actual thermostat, where the band is physically connected to the controller. The controller has two states (typically it is just a switch in the thermostat) as illustrated in Figure 2. The output function of the thermostat controller provides two controller symbols, *on* and *off*.

$$\phi(\tilde{s}_1) = \text{on} \qquad \phi(\tilde{s}_2) = \text{off}. \tag{20}$$

$$\begin{tikzcd} s_1 \arrow[r, bend left, "hot"] & s_2 \arrow[l, bend left, "cold"] \end{tikzcd}$$

Figure 2. Controller in thermostat.

Finally the actuator converts the symbolic output of the controller to a continuous input for the plant.

$$\gamma(\text{on}) = 110 \qquad \gamma(\text{off}) = 0. \tag{21}$$

In this case, the plant input is the voltage supply to the heater, 0 or 110 volts. Physically, the symbolic output from the controller could be a low voltage signal, say 0 or 12 volts, or perhaps a pneumatic signal.

The thermostat/heater example has a simple DES plant model which is useful to illustrate how these models work. Figure 3 shows the DES plant model for the heater/thermostat. The convention for labeling the arcs is to list the controller symbols which enable the transition followed by a "/" and then the plant symbols which can be generated by the transition. Notice that two of the transitions are labeled with null symbols, ϵ. This reflects the fact that nothing actually happens in the system at these transitions. When the controller receives a null symbol it remains in the same state and reissues the current controller symbol. This is equivalent to the controller doing nothing, but it serves to keep all the symbolic sequences, $\tilde{s}, \tilde{p}$, etc., in phase with each other.

$$\begin{tikzcd} \tilde{p}_{10} \arrow[r, bend left, "on/\varepsilon"] & \tilde{p}_{00} \arrow[l, bend left, "off/cold"] \arrow[r, bend left, "on/hot"] & \tilde{p}_{01} \arrow[l, bend left, "off/\varepsilon"] \end{tikzcd}$$

Figure 3. DES plant for thermostat/heater.

In Section 4, an approach to DES controller design is presented that is based on the DES plant model derived above. In general the DES plant model is a nondeterministic automaton. In fact, this is the case for all the examples in the next section. Below we briefly discuss the concepts of determinism and quasideterminism.

3.4. Determinism and Quasideterminism

When an automaton is said to be deterministic, it typically means that for a given state and event, there is only one possible subsequent state. The DES plant model has input (controller symbols) and output (plant symbols), either of which can treated as the event. If the plant symbol is considered to be the event then determinism becomes the observability defined above. Here, however, determinism is defined with respect to the input (controller symbol), which is more in keeping with the classical systems definition of determinism. Thus, the DES plant model will be deterministic if the subsequent state can always be determined uniquely from the current state and input.

There is an advantage to having a hybrid control system in which the DES plant model is deterministic. It allows the controller to drive the plant state through any desired sequence of regions provided, of course, that the corresponding state transitions exist in the DES plant model. If the DES plant model is not deterministic, this will not always be possible. This is because even if the desired sequence of transitions exists, the sequence of inputs which achieves it may also permit other sequences.

As mentioned above, determinism is desirable from the standpoint of control. Unfortunately, given a continuous time plant, it may be difficult or even impossible to design an interface that leads to a DES plant model which is deterministic. Fortunately it is not generally necessary to have a deterministic DES plant model in order to control it. We now introduce *quasideterminism*. This term is used here to describe a DES plant for which a subsequent state can be determined uniquely from the current state, the previous input symbol, and the previous output symbol. Note quasideterminism is similar to the determinism defined above, except that with quasideterminism the output symbol is included.

It is possible to design an interface that yields a quasideterministic DES plant model by carefully designing the plant symbol functions, α_i. This can be a computationally demanding task as it requires calculating trajectories backwards to find out where they enter a given region. The plant symbol functions are then designed to produce different symbols if the triggering trajectories will eventually flow to different regions under the same control policy. More details on quasideterminism can be found in [4].

4. SUPERVISORY CONTROL VIA DES PLANT MODELS

In this section, we use the language generated by the DES plant to examine the controllability of the hybrid control system. This work builds upon the work done by Ramadge and Wonham on the controllability of discrete event systems in a logical framework [31–35]. Here we adapt several of those results and apply them to the DES plant model obtained from a hybrid control system.

Before existing techniques, developed in the logical DES framework can be extended, certain differences must be dealt with. The Ramadge-Wonham model (RWM) consists of two interacting DES's called here the *RWM generator* and *RWM supervisor*. The RWM generator is analogous to the DES plant and the RWM supervisor is analogous to the DES controller. The RWM generator shares its name with the generator found in the hybrid control system interface, but the two should not be confused. In the RWM, the plant symbols are usually referred to as "events," but we will continue to call them plant symbols to avoid confusion. The plant symbols in the RWM are divided into two sets, those which are controllable and those which are uncontrollable: $\tilde{X} = \tilde{X}_c \cup \tilde{X}_u$. A plant symbol being controllable means that the supervisor can prevent it from being issued by the RWM generator. When the supervisor prevents a controllable plant symbol from being issued, the plant symbol is said to be *disabled*. The plant symbols in $\tilde{X}_c$ can be individually disabled, at any time and in any combination, by a command from the RWM supervisor, while the plant symbols in $\tilde{X}_u$ can never be disabled. This is in contrast to our DES plant where each command (controller symbol) from the DES controller disables a particular subset of $\tilde{X}$ determined by the complement of the set given by the transition function, ψ. Furthermore, this set of disabled plant symbols depends not only on the controller symbol but also the present state of the DES plant. In addition, there is no guarantee that any arbitrary subset of $\tilde{X}$ can be disabled while the other plant symbols remain enabled.

The general inability to disable plant symbols individually is what differentiates the DES plant model from the automata of earlier frameworks.

4.1. The DES Plant Language and Observability

The behavior of a DES can be characterized by the set of all finite sequences of symbols which it can generate. This set is referred to as the language of the DES, and is denoted by L. Given the set of all plant symbols, $\tilde{X}$, the alphabet $\tilde{X}^*$ refers to all finite sequences of symbols from the alphabet. The language, L, is a subset of $\tilde{X}^*$. The following defines which strings, $\tilde{x}$, are in the language of a given DES plant model.

DEFINITION 4. *Given a finite sequence of plant symbols, $\tilde{x} : \mathbf{N} \to \tilde{X}$, defined over the set $\mathbf{N} = \{1, \dots, N\}$, then $\tilde{x} \in L$ if there exists $\tilde{p} \in \tilde{P}^*$ and $\tilde{r} \in \tilde{R}^*$, such that the following hold:*

$$\tilde{p}[n+1] \in \psi(\tilde{p}[n], \tilde{r}[n]), \qquad \forall n \in \mathbf{N}, \tag{22}$$

$$\tilde{x}[n] \in \lambda(\tilde{p}[n-1], \tilde{p}[n]), \qquad \forall n \in \mathbf{N}. \tag{23}$$

The language of a DES plant model may or may not provide a useful feedback signal to the controller. For example, suppose there is only one plant symbol and it is associated with every plant event. The controller would not receive much useful information in such a case. On the other hand, if the language of the DES plant model is sufficiently rich that its current state can be ascertained from its initial state and past output then the controller has more detailed information about the current state of the plant necessary to provide an adequate sequence of control policies.

DEFINITION 5. *A DES plant model is observable if the current state can be determined uniquely from the previous state and plant symbol. That is, observability means that $\forall \tilde{p}_b, \tilde{p}_c, \tilde{p}_d \in \tilde{P}$ and $\tilde{x}_\ell \in \tilde{X}$, if*

$$\tilde{x}_\ell \in \lambda(\tilde{p}_b, \tilde{p}_c)$$

and

$$\tilde{x}_\ell \in \lambda(\tilde{p}_b, \tilde{p}_d),$$

then

$$\tilde{p}_c = \tilde{p}_d.$$

The following proposition follows immediately from the above definition.

PROPOSITION 3. *If a DES plant model is observable, then for any initial state, $\tilde{p}[0]$, and sequence of plant symbols, $\tilde{x}$, produced by the DES, there exists a unique sequence of DES plant states, $\tilde{p}$, capable of producing the sequence $\tilde{x}$.*

PROOF. The definition of observability can be applied iteratively to prove that the each state of the sequence, $\tilde{p}$, is determined uniquely by the previous state and current plant symbol. ∎

In cases where the DES plant model is observable, the above proposition implies the existence of a mapping, obs : $\tilde{P} \times L \to \tilde{P}^*$, which takes an initial state together with a string from the language and maps them to the corresponding sequence of states. The nth state in the sequence, $\tilde{p}[n]$, can also be written, $\text{obs}(q_0, \tilde{x})[n]$, where $q_0 \in \tilde{P}$ was the initial state.

4.2. Controllability and Supervisor Design

A DES is controlled by having various symbols disabled by the controller based upon the sequence of symbols which the DES has already generated. When a DES is controlled, it will generate a set of symbol sequences which lie in a subset of its language. If we denote this language of the DES under control as L_c then $L_c \subset L$.

It is possible to determine whether a given RWM generator can be controlled to a desired language [31]. That is, whether it is possible to design a controller, such that, the RWM generator will be restricted to some target language K. Such a controller can be designed if K is prefix closed and

$$\bar{K}\tilde{X}_u \cap L \subset \bar{K}, \tag{24}$$

where $\bar{K}$ represents the set of all prefixes of K. A prefix of K is a sequence of symbols, to which another sequence can be concatenated to obtain a sequence found in K. A language is said to be prefix closed if all the prefixes of that language are also in the language.

When (24) is true for a given RWM generator, the desired language K is said to be controllable, and provided K is prefix closed, a controller can be designed which will restrict the generator to the language K. This condition requires that if an uncontrollable symbol occurs after the generator has produced a prefix of K, the resulting string must still be a prefix of K because the uncontrollable symbol cannot be prevented.

Since the DES plant model belongs to a slightly different class of automata than the RWM, we present another definition for controllable language which applies to the DES plant. We assume in this section that we are dealing with observable DES plant models, that all languages are prefix closed, and that q_0 is the initial state.

Definition 6. *A language, K, is controllable with respect to a given DES plant if $\forall \tilde{x} \in K$, there exists $\rho \in \tilde{R}$, such that,*

$$\tilde{x}\lambda(q, \psi(q, \rho)) \subset K, \tag{25}$$

where $q = \mathrm{obs}(q_0, \tilde{x})[N]$.

This definition requires that for every prefix of the desired language, K, there exists a control, ρ, which will enable only symbols which will cause string to remain in K.

Proposition 4. *If the language K is controllable according to (6), then a controller can be designed which will restrict the given DES plant to the language K.*

Proof. Let the controller be given by con : $\tilde{X}^* \to \tilde{R}$ where $\mathrm{con}(\tilde{x}) \in \{\rho \in \tilde{R} : \tilde{x}\lambda(q, \psi(q, \rho)) \subset K$, $q = \mathrm{obs}(q_0, \tilde{x})[N]\}$. $\mathrm{con}(\tilde{x})$ is guaranteed to be nonempty by (25). We can now show by induction that $\tilde{x} \in L_{\mathrm{con}} \Rightarrow \tilde{x} \in K$.

(1) $\forall \tilde{x} \in L_f$, such that $|\tilde{x}| = 0$, we have $\tilde{x} \in K$. This is trivial because the only such $\tilde{x}$ is the null string ϵ and $\epsilon \in K$ because K is prefix closed.
(2) Let ${L_f}^i = \{\tilde{x} : \tilde{x} \in L_f, |\tilde{x}| = i\}$, that is ${L_f}^i$ is the set of all sequences of length i found in L_f. Given ${L_f}^i$, ${L_f}^{i+1} = \{w \in \tilde{X}^* : w = \tilde{x}\lambda(q, \psi(q, \mathrm{con}(\tilde{x})), \tilde{x} \in {L_f}^i\}$. Now with the definition of $\mathrm{con}(\tilde{x})$ and (25) we have ${L_f}^i \subset K \Rightarrow {L_f}^{i+1} \subset K$.

So $\tilde{x} \in L_f \Rightarrow w \in K$. ∎

Since the DES plant can be seen as a generalization of the original RWM, the conditions in (25) reduce to those of (24) under appropriate restrictions; namely that the plant symbols fall into a controllable/uncontrollable dichotomy and a control policy exists to disable any combination of controllable plant symbols. This is indeed the case.

If the desired language is not attainable for a given DES, it may be possible to find a more restricted language which is. If so, the least restricted behavior is desirable. A method for finding this behavior which is referred to as the *supremal controllable sublanguage*, $K^{\uparrow}$, of the desired language is described and provided in [31,34]. The supremal controllable sublanguage is the largest subset of K which can be attained by a controller. $K^{\uparrow}$ can be found via the following iterative procedure:

$$K_0 = K \tag{26}$$

$$K_{i+1} = \left\{ w : w \in \bar{K}, \bar{w}\tilde{X}_u \cap L \subset \overline{K_i} \right\} \tag{27}$$

$$K^{\uparrow} = \lim_{i \to \infty} K_i. \tag{28}$$

Once again, this procedure applies to the RWM. For hybrid control systems, the supremal controllable sublanguage of the DES plant can be found by a similar iterative scheme:

$$K_0 = K \tag{29}$$

$$K_{i+1} = \left\{ w \in K : \forall \tilde{x} \in \bar{w} \; \exists \; \rho \in \tilde{R} \text{ such that } \tilde{x}\lambda\left(q, \psi(q,\rho)\right) \subset K_i \right\} \tag{30}$$

$$K^{\uparrow} = \lim_{i \to \infty} K_i. \tag{31}$$

This result yields the following proposition.

PROPOSITION 5. *For a DES plant and language K, $K^{\uparrow}$ is controllable and contains all controllable sublanguages of K.*

PROOF. From (30) we have

$$K^{\uparrow} = \left\{ w \in K : \forall \tilde{x} \in \bar{w} \; \exists \; \rho \in \tilde{R} \text{ such that } \tilde{x}\lambda\left(q, \psi(q,\rho)\right) \subset K^{\uparrow} \right\}, \tag{32}$$

which implies

$$\tilde{x} \in K^{\uparrow} \Rightarrow \exists \; \rho \in \tilde{R} \text{ such that } \tilde{x}\lambda(q, \psi(q,\rho)) \subset K^{\uparrow}. \tag{33}$$

From (33) it is clear that $K^{\uparrow}$ is controllable. We prove that every prefix closed, controllable subset of K is in $K^{\uparrow}$ by assuming there exists $M \subset K$, such that, M is controllable but $M \not\subset K^{\uparrow}$ and showing this leads to a contradiction:

$$\begin{array}{llll} & \exists M \subset K & \text{s.t. } M \not\subset K^{\uparrow} & \quad (34) \\ \Rightarrow & \exists w \in M & \text{s.t. } w \notin K^{\uparrow} & \quad (35) \\ \Rightarrow & \exists i & \text{s.t. } w \in K_i, \quad w \notin K_{i+1} & \quad (36) \\ \Rightarrow & \exists \tilde{x} \in \bar{w} & \text{s.t. } \forall \rho \in \tilde{R}, \quad \tilde{x}\lambda(q,\psi(q,\rho)) \not\subset K_i & \quad (37) \\ \Rightarrow & \exists w' \in \tilde{x}\lambda(q,\psi(q,\rho)) & \text{s.t. } w' \in M, \quad w' \notin K_i & \quad (38) \\ \Rightarrow & \exists j < i & \text{s.t. } w' \in K_j, \quad w' \notin K_{j+1}. & \quad (39) \end{array}$$

If the sequence is repeated with $i = j$ and $w = w'$ we eventually arrive at the conclusion that $w' \in M$ but $w' \notin K_0$ which violates the assumption that $M \subset K$ and precludes the existence of such an M. ∎

4.3. Example—Double Integrator

The system consists of a double integrator plant which is controlled by a discrete event system. The plant is given by the differential equation,

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{r}(t). \tag{40}$$

The generator recognizes four plant events.

$$h_1(\mathbf{x}) = x_1, \qquad h_2(\mathbf{x}) = -x_1, \tag{41}$$

$$h_3(\mathbf{x}) = x_2, \qquad h_4(\mathbf{x}) = -x_2. \tag{42}$$

These four plant events are generated when the plant state crosses either the x_1 or x_2 axis, in positive or negative direction. Symbols are attached to the plant events as follows:

$$\alpha_1(\mathbf{x}) = \tilde{x}_1, \qquad \alpha_2(\mathbf{x}) = \tilde{x}_1, \tag{43}$$

$$\alpha_3(\mathbf{x}) = \tilde{x}_2, \qquad \alpha_4(\mathbf{x}) = \tilde{x}_2. \tag{44}$$

Notice that the same symbol can be used to label more than one plant event and that α_i does not necessarily have to depend on the state $\mathbf{x}(t)$. In this example the plant symbol only identifies the axis which was crossed. Figure 4 illustrates this.

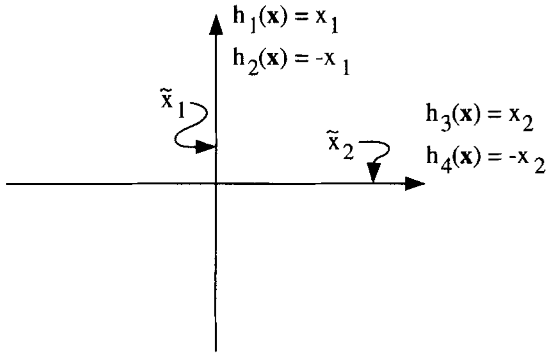

Figure 4. Generator for double integrator example.

There are two controller symbols, $\tilde{R} = \{1, 2\}$, so the actuator provides two possible inputs to the plant.

$$\gamma(\tilde{r}) = \begin{cases} -1 & \text{if } \tilde{r} = 1, \\ 1 & \text{if } \tilde{r} = 2. \end{cases} \tag{45}$$

These inputs were chosen so that the plant can be driven to any state by applying them in the proper sequence.

Using Proposition 1, we can extract the DES plant for this system. It is shown in Figure 5. To illustrate how the DES plant was extracted, start with the DES plant state $\tilde{p}_9$ (i.e., $\tilde{p}_{1001}$) and consider whether $\tilde{p}_5 \in \psi(\tilde{p}_9, \tilde{r}_2)$. $i = 1$ and $\xi = [0\ 1]'$ satisfy the conditions of the proposition, showing that indeed $\tilde{p}_5 \in \psi(\tilde{p}_9, \tilde{r}_2)$. Proceeding in this way we extract the DES plant model. At the same time, Proposition 2 is used to find the plant symbols generated by the transitions. In the sample instance, $\lambda(\tilde{p}_9, \tilde{p}_5)$ there are two possible symbols, $\tilde{x}_1$ and ϵ. By convention the nonsilent symbol takes precedence, so $\{\tilde{x}_1\} = \lambda(\tilde{p}_9, \tilde{p}_5)$.

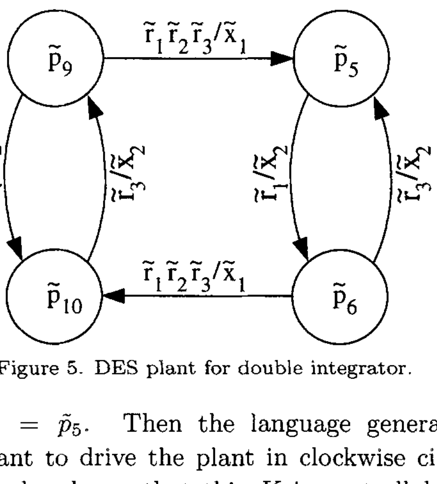
$\tilde{p}_9$
$\tilde{r}_1\tilde{r}_2\tilde{r}_3/\tilde{x}_1$
$\tilde{p}_5$
$\tilde{r}_1/\tilde{x}_2$
$\tilde{r}_3/\tilde{x}_2$
$\tilde{r}_1/\tilde{x}_2$
$\tilde{r}_3/\tilde{x}_2$
$\tilde{p}_{10}$
$\tilde{r}_1\tilde{r}_2\tilde{r}_3/\tilde{x}_1$
$\tilde{p}_6$

Figure 5. DES plant for double integrator.

Let the initial state be $q_0 = \tilde{p}_5$. Then the language generated by this automaton is $L = \overline{(\tilde{x}_2(\tilde{x}_2\tilde{x}_2)^*\tilde{x}_1)^*}$. If we want to drive the plant in clockwise circles, then the desired language is $K = \overline{(\tilde{x}_2\tilde{x}_1)^*}$. It can be shown that this K is controllable because it satisfies (25). Therefore according to Proposition 4, a controller can be designed to achieve the stated control goal.

4.4. Example—A More Complex DES Plant Model

This example has a richer behavior and will illustrate the generation of a supremal controllable sublanguage as well as the design of a controller. We start immediately with the DES plant model shown in Figure 6.

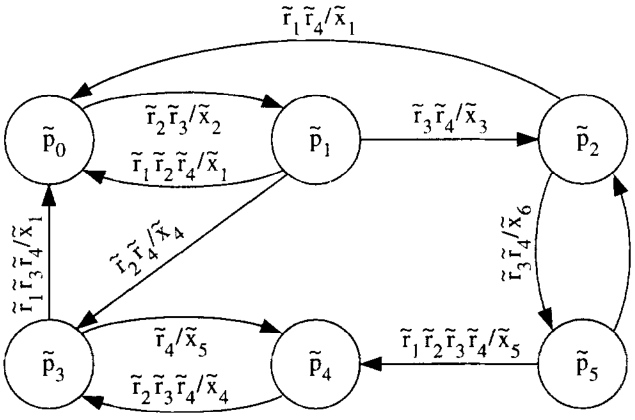
$\tilde{r}_1\tilde{r}_4/\tilde{x}_1$
$\tilde{p}_0$
$\tilde{r}_2\tilde{r}_3/\tilde{x}_2$
$\tilde{r}_1\tilde{r}_2\tilde{r}_4/\tilde{x}_1$
$\tilde{p}_1$
$\tilde{r}_3\tilde{r}_4/\tilde{x}_3$
$\tilde{p}_2$
$\tilde{r}_1\tilde{r}_3\tilde{r}_4/\tilde{x}_1$
$\tilde{r}_2\tilde{r}_4/\tilde{x}_4$
$\tilde{r}_3\tilde{r}_4/\tilde{x}_6$
$\tilde{r}_2/\tilde{x}_3$
$\tilde{p}_3$
$\tilde{r}_4/\tilde{x}_5$
$\tilde{r}_2\tilde{r}_3\tilde{r}_4/\tilde{x}_4$
$\tilde{p}_4$
$\tilde{r}_1\tilde{r}_2\tilde{r}_3\tilde{r}_4/\tilde{x}_5$
$\tilde{p}_5$

Figure 6. DES plant model for Example 2.

The language generated by this DES is $L = \overline{L_m}$ where

$$L_m = \left(\tilde{x}_2\left(\tilde{x}_1 + \tilde{x}_4\left(\tilde{x}_5\tilde{x}_4\right)^*\tilde{x}_1 + \tilde{x}_3\left(\tilde{x}_6\tilde{x}_3\right)^*\left(\tilde{x}_1 + \tilde{x}_6\tilde{x}_5\tilde{x}_4\left(\tilde{x}_5\tilde{x}_4\right)^*\tilde{x}_1\right)\right)\right)^*. \tag{46}$$

Suppose we want to control the DES so that it never enters state $\tilde{p}_4$. We simple remove the transitions to $\tilde{p}_4$ and then compute the resulting language. This desired language is therefore,

$$K = \overline{\left(\tilde{x}_2\left(\tilde{x}_1 + \tilde{x}_4\tilde{x}_1 + \tilde{x}_3\left(\tilde{x}_6\tilde{x}_3\right)^*\tilde{x}_1\right)\right)^*}. \tag{47}$$

In this example, the language K is not controllable. This can be seen by considering the string $\tilde{x}_2\tilde{x}_3\tilde{x}_6 \in K$, for which there exists no $\rho \in \tilde{R}$ which will prevent the DES from deviating from K by generating $\tilde{x}_5$ and entering state $\tilde{p}_4$.

Since K is not controllable, we find the supremal controllable sublanguage of K as defined in (31). The supremal controllable sublanguage is

$$K^{\uparrow} = K_1 = \overline{(\tilde{x}_2(\tilde{x}_1 + \tilde{x}_4\tilde{x}_1 + \tilde{x}_3\tilde{x}_1))^*}. \tag{48}$$

Obtaining a DES controller once the supremal controllable sublanguage has been found is straightforward. The controller is a DES whose language is given by $K^{\uparrow}$ and the output of the controller in each state, $\phi(\tilde{s})$, is the controller symbol which enables only transitions which are found in the controller. The existence of such a controller symbol is guaranteed by the fact that $K^{\uparrow}$ is controllable. For Example 2, the controller is shown in Figure 7 and its output function, ϕ, is as follows:

$$\phi\left(\tilde{s}_1\right) = \tilde{r}_2, \qquad \phi\left(\tilde{s}_2\right) = \tilde{r}_4, \tag{49}$$

$$\phi\left(\tilde{s}_3\right) = \tilde{r}_1, \qquad \phi\left(\tilde{s}_4\right) = \tilde{r}_1. \tag{50}$$

4.5. Example—Distillation Column

This example uses the model of a two product distillation column with a single feed. A complete description of the nonlinear model can be found in [36]. Here a condensed description is given

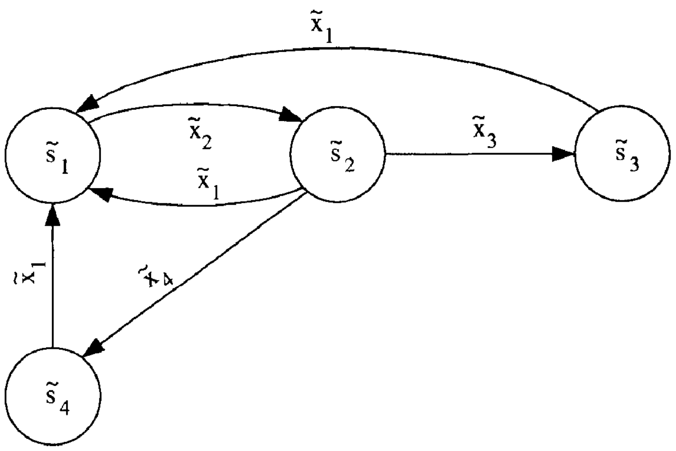

Figure 7. DES controller for Example 2.

to show the source of the DES plant model and provide insight into the physical meaning of the states and events.

Figure 8 shows the distillation column. F represents the feed flow into the column, B is the flow of bottom product out of the column, x_B is the mole fraction of the light compound in the bottom product, D is the flow of distillate out of the column, and y_D is the mole fraction of light compound in the distillate. The boilup flow is denoted by V and the reflux flow by L. All units are in kmol's and minutes. The column can be controlled by setting the feed, boilup, and reflux. In general, the goal is to have a high level of light compound in the distillate ($y_D \to 1$) and a low level of light compound in the bottom product ($x_B \to 0$).

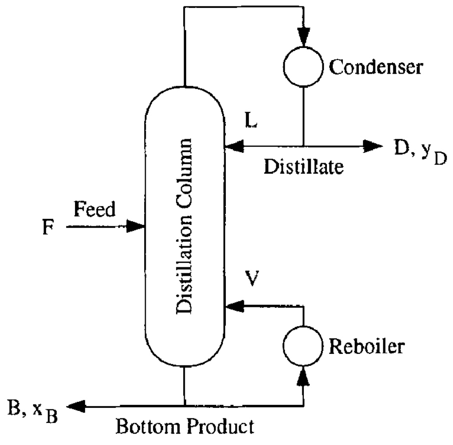

Figure 8. Distillation column.

There are 40 trays stacked vertically in the column. The state consists of the mole fractions of light compound in the liquid of each tray. The states evolve according to the following equations:

$$\begin{aligned}
2\dot{x}_1 &= (L+F_L)x_2 - Vy_1 - Bx_1,\\
2\dot{x}_i &= (L+F_L)x_{i+1} + Vy_i - (L+F_L)x_i - Vy_i,\\
2\dot{x}_{21} &= Lx_{22} + Vy_{20} - (L+F_L)x_{21} - Vy_{21} + F_Lx_F,\\
2\dot{x}_{22} &= Lx_{23} + Vy_{21} - Lx_{22} - (V+F_V)y_{22} + F_Vy_F,\\
2\dot{x}_j &= Lx_{j+1} + (V+F_V)y_j - Lx_j - (V+F_V)y_j,\\
2\dot{x}_{41} &= (V+F_V)y_{40} - Lx_{41} - Dx_{41},
\end{aligned}$$

where $2 \le i \le 20$ and $23 \le j \le 40$. Trays 21 and 22 are special because they are below and above the feed location. Tray 41 is actually the condenser. The quantities y_i are the mole fractions of light compound in the vapor, given by

$$y_i = \frac{\alpha x_i}{1 + (\alpha - 1)x_i},$$

where $\alpha = 1.5$ is relative volatility. Other quantities of interest are

$$\begin{aligned} F_L &= .6F, \\ F_V &= F - F_L, \\ x_F &= \frac{.5F - F_V y_F}{F_L}, \end{aligned}$$

and the outputs are

$$\begin{aligned} D &= V + F_V - L, \\ B &= L + F_L - V, \\ y_D &= y_{41}, \\ x_B &= x_1. \end{aligned}$$

To obtain a hybrid control system, appropriate control policies and plant symbols must be chosen. Their selection is based on our knowledge of the control goals and the design constraints, and it will determine the interface. Let the control policies be

$$\mathbf{r}(t) = \begin{bmatrix} L \\ V \\ F \end{bmatrix} \in \left\{ \begin{bmatrix} 9.5 \\ 10 \\ 1 \end{bmatrix}, \begin{bmatrix} 10 \\ 10 \\ 0.1 \end{bmatrix}, \begin{bmatrix} 9.5 \\ 10 \\ 2 \end{bmatrix}, \begin{bmatrix} 10 \\ 10 \\ 2 \end{bmatrix} \right\}.$$

These input values correspond to $\tilde{r}_1, \tilde{r}_2, \tilde{r}_3$, and $\tilde{r}_4$. Next, plant symbols are defined based on events as follows.

$\tilde{x}_1$	$B + D$ falls below 2
$\tilde{x}_2$	$B + D$ exceeds 2
$\tilde{x}_3$	x_B falls below .13
$\tilde{x}_4$	x_B exceeds .13
$\tilde{x}_5$	x_B falls below .12
$\tilde{x}_6$	x_B exceeds .12
$\tilde{x}_7$	x_B falls below .08
$\tilde{x}_8$	x_B exceeds .08
$\tilde{x}_9$	y_D falls below .84
$\tilde{x}_{10}$	y_D exceeds .84
$\tilde{x}_{11}$	y_D falls below .85
$\tilde{x}_{12}$	y_D exceeds .85
$\tilde{x}_{13}$	y_D falls below .95
$\tilde{x}_{14}$	y_D exceeds .95

We would like to keep x_B below .13, y_D above .95, and the feed at 2. These conditions correspond to increased production of high purity products. Simulations reveal that given the available controls and events this is not possible, that is, even if the initial state is in this region, no available control policy will cause it to remain there. It is possible to drive the system close to this point, however. Specifically, our control goal shall be two-fold: first, to drive the system near the ideal point, and second, to avoid having a high feed rate (2 kmol/min) when the system is not near the ideal point.

The distillation column is an example of a rather complex hybrid system. The generator was designed to recognize 14 different plant events. This leads to 32 distinct regions in the state space, and therefore, there are 32 DES plant states. Figure 9 shows the DES plant model. The two states labeled 'G' correspond to the desired operating regions of the system. This DES plant model was extracted by automating the testing process and implementing it on a computer.

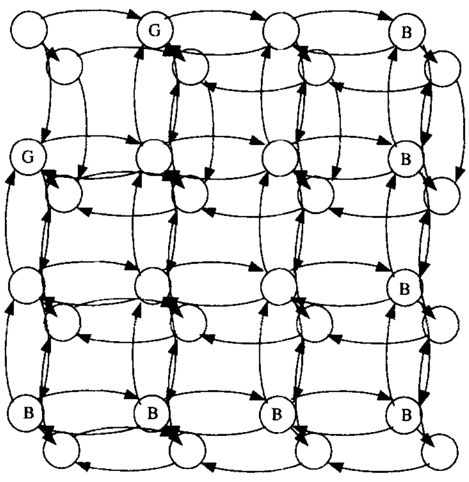

Figure 9. DES plant for distillation column.

A controller was obtained by automating the procedure for finding the supremal controllable sublanguage. The controller is shown in Figure 10. This controller drives the plant from the initial state to a loop containing the two good states. Notice that in this figure, the states of the controller have been labeled with the controller symbol which is generated by that state.

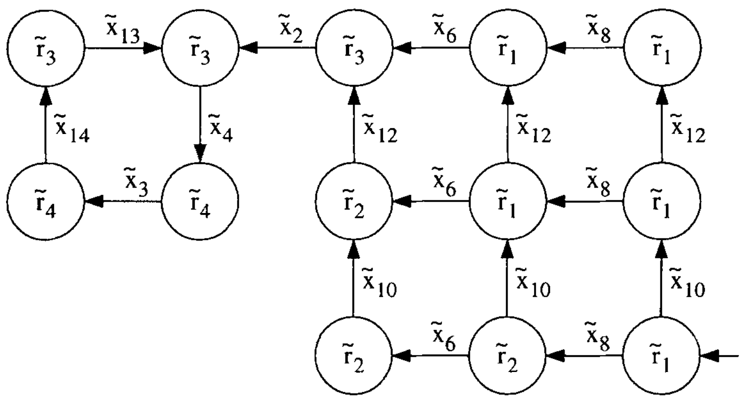

Figure 10. Sample controller for distillation column.

5. HYBRID CONTROL SYSTEM WITH DISCRETE TIME PLANT

The primary model used in this paper features a continuous-time plant. However, hybrid control systems with discrete-time plants can also be handled with few adjustments. This section summarizes the adjustments. First of all, the plant is modeled as a discrete-time system

$$\mathbf{x}(k+1) = F(\mathbf{x}(k), \mathbf{r}(k)), \tag{51}$$

where $\mathbf{x}(k) \in \Re^n$ and $\mathbf{r}(k) \in \Re^m$ are the state and input vectors respectively. $F : \Re^n \times \Re^m \to \Re^n$ is a smooth function.

The generator is defined by a set of smooth functionals, $h_i : \Re^n \to \Re$, and a function, $\alpha : \Re^n \to \tilde{X}$. Notice that unlike the continuous-time case, there is only one α. Since the plant is discrete-time, the generator will not be able to identify the exact moment that a hypersurface is crossed. Rather, it identifies the first sample after a crossing has occurred. So the sequence of plant symbol instants is given by the following equations.

$$\tau_e[0] = 0, \tag{52}$$

$$\tau_e[n] = \min\left\{k > \tau_e[n-1] : \exists i, h_i\left(\mathbf{x}(k)\right) \cdot h_i\left(\mathbf{x}\left(\tau_e[n-1]\right)\right) < 0\right\}. \tag{53}$$

At each plant symbol instant a plant symbol is generated according to α.

$$\tilde{x}[n] = \alpha\left(\mathbf{x}\left(\tau_e[n]\right)\right). \tag{54}$$

Again, not all plant events have to be represented by a plant symbol. Such events are silent, accompanied by the null symbol, ϵ.

The actuator, $\gamma : \tilde{R} \to \Re^m$, converts a sequence of controller symbols to a plant input as follows:

$$\mathbf{r}(k) = \sum_{n=0}^{\infty} \gamma\left(\tilde{r}[n]\right) I\left(k, \tau_c[n], \tau_c[n+1]\right), \tag{55}$$

where $I(\cdot)$ is the following indicator function:

$$I\left(k, \tau_c[n], \tau_c[n+1]\right) = \begin{cases} 1, & \text{if } \tau_c[n] \leq k < \tau_c[n+1], \\ 0, & \text{if otherwise,} \end{cases} \tag{56}$$

and $\tau_c[n]$ is the sequence of control instants. Unlike the continuous-time case, here $\tau_c[n]$ is an integer. This sequence is based on the sequence of event instants, defined in (53), according to

$$\tau_c[n] = \tau_e[n] + \tau_d, \tag{57}$$

where τ_d is the delay associated with the controller and is generally equal to one sampling period.

REMARK. In this case, the continuous state of the plant is observed only at discrete instants of time. The state of the DES plant model is determined by the region containing the continuous state at that particular time. So one does not have to deal with crossings of hypersurfaces by continuous trajectories as was the case previously. However, in general, it is difficult to extract a DES plant model in this case exactly because the crossings of the hypersurfaces are not detected. Under certain periodicity conditions [37], a deterministic DES plant model can be shown to exist.

5.1. Example—The Double Integrator

The double integrator modeled as a discrete-time plant is

$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \mathbf{r}(k). \tag{58}$$

The events are formed by the following two hypersurfaces

$$h_1(\xi) = \xi_1, \tag{59}$$

$$h_2(\xi) = \xi_2, \tag{60}$$

where $\xi = [\xi_1, \xi_2]$. The events generate plant symbols according to α

$$\alpha(\xi) = \begin{cases} \tilde{x}_1, & \text{if } \xi_1 > 0, \quad \xi_2 > 0, \\ \tilde{x}_2, & \text{if } \xi_1 < 0, \quad \xi_2 > 0, \\ \tilde{x}_3, & \text{if } \xi_1 < 0, \quad \xi_2 < 0, \\ \tilde{x}_4, & \text{if } \xi_1 > 0, \quad \xi_2 < 0. \end{cases} \tag{61}$$

Note that α effectively identifies which of the four quadrants is being entered.

The actuator provides three possible inputs to the plant.

$$\gamma(\tilde{r}) = \begin{cases} -1, & \text{if } \tilde{r} = \tilde{r}_1, \\ 0, & \text{if } \tilde{r} = \tilde{r}_2, \\ 1, & \text{if } \tilde{r} = \tilde{r}_3. \end{cases} \tag{62}$$

6. CONCLUDING REMARKS

The contributions of this paper include and are based on a formal model for hybrid control systems that uses a simple interface. The model is simple enough to allow analysis and general enough to include a wide variety of hybrid systems. Conditions are given which allow the extraction of a DES model from the plant and interface. A method for DES controller design is presented.

The results of this paper were used to design a DES controller for a hybrid control system, when the plant and control goals are given. There are several important problems that this paper has not addressed, the most significant being perhaps the systematic design of the interface. Specifically, the problem of selecting the finite number of inputs to the plant is still largely unsolved. However, there are cases when the possible inputs are dictated by the particular application considered; for example, in a satellite when the control jets can only be turned on or off. The design of the generator has also only been partially solved. In this case as well, large parts of the generator may be fixed because of availability of certain types of sensors, etc. In general, using quasideterminism, it is still up to the designer to make the initial selection of a partition, a selection which has a large impact on the resulting system. In [6], an alternative approach based on the invariants of the continuous system, where both the interface and the DES controller are designed, is presented. Finally, the computational issue, which is very important in hybrid control, was not addressed in this paper.

It is important to note that the core issue in hybrid control systems is the way in which the interface relates the continuous plant to the DES plant. This issue is rather fundamental and quite difficult to resolve. Its solution will depend on the answer to the question of what is the minimum amount of information about the plant that will allow, with the controls available, the accomplishment of the control objectives.

Note that the approach taken in this paper, where a DES plant model is derived to describe the dynamic behavior of the continuous plant together with the interface, is similar to one of the main approaches in digital control (sampled data) systems. There the continuous-time plant is combined with the A/D and D/A converters, typically a sampler and a zero-order hold, to obtain a discrete-time plant model. Then the discrete-time controller is designed using discrete-time control design techniques. Of course attention should be paid to the intersample behavior of the continuous system and the selection of the sampling period. There is an alternative approach to digital control design that may lead to an alternative approach to hybrid controller design. In this approach, the discrete-time controller is an approximation of an existing continuous-time controller; that is, the control design is carried out in the continuous domain and then appropriate approximations (similar to the ones used in digital filter design) are employed to derive the discrete-time controller. This is a convenient design approach, however it does not take full advantage of the discrete nature of the controller, since the discrete controller can be, at best, as good as the continuous controller and behavior such as deadbeat is not attained; furthermore this approach typically requires higher sampling rates than the previous ones.

In the approach presented in this paper, the derivation of a well defined DES model for the plant, which describes the continuous plant together with the interface, was quite challenging. Although the idea of obtaining a DES plant model may appear to be simple and straightforward, especially in view of the choice for the interface, this problem turned out to be very complex. There were a number of challenging issues that were addressed. These included dealing with chattering by introducing a delay, with multiple crossings within the time of a single delay, and crossings at inflection points; also dealing with nondeterministic DES plants and describing the relation of the DES plant model to the actual system. Several of these problems are avoided when the continuous state plant is taken to be a discrete-time system, and this was discussed in Section 5. An important future research direction would be to derive conditions under which the DES plant model has certain desirable properties. Research toward that goal for the DES plant model to be deterministic [1] and stable [4] has already been done.

Finally in this paper, finite automata models were used to represent the DES plant model. This was done because the purpose of this work was to use the tools from the logical theory of DES, with appropriate modifications, to design controllers for hybrid systems. Note that the controller used is also a finite automaton. It is of course possible to use other models to represent the DES plant, such as Petri nets.

REFERENCES

1. P.J. Antsaklis, J.A. Stiver and M.D. Lemmon, Hybrid system modeling and autonomous control systems, In *Hybrid Systems, Lecture Notes in Computer Science*, Volume 736, (Edited by R.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel), pp. 366–392, Springer-Verlag, (1993).
2. P.J. Antsaklis, Defining intelligent control, *IEEE Control Systems Magazine* **14** (3) (June 1994); Report of the Task Force on Intelligent Control .
3. J.A. Stiver and P.J. Antsaklis, On the controllability of hybrid control systems, In *Proceedings of the 32nd Conference on Decision and Control*, San Antonio, TX, pp. 3748–3751, (December 1993).
4. M.D. Lemmon and P.J. Antsaklis, Inductively inferring valid logical models of continuous-state dynamical systems, *Theoretical Computer Science* **138**, 201–210 (1995).
5. J.A. Stiver, P.J. Antsaklis and M.D. Lemmon, A logical DES approach to the design of hybrid systems, Technical Report of the ISIS Group (Interdisciplinary Studies of Intelligent Systems) ISIS-94-011, University of Notre Dame, Notre Dame, IN, (October 1994).
6. J.A. Stiver, P.J. Antsaklis and M.D. Lemmon, Interface design for hybrid control systems, Technical Report of the ISIS Group (Interdisciplinary Studies of Intelligent Systems) ISIS-95-001, University of Notre Dame, Notre Dame, IN, (January 1995).
7. J.A. Stiver, Modeling of hybrid control systems using discrete event system models, Master's Thesis, Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, (May 1991).
8. J.A. Stiver and P.J. Antsaklis, A novel discrete event system approach to modeling and analysis of hybrid control systems, In *Proceedings of the Twenty-Ninth Annual Allerton Conference on Communication, Control, and Computing*, University of Illinois at Urbana-Champaign, Urbana, IL, (October 1991).
9. A. Nerode and W. Kohn, Models for hybrid systems: Automata, topologies, controllability observability, In *Hybrid Systems*, (Edited by R.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel), pp. 317–356, Springer-Verlag, (1993).
10. R. Brockett, Language driven hybrid systems, In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pp. 4210–4214, Lake Buena Vista, FL, (December 1994).
11. L. Tavernini, Differential automata and their discrete simulators, *Nonlinear Analysis, Theory, Methods, and Applications* **11** (6), 665–683 (1987).
12. A. Back, J. Guckenheimer and M. Myers, A dynamic simulation facility for hybrid systems, In *Hybrid Systems, Lecture Notes in Computer Science*, Volume 736, (Edited by R.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel), pp. 255–267, Springer-Verlag, (1993).
13. P. Peleties and R. DeCarlo, Analysis of a hybrid system using symbolic dynamics and petri nets, *Automatica* **30** (9), 1421–1427 (1994).
14. B.P. Zeigler, DEVS representation of dynamical systems: Event based intelligent control, *Proceedings of the IEEE* **77** (1), 72–80 (1989).
15. S. Di Gennaro, C. Horn, S. Kulkarni and P. Ramadge, Reduction of timed hybrid systems, In *Proceedings of the 33rd IEEE Conference on Decision and Control*, Lake Buena Vista, FL, pp. 4215–4220, (December 1994).

16. A. Deshpande and P. Varaiya, Viable control of hybrid systems, Ph.D. Dissertation, Can be located at `ftp:eclair.eecs.berkeley.edu`, (June 1994).
17. M. Tittus and B. Egardt, Control-law synthesis for linear hybrid systems, In *Proceedings of the 33rd IEEE Conference on Decision and Control*, Lake Buena Vista, FL, pp. 961–966, (December 1994).
18. L. Holloway and B. Krogh, Properties of behavioral models for a class of hybrid dynamical systems, In *Proceedings of the 31st Conference on Decision and Control*, Tucson, AZ, pp. 3752–3757, (December 1992).
19. A. Benveniste and P. Le Guernic, Hybrid dynamical systems and the signal language, *IEEE Transactions on Automatic Control* **35** (5), 535–546 (May 1990).
20. R. Grossman and R. Larson, Viewing hybrid systems as products of control systems and automata, In *Proceedings of the 31st Conference on Decision and Control*, Tucson, AZ, pp. 2953–2955, (December 1992).
21. W. Kohn and A. Nerode, Multiple agent autonomous hybrid control systems, In *Proceedings of the 31st Conference on Decision and Control*, Tucson, AZ, pp. 2956–2966, (December 1992).
22. M. Branicky, V. Borkar and K. Mitter, A unified framework for hybrid control, In *Proceedings of the 33rd IEEE Conference on Decision and Control*, Lake Buena Vista, FL, pp. 4228–4234, (December 1994).
23. P. Peleties and R. DeCarlo, A modeling strategy with event structures for hybrid systems, In *Proceedings of the 28th Conference on Decision and Control*, Tampa, FL, pp. 1308–1313, (December 1989).
24. A. Gollu and P. Varaiya, Hybrid dynamical systems, In *Proceedings of the 28th Conference on Decision and Control*, Tampa, FL, pp. 2708–2712, (December 1989).
25. R.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel, Editors, *Hybrid Systems*, Volume 736 of *Lecture Notes in Computer Science*, Springer-Verlag, (1993).
26. C. Chase and P.J. Ramadge, Dynamics of a switched n buffer system, In *Proceedings of the Twenty-Eighth Annual Allerton Conference on Communication, Control, and Computing*, University of Illinois at Urbana-Champaign, pp. 455–464, (October 1991).
27. J.A. Stiver and P.J. Antsaklis, Modeling and analysis of hybrid control systems, In *Proceedings of the 31st Conference on Decision and Control*, Tucson, AZ, pp. 3748–3751, (December 1992).
28. J.A. Stiver and P.J. Antsaklis, State space partitioning for hybrid control systems, In *Proceedings of the American Control Conference*, San Francisco, CA, pp. 2303–2304, (June 1993).
29. P.J. Antsaklis, M.D. Lemmon and 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).
30. P.J. Antsaklis, M.D. Lemmon and J.A. Stiver, Learning to be autonomous: Intelligent supervisory control, Technical Report of the ISIS Group ISIS-93-003, University of Notre Dame, Notre Dame, IN, (April 1993); *Intelligent Control: Theory and Applications*, IEEE Press (to appear) .
31. P.J. Ramadge and W.M. Wonham, Supervisory control of a class of discrete event processes, Systems Control Group Report 8515, University of Toronto, Toronto, Canada, (November 1985).
32. P. Ramadge and W.M. Wonham, Supervisory control of a class of discrete event processes, *SIAM Journal of Control and Optimization* **25** (1), 206–230 (January 1987).
33. P. Ramadge and W.M. Wonham, The control of discrete event systems, *Proceedings of the IEEE* **77** (1), 81–89 (January 1989).
34. W.M. Wonham and P.J. Ramadge, On the supremal controllable sublanguage of a given language, Systems Control Group Report 8312, University of Toronto, Toronto, Canada, (November 1983).
35. W.M. Wonham and P.J. Wonham, On the supremal controllable sublanguage of a given language, *SIAM Journal of Control and Optimization* **25** (3), 637–659 (May 1987).
36. M. Morari and E. Zafiriou, *Robust Process Control*, Prentice Hall, (1989).
37. P.J. Ramadge, On the periodicity of symbolic observations of piecewise smooth discrete-time systems, *IEEE Transactions on Automatic Control* **35** (7), 807–812 (July 1990).