Interface and Controller Design for Hybrid Control Systems

James A. Stiver, Panos J. Antsaklis, and Michael D. Lemmon

Department of Electrical Engineering
University of Notre Dame, Notre Dame, IN 46556

Abstract. The hybrid control systems considered here consist of a continuous-time plant under the control of a discrete event system. Communication between the plant and controller is provided by an interface which can convert signals from the continuous domain of the plant to the discrete, symbolic domain of the controller, and vise-versa. When designing a controller for a hybrid system, the designer may or may not be free to design the interface as well. This paper examines these two cases. First, a methodology is presented for designing a controller when the interface and plant are given. This approach is based on the methodology for controller design in logical discrete event systems. Second, a method is presented to design both the interface and controller. This approach is based on the natural invariants of the system.

1 Introduction

The hybrid control systems considered in this paper consist of three chief components: a continuous-time plant, a discrete event system (DES) controller, and an interface. This work uses a modeling framework for hybrid control previously developed by the authors [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Hybrid systems in general have attracted significant interest in recent years. Efforts include the work by Nerode, Kohn, et al. [11, 12, 13], Brockett [14], Ramadge, et al. [15, 16, 17], Varaiya, et al. [18, 19] and Tittus, et al. [20, 21]. Articles on these and other approaches can be found in [22].

In some of our previous work involving this framework, attention was focused on designing the controller, given a plant and interface [6, 8, 9]. In particular, a discrete event system, called the DES plant model, was developed to model the combined plant and interface. Then existing techniques for the design of discrete event system controllers were extended to design controllers for DES plant models.

Later work has focused on designing the interface as well as the controller [7, 10]. In this case the goal is to develop a method to design the interface and controller for a hybrid system when only the plant and control goals are given. The interface is designed to distinguish regions of the plant state space based on where the trajectories lead for a given control policy. Subsets of these regions, called *common flow regions*, are identified and then bounded using invariant manifolds. This provides a means for the system to determine when the state lies in such a region and to apply the appropriate control policy.

In this paper, we present both of these methods. First the modeling framework is described and the examples, which are used throughout, are presented. In Section 3, we describe controller design techniques for cases in which the plant and interface are both specified. This is refered to as the *logical approach*, because it is based on techniques developed for the control of logical discrete event systems. Then in the Section 4, the *invariant based approach* is presented. This is a method to design the interface using the invariants of the system, and the method leads directly to a controller design as well.

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 provides 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.

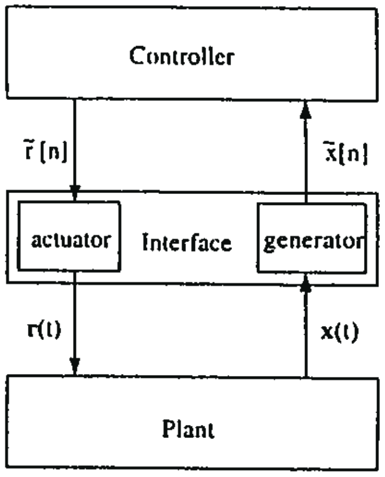


Fig. 1. Hybrid Control System

2.1 Plant

The plant is the part of the model which represents the entire continuous-time portion of the hybrid control system. The distinguishing feature of the plant is that it has a continuous state space, where the state takes on values that are real numbers, and evolves 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, continuous-time, controller.

Mathematically, the plant is represented by the equation

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

where $\mathbf{x}(t) \in \Re^n$ and $\mathbf{r}(t) \in \Re^m$ are the state and input vectors respectively. $f : \Re^n \times \Re^m \rightarrow \Re^n$ is a continuous function which satisfies the Lipschitz conditions, thus guaranteeing the existence and uniqueness of its solutions. 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*, $\delta : \tilde{S} \times \tilde{X} \rightarrow \tilde{S}$ is the state transition function, and $\phi : \tilde{S} \rightarrow \tilde{R}$ 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

$$\tilde{s}[n] = \delta(\tilde{s}[n-1], \tilde{x}[n]) \tag{2}$$
$$\tilde{r}[n] = \phi(\tilde{s}[n]) \tag{3}$$

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 nth symbol of a sequence of plant symbols. Subscripts are also used, e.g. $\tilde{x}_i$ which denotes the ith 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.

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 that will determine *when* a plant symbol should be generated, and second, a process is required to determine *which* particular plant symbol should be generated.

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. For the hybrid control systems studied here, a plant event is defined by a hypersurface that separates the plant state space into two open regions. A plant event occurs whenever the plant state crosses its associated hypersurface in a given direction.

The set of plant events recognized by the generator is given by a set of smooth functionals, $\{h_i : \Re^n \rightarrow \Re, i \in I\}$, defined on the state space of the plant. Each functional must satisfy the condition,

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

where ∇_x denotes the gradient with respect to x. This condition 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.

A plant event occurs whenever the state crosses a hypersurface, as given by the condition

$$\exists i \in I \text{ s.t. } h_i(\mathbf{x}(t)) = 0, \frac{d}{dt}h_i(\mathbf{x}(t)) \neq 0 \tag{5}$$

Notice that the condition stated above is true whenever a hypersurface is crossed, regardless of the direction.

The sequence of plant events is denoted as $\epsilon[n]$, where $\epsilon[n] = i$ indicates the nth event occurred by crossing the hypersurface, h_i. The sequence of plant event instants, that is the times at which the events occurred, is expressed as $\tau[n]$. These sequences are defined as follows.

$$\begin{aligned} \tau[0] &= 0 \\ \epsilon[0] &= 0 \end{aligned} \tag{6}$$
$$\tau[n] = \inf\{t \geq \tau[n-1] : \exists \epsilon[n] = \min\{i \in I : h_i(\mathbf{x}(t)) = 0 \\ \wedge \tfrac{d}{dt}h_i(\mathbf{x}(t)) \neq 0 \wedge (t > \tau[n-1] \vee i \neq \epsilon[n-1])\}\}$$

Perhaps a bit of explanation is required for the above equation. The sequence of plant events, $\epsilon[n]$, is ordered according to the time at which the events occur, and for events occurring simultaneously, the order is determined by the value of i. So, for example, if h_2 and h_3 are crossed at the same time, then $\epsilon[n] = 2$ and $\epsilon[n+1] = 3$.

The generator must also determine which plant symbol will be generated when a plant event occurs. The plant symbol notifies the controller that a plant event has occurred. It can also reflect which particular plant event has occurred and provide information about the current value of the state. This is modeled by a set of *plant symbol generating functions*, one for each hypersurface, which maps the state, at the time of the event, to a plant symbol. In the case of a silent event, the plant symbol generating function maps to the *null symbol*, which the controller will not recognize as a plant symbol.

The sequence of plant symbols is defined as

$$\tilde{x}[n] = \begin{cases} \alpha_i(\mathbf{x}(\tau_e[n])) & \text{if} \quad \frac{d}{dt}h_{\epsilon[n]}(\mathbf{x}(\tau[n])) < 0 \\ \epsilon & \text{otherwise} \end{cases} \tag{7}$$

The ϵ indicates the null symbol which is issued when the hypersurface is crossed in the direction opposite of the way the plant event was defined. This is discussed further below.

Several interesting issues arise in considering this mechanism for defining plant events. One issue is the question of whether the plant events should be "one-sided" or "two-sided", that is, should a plant event occur when the hypersurface is crossed in only one, or in either direction. The advantage of one-sided events is that they make the model more general, the disadvantage is that they cause complications when the model is used for analysis (some of these complications will be noted in remarks later in this paper). The solution adopted here is to defined the events as two-sided, but to only recognize the event when the hypersurface is crossed in a defined direction. When the hypersurface is crossed in the other direction, the event is called a *silent event*. Thus, the model can handle one-sided events without the complications mentioned above.

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

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

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. $\tau[n]$ is the time of the nth control symbol which is defined in equation 6.

The plant input, $\mathrm{r}(t)$, 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. We refer to the various inputs as *control policies*. Each controller symbol initiates a particular control policy.

2.4 Example - 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. Note that the actual temperature settings in a real system may be different. 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.

$$\dot{\mathrm{x}}(t) = .0025(T_O - \mathrm{x}(t)) + .02\mathrm{r}(t) \tag{9}$$

Here $\mathrm{x}(t)$ is the room temperature. T_O is the outside temperature, and $\mathrm{r}(t)$ 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.

$$h_1(\mathrm{x}) = \mathrm{x} - 70 \tag{10}$$
$$h_2(\mathrm{x}) = -\mathrm{x} + 75 \tag{11}$$

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.

$$\alpha_1(\xi) = cold \tag{12}$$
$$\alpha_2(\xi) = hot \tag{13}$$

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,

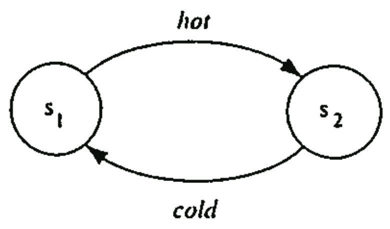


Fig. 2. Controller in Thermostat

on and *off*.

$$\phi(\tilde{s}_1) = on \qquad \phi(\tilde{s}_2) = off \tag{14}$$

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

$$\gamma(on) = 110 \qquad \gamma(off) = 0 \tag{15}$$

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.

2.5 Example - Double Integrator

The following simple example will be used throughout the paper to illustrate the work. The system consists of a double integrator plant which is controlled by a discrete event system. The control goal is to drive the state of the plant to the region of the origin.

First, the plant and interface will be presented. The plant is given by the differential equation,

$$\dot{\mathrm{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathrm{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathrm{r}(t) \tag{16}$$

The generator recognizes four plant events which occur when the following hy-
ersurfaces are crossed.

$$h_1(\mathrm{x}) = x_1 \qquad h_2(\mathrm{x}) = -x_1 \tag{17}$$
$$h_3(\mathrm{x}) = x_1 + 10x_2 \; h_4(\mathrm{x}) = -x_1 - 10x_2 \tag{18}$$

Two of these hypersurfaces lie on the x_2 axis and the other two lie on a line of slope
.1 passing through the origin. There are four events rather than two so that crossings
an be detected in both directions for each hypersurface. Symbols are attached to
he plant events as follows.

$$\alpha_1(\mathrm{x}) = \tilde{x}_1 \; \alpha_2(\mathrm{x}) = \tilde{x}_1 \tag{19}$$
$$\alpha_3(\mathrm{x}) = \tilde{x}_2 \; \alpha_4(\mathrm{x}) = \tilde{x}_2 \tag{20}$$

Notice that the same symbol can be used to label more than one plant event and
hat the value of the mapping α_i does not depend on the state, $\mathrm{x}(t)$, in this case. In
his example the plant symbol only identifies which hypersurface was crossed. Figure
illustrates this.

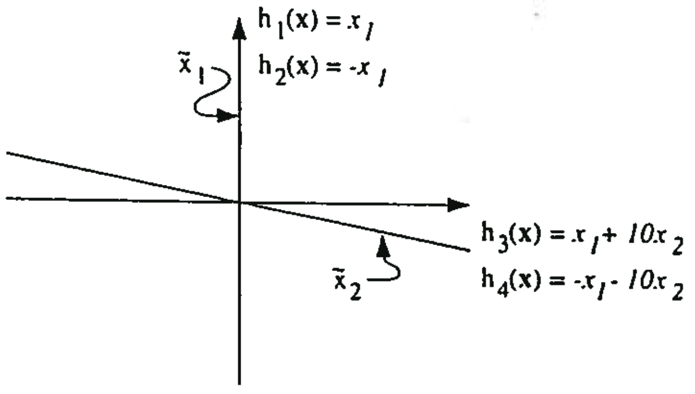


Fig. 3. Generator for Double Integrator Example

The actuator provides three possible inputs to the plant.

$$\gamma(\tilde{r}) = \begin{cases} -1 & \text{if } \tilde{r} = \tilde{r}_1 \\ 0 & \text{if } \tilde{r} = \tilde{r}_2 \\ 1 & \text{if } \tilde{r} = \tilde{r}_3 \end{cases} \tag{21}$$

These inputs were chosen so that the plant can be driven to the origin by applying
hem in the proper sequence.

Finally, the controller is the four state automaton pictured in Figure 4. The ouput
unction of the controller is the following.

$$\phi(\tilde{s}_1) = \tilde{r}_1 \; \phi(\tilde{s}_2) = \tilde{r}_2 \tag{22}$$
$$\phi(\tilde{s}_3) = \tilde{r}_2 \; \phi(\tilde{s}_4) = \tilde{r}_3 \tag{23}$$

$$\begin{tikzcd}
s_4 \arrow[r, "\tilde{x}_1"] & s_1 \arrow[d, "\tilde{x}_2"] \\
s_3 \arrow[u, "\tilde{x}_2"] & s_2 \arrow[l, "\tilde{x}_1"]
\end{tikzcd}$$

Fig. 4. Controller for Double Integrator Example

The controller in this example was designed to drive the state of the double integrator to the origin, and Figure 5 shows that this goal is indeed achieved. However, the design was adhoc. The following sections will show a more systematic method of designing an interface and controller. This example will be used again there.

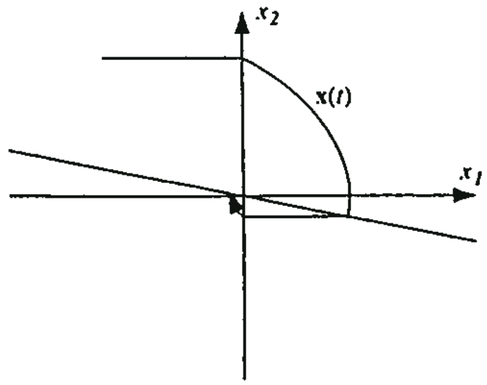


Fig. 5. State Space Trajectory for Double Integrator Example

3 Logical Approach

3.1 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.

The DES plant model is an 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 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

$$\xi_1 \equiv_P \xi_2 \text{ iff } h_i(\xi_1)h_i(\xi_2) > 0, \forall i \in I. \tag{24}$$

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, $b \in \mathrm{B}^I$, such that b_i is the ith 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.

$$\tilde{P} = \{\xi \in \Re^n : h_i(\xi) \neq 0, i \in I\} / \equiv_P \tag{25}$$

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

$$\tilde{p}_b = \{\xi \in \Re^n : b_i = 0 \Rightarrow h_i(\xi) > 0 \text{ and } b_i = 1 \Rightarrow h_i(\xi) < 0\} \tag{26}$$

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

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

$$\tilde{p}[n] = \tilde{p}_b \tag{27}$$

where

$$\lim_{\epsilon \to 0^+} \mathbf{x}(\tau[n] + \epsilon) \in \tilde{p}_b \tag{28}$$

So the current state of the DES corresponds to the most recently entered region of the plant state space. The limit must be used because at exactly $\tau[n]$ 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 $j \in I$,

$$\mathcal{N}(h_j) = \mathcal{N}(h_i) \Rightarrow b_j \neq c_j$$

$$\xi \in \overline{\tilde{p}_b} \cap \overline{\tilde{p}_c}.$$

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.

Proposition 4. *Given a hybrid control system, described by (1) - (8), with f and h_i smooth, if $\exists i \in I$ and $\xi \in \mathcal{N}(h_i)$ such that following conditions are satisfied,*

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

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 computation required.

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

Proposition 5. *Given a hybrid control system described by (1) - (8), 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 4 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 Example - Thermostat

The thermostat/heater example has a simple DES plant model which is useful to illustrate how these models work. Figure 6 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.

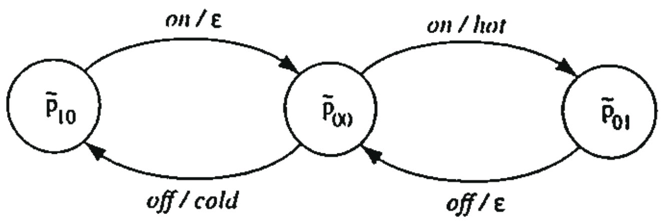
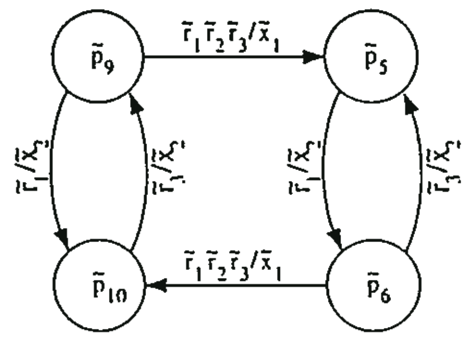


Fig. 6. DES Plant for Thermostat/Heater

3.3 Example - Double Integrator

Now we return to the double integrator example from Section 2. Using Proposition 4, we can extract the DES plant for this system. It is shown in Figure 7. 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 5 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 $\{\tilde{x}_1\} = \lambda(\tilde{p}_9, \tilde{p}_5)$.



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

Fig. 7. DES Plant for Double Integrator

Now that the plant and interface have been converted to a discrete event system, techniques for controller design from that area can be applied.

3.4 Logical Approach to DES Control

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 [23, 24, 25, 26, 27]. 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 our 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, in the hybrid system context, from the automata of earlier frameworks.

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

$$\tilde{p}[n+1] \in \psi(\tilde{p}[n], \tilde{r}[n]) \, \forall n \in \mathrm{N} \tag{29}$$

$$\tilde{x}[n] \in \lambda(\tilde{p}[n-1], \tilde{p}[n]) \, \forall n \in \mathrm{N} \tag{30}$$

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 the current state of the DES plant can be ascertained from its initial state and past output, the output provides more useful feedback.

Definition 7. 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 8. *If a DES plant model is observable, then for any initial state, $\tilde{p}[0]$ and sequence of plant symbols, $\tilde{x} \in L$, 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, $obs(q_0, \tilde{x})[n]$, where $q_0 \in \tilde{P}$ was the initial state.

3.6 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 [23]. 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

$$\bar{K}\tilde{X}_u \cap L \subset \bar{K} \tag{31}$$

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 equation 31 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 9. 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

$$\tilde{x}\lambda(q, \psi(q, \rho)) \subset K, \tag{32}$$

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

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 10. *If the language K is controllable according to (9), then a controller can be designed which will restrict the given DES plant to the language K.*

Proof. Let the controller be given by $con : \tilde{X}^* \rightarrow \tilde{R}$ where $con(\tilde{x}) \in \{\rho \in \tilde{R} : \tilde{x}\lambda(q, \psi(q, \rho)) \subset K, q = obs(q_0, \tilde{x})[N]\}$. $con(\tilde{x})$ is guaranteed to be non-empty by (32). We can now show by induction that $\tilde{x} \in L_{con} \Rightarrow \tilde{x} \in K$.

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 $L_f{}^i$, $L_f{}^{i+1} = \{w \in \tilde{X}^* : w = \tilde{x}\lambda(q, \psi(q, con(\tilde{x})), \tilde{x} \in L_f{}^i\}$. Now with the definition of $con(\tilde{x})$ and (32) 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 (32) should reduce to those of (31) under the appropriate restrictions. 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. [23] and [26] describe and provide a method for finding this behavior which is referred to as the *supremal controllable sublanguage*, $K^\dagger$, of the desired language. The supremal controllable sublanguage is the largest subset of K which can be attained by a controller. $K^\dagger$ can be found via the following iterative procedure.

$$K_0 = K \tag{33}$$
$$K_{i+1} = \{w : w \in \bar{K}, \bar{w}\tilde{X}_u \cap L \subset \overline{K_i}\} \tag{34}$$
$$K^\dagger = \lim_{i\rightarrow\infty} K_i \tag{35}$$

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.

$$K_0 = K \tag{36}$$
$$K_{i+1} = \{w \in K : \forall \tilde{x} \in \bar{w}\ \exists\, \rho \in \tilde{R} \text{ such that } \tilde{x}\lambda(q, \psi(q, \rho)) \subset K_i\} \tag{37}$$
$$K^\dagger = \lim_{i\rightarrow\infty} K_i \tag{38}$$

This result yields the following proposition.

Proposition 11. *For a DES plant and language K, $K^\dagger$ is controllable and contains all controllable sublanguages of K.*

Proof. From (37) we have

$$K^\dagger = \{w \in K : \forall \tilde{x} \in \bar{w}\ \exists\, \rho \in \tilde{R} \text{ such that } \tilde{x}\lambda(q, \psi(q, \rho)) \subset K^\dagger\} \tag{39}$$

which implies

$$\tilde{x} \in K^\dagger \Rightarrow \exists\, \rho \in \tilde{R} \text{ such that } \tilde{x}\lambda(q, \psi(q, \rho)) \subset K^\dagger \tag{40}$$

From (40) it is clear that $K^\dagger$ is controllable. We prove that every prefix closed, controllable subset of K is in $K^\dagger$ by assuming there exists $M \subset K$ such that M is controllable but $M \not\subset K^\dagger$ and showing this leads to a contradiction.

$$\exists M \subset K \text{ s.t. } M \not\subset K^\dagger \tag{41}$$
$$\Rightarrow \quad \exists w \in M \text{ s.t. } w \notin K^\dagger \tag{42}$$
$$\Rightarrow \quad \exists i \text{ s.t. } w \in K_i, w \notin K_{i+1} \tag{43}$$
$$\Rightarrow \quad \exists \tilde{x} \in \bar{w} \text{ s.t. } \forall \rho \in \tilde{R}, \tilde{x}\lambda(q, \psi(q, \rho)) \not\subset K_i \tag{44}$$
$$\Rightarrow \exists w' \in \tilde{x}\lambda(q, \psi(q, \rho)) \text{ s.t. } w' \in M, w' \notin K_i \tag{45}$$
$$\Rightarrow \quad \exists j < i \text{ s.t. } w' \in K_j, w' \notin K_{j+1} \tag{46}$$

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.

3.7 Example - Double Integrator

We use the double integrator example again because the DES plant was found earlier. This DES is represented by the automaton in Figure 8.

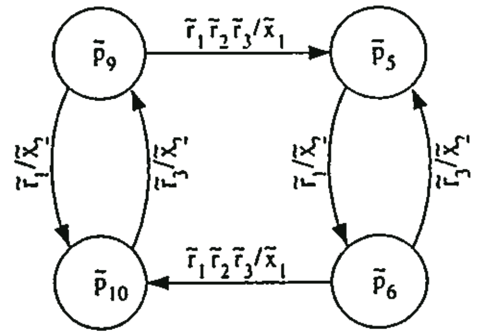
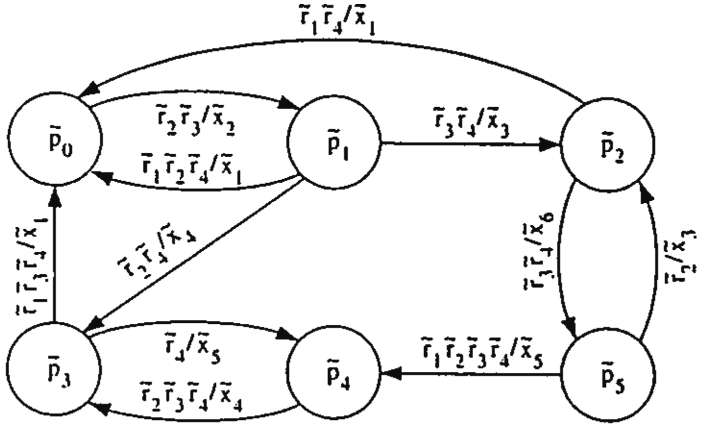

$\tilde{p}_9$
$\tilde{p}_5$
$\tilde{p}_{10}$
$\tilde{p}_6$
$\tilde{r}_1\tilde{r}_2\tilde{r}_3/\tilde{x}_1$
$\tilde{r}_1/\tilde{x}_2$
$\tilde{r}_3/\tilde{x}_2$
$\tilde{r}_1/\tilde{x}_2$
$\tilde{r}_3/\tilde{x}_2$
$\tilde{r}_1\tilde{r}_2\tilde{r}_3/\tilde{x}_1$

Fig. 8. DES Plant Model for Example 1

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 Equation (32). Therefore according to Proposition 10, a controller can be designed to achieve the stated control goal.

.8 A More Complex DES Plant Model

his example has a richer behavior and will illustrate the generation of a supremal ontrollable sublanguage as well as the design of a controller. We start immediately vith the DES plant model shown in Figure 9.



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

ig. 9. DES Plant Model for Example 2

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

$$L_m = (\tilde{x}_2(\tilde{x}_1 + \tilde{x}_4(\tilde{x}_5\tilde{x}_4)^*\tilde{x}_1 + \tilde{x}_3(\tilde{x}_6\tilde{x}_3)^*(\tilde{x}_1 + \tilde{x}_6\tilde{x}_5\tilde{x}_4(\tilde{x}_5\tilde{x}_4)^*\tilde{x}_1)))^* \tag{47}$$

uppose we want to control the DES so that it never enters state $\tilde{p}_4$. We simply emove the transitions to $\tilde{p}_4$ and then compute the resulting language. This desired anguage is therefore

$$K = \overline{(\tilde{x}_2(\tilde{x}_1 + \tilde{x}_4\tilde{x}_1 + \tilde{x}_3(\tilde{x}_6\tilde{x}_3)^*\tilde{x}_1))^*} \tag{48}$$

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 is defined in equation (38). The supremal controllable sublanguage is

$$K^{\uparrow} = K_1 = \overline{(\tilde{x}_2(\tilde{x}_1 + \tilde{x}_4\tilde{x}_1 + \tilde{x}_3\tilde{x}_1))^*} \tag{49}$$

Obtaining a DES controller once the supremal controllable sublanguage has been ound is straight forward. 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 10 and its output function, ϕ, is as follows:

$$\phi(\tilde{s}_1) = \tilde{r}_2 \quad \phi(\tilde{s}_2) = \tilde{r}_4 \tag{50}$$

$$\phi(\tilde{s}_3) = \tilde{r}_1 \quad \phi(\tilde{s}_4) = \tilde{r}_1 \tag{51}$$

□

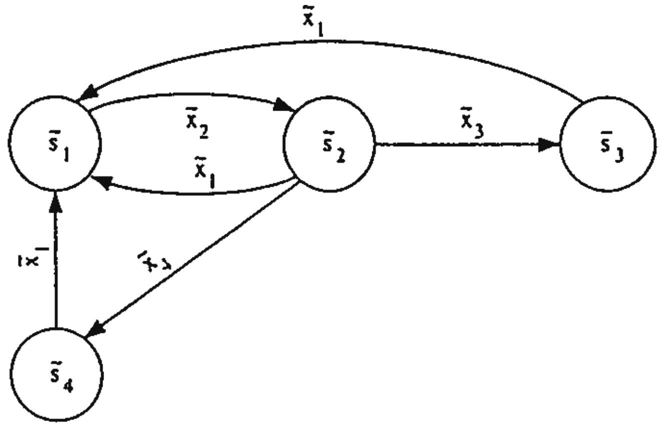


Fig. 10. DES Controller for Example 2

3.9 Remarks

The approach described above is also discussed in detail in [9]. There is, in addition, a discussion of determinism and quasideterminism. The hybrid control system is also extended to include systems with discrete time plants. An example with a nonlinear plant is presented.

For a detailed description of the derivation of a formula for computing the supremal controllable sublanguage of a given language see [28].

4 Invariant Based Approach

If the interface is not given the designer must design both the interface and the controller. One could, of course, design the interface using any technique and then use the logical approach to design the controller. Here a methodology is presented to design the controller and the interface together based on the natural invariants of a plant described by

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{r}(t)) \tag{52}$$

where certain smoothness assumptions apply.

In particular, this section discusses the design of the generator, which is part of he interface, and the design of the controller. We assume that the plant is given, he set of available control policies is given, and the control goals are specified as ollows. Each control goal for the system is given as a starting set and a target set, ach of which is an open subset of the plant state space. To realize the goal, the ontroller must be able to drive the plant state from anywhere in the starting set to omewhere in the target set using the available control policies. Generally, a system vill have multiple control goals.

To successfully control the plant, the controller must know which control policy o apply and when to apply it. The controller receives all its information about the lant from the generator, and therefore the generator must be designed to provide hat information which the controller requires.

We propose the following solution to this design problem. For a given target egion, identify the states which can be driven to that region by the application of single control policy. If the starting region is contained within this set of states, he control goal is achievable via a single control policy. If not, then this new set f states can be used as a target region and the process can be repeated. This will esult in a set of states which can be driven to the original target region with no nore than two control policies applied in sequence. This process can be repeated ntil the set of states, for which a sequence of control policies exists to drive them o the target region, includes the entire starting region (provided the set of control olicies is adequate as mentioned below).

When the regions have been identified, the generator is designed to tell the controller, via plant symbols, which region the plant state is currently in. The controller vill then call for the control policy which drives the states in that region to the target egion.

.1 Generator Design

To describe the regions mentioned above, we use the concept of the flow [29]. Let he flow for the plant (1) be given by $F_k : \mathbf{X} \times \Re \to \mathbf{X}$, where

$$\mathbf{x}(t) = F_k(\mathbf{x}(0), t). \tag{53}$$

The flow represents the state of the plant after an elapsed time of t, with an initial tate of $\mathbf{x}(0)$, and with a constant input of $\gamma(\tilde{r}_k)$. Since the plant is time invariant, here is no loss of generality when the initial state is defined at $t = 0$. The flow is lefined over both positive and negative values of time. The flow can be extended ver time using the forward flow function, $F_k^+ : \mathbf{X} \to \mathbf{P}(\mathbf{X}^n)$, and the backward flow unction, $F_k^- : \mathbf{X} \to \mathbf{P}(\mathbf{X}^n)$, which are defined as follows.

$$F_k^+(\xi) = \bigcup_{t \geq 0} \{F_k(\xi, t)\} \tag{54}$$

$$F_k^-(\xi) = \bigcup_{t \leq 0} \{F_k(\xi, t)\} \tag{55}$$

The backward and forward flow functions can be defined on an arbitrary set of states in the following natural way.

$$F_k^+(\mathbf{A}) = \bigcup_{\xi \in \mathbf{A}} \{F_k^+(\xi)\} \tag{56}$$

$$F_k^-(\mathbf{A}) = \bigcup_{\xi \in \mathbf{A}} \{F_k^-(\xi)\} \tag{57}$$

where $\mathbf{A} \subset \mathbf{X}$. For a target region, T, $F_k^-(T)$ is the set of initial states from which the plant can be driven to T with the input $\gamma(\tilde{r}_k)$. In addition, $F_k^+(T)$ is the set of states which can be reached with input $\gamma(\tilde{r}_k)$ and an initial state in T.

Now a generator design procedure can be described using the backward flow function. This is a preliminary procedure, upon which the final design method, developed subsequently, is based. For a given starting region, $S \subset \mathbf{X}$, and target region, $T \subset \mathbf{X}$, use the following algorithm.

1. If $S \subset T$, stop.
2. Identify the regions, $F_k^-(T), \forall \tilde{r}_k \in \tilde{R}$.
3. Let $T = \bigcup_{\tilde{r}_k \in \tilde{R}} F_k^-(T)$
4. Go to 1.

There are two problems associated with this algorithm as stated. First, it will not stop if there is no sequence of available control policies which will achieve the control goal, and second, actually identifying the regions given by the flow functions is quite involved. The first issue is related to the adequacy of the available control policies and will not be dealt with here. The second problem will be addressed. The difficulty in identifying a region given by a flow function is integrating over all the points in the target region. In the generator design procedure developed here, we will concentrate on finding a subset of the region $F_k^-(T)$, rather than the region itself. By definition, all the trajectories passing through $F_k^-(T)$ lead to the target region, T, and therefore all the trajectories found in a subset of $F_k^-(T)$ will also lead to the target.

Here, we will focus on identifying subsets of $F_k^-(T)$ which we call *common flow regions*. Common flow regions are bounded by invariant manifolds and an exit boundary. The invariant manifolds are used because the state trajectory can neither enter nor leave the common flow region through an invariant manifold. The exit boundary is chosen as the only boundary through which state trajectories leave the common flow region.

To design the generator, it is necessary to select the set of hypersurfaces, $\{h_i : \mathbf{X} \to \Re \mid i \in I\}$ and the associated functions, $\{\alpha_i : \mathcal{N}(h_i) \to \tilde{R} \mid i \in I\}$, described in Section 2.3. These hypersurfaces make up the invariant manifolds and exit boundaries mentioned above, as well as forming the boundary for the target region(s).

A target region, T, is specified as

$$T = \{\xi \in \mathbf{X} : \forall i \in I_T, h_i(\xi) < 0\}, \tag{58}$$

where I_T is the index set indicating which hypersurfaces bound the target region. A common flow region, B, is specified as

$$B = \{\xi \in \mathbf{X} : h_i(\xi) < 0, h_e(\xi) > 0, \forall i \in I_B\}. \tag{59}$$

where I_B is an index set indicating which hypersurfaces form the invariant manifolds bounding B and h_e defines the exit boundary for B.

The goal, of course, is that B should include only states whose trajectories lead to the target region. Figure 11 shows an example of this where $I_T = \{1\}$ and $I_B = \{2,3\}$. The target region, T, is surrounded by h_1, the common flow region lies between h_2 and h_3 above the exit boundary, h_e.

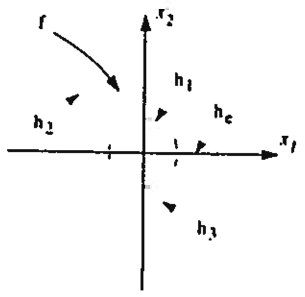


Fig. 11. Target Region and Invariants

We now present two propositions which can be used to determine the suitability of a set of hypersurfaces to achieve our goal of identifying a common flow region. In different situations, one of the propositions may be easier to apply than the other. The following propositions give sufficient conditions for the hypersurfaces bounding B and T to ensure that all state trajectories in B will reach the target region.

Proposition 12. *Given the following:*

1. *A flow generated by a smooth vector field, f_k*
2. *A target region, $T \subset \mathbf{X}$*
3. *A set of smooth hypersurfaces, $h_i, i \in I_B \subset 2^I$*
4. *A smooth hypersurface (exit boundary), h_e*

such that $B = \{\xi \in \mathbf{X} : h_i(\xi) < 0, h_e(\xi) > 0, \forall i \in I_B\} \neq \emptyset$. For all $\xi \in B$ there is a finite time, t, such that $F_k(\xi,t) \in T$, if the following conditions are satisfied:

1. $\nabla_\xi h_i(\xi) \cdot f(\xi) = 0, \forall i \in I_B$
2. $\exists \epsilon > 0, \nabla_\xi h_e(\xi) \cdot f(\xi) < -\epsilon, \forall \xi \in B$
3. $B \cap \mathcal{N}(h_e) \subset T$

Proof. The proof of this proposition is straightforward. The first condition of the proposition, which can be rewritten as

$$\frac{dh_i(\mathbf{x}(t))}{dt} = 0, \tag{60}$$

precludes the state trajectory crossing any hypersurface indexed by the set I_B, thus ensuring no trajectory in B will leave B except through the remaining boundary. The second condition, which can be rewritten as

$$\frac{dh_e(\mathbf{x}(t))}{dt} < -\epsilon, \tag{61}$$

ensures that within a finite time,

$$t < \frac{h_e(\xi)}{\epsilon}, \tag{62}$$

the trajectory at $\xi \in B$ will cross the exit boundary. The final condition guarantees that any trajectory leaving B through the exit boundary will be in the target region when it does so. Together these conditions are sufficient to guarantee that any state in B will enter the target region in finite time.

The second proposition uses a slightly different way of specifying a common flow region. In addition to the invariant manifolds and the exit boundary, there is also a cap boundary. The cap boundary is used to obtain a common flow region which is bounded. So for this case

$$B = \{\xi \in \mathbf{X} : h_i(\xi) < 0, h_e(\xi) > 0, h_c(\xi) < 0, \forall i \in I_B\}. \tag{63}$$

Proposition 13. *Given the following:*

1. *A flow generated by a smooth vector field, f_k*
2. *A target region, $T \subset \mathbf{X}$*
3. *A set of smooth hypersurfaces, $h_i, i \in I_B \subset 2^I$*
4. *A smooth hypersurface (exit boundary), h_e*
5. *A smooth hypersurface (cap boundary), h_c*

such that $B = \{\xi \in \mathbf{X} : h_i(\xi) < 0, h_e(\xi) > 0, h_c(\xi) < 0, \forall i \in I_B\} \neq \emptyset$ and $\overline{B}$ (closure of B) is compact. For all $\xi \in B$ there is a finite time, t, such that $F_k(\xi,t) \in T$, if the following conditions are satisfied:

1. $\nabla_\xi h_i(\xi) \cdot f(\xi) = 0, \forall i \in I_B$
2. $\nabla_\xi h_c(\xi) \cdot f(\xi) < 0, \forall \xi \in B \cap \mathcal{N}(h_c)$
3. $B \cap \mathcal{N}(h_e) \subset T$
4. *There are no limit sets in $\overline{B}$*

Proof. As in Proposition 12, the first condition precludes the state trajectory crossing any hypersurface indexed by the set I_B, thus ensuring no trajectory in B will leave B except through one of the remaining boundaries. The second condition, which can be rewritten as

$$\frac{dh_c(\mathbf{x}(t))}{dt} < 0, \tag{64}$$

ensures that no trajectory can leave B through the cap boundary. Thus, the exit boundary provides the only available egress from B. The third condition guarantees that any trajectory leaving B through the exit boundary will be in the target region when it does so. The final condition permits the application of a previously known result [30], stating that any state within a compact set without limit sets will leave that compact set in finite time.

Consider the hypersurfaces defined by $\{h_i : i \in I_B\}$. These hypersurfaces must first be invariant under the vector field of the given control policy, f. This can be achieved by choosing them to be integral manifolds of an $n-1$ dimensional distribution which is invariant under f. An $n-1$ dimensional distribution, $\Delta(x)$, is nvariant under f if it satisfies

$$[f(x), \Delta(x)] \subset \Delta(x). \tag{65}$$

where the $[f(x), \Delta(x)]$ indicates the Lie bracket. Of the invariant distributions, those hat have integral manifolds as we require, are exactly those which are involutive according to Frobenius). This means

$$\delta_1(x), \delta_2(x) \in \Delta(x) \Rightarrow [\delta_1(x), \delta_2(x)] \in \Delta(x). \tag{66}$$

Therefore by identifying the involutive distributions which are invariant under the vector field, f, we have indentified a set of candidate hypersurfaces. For details of hese relationships between vector fields and invariant distributions, see [31].

Since an $n-1$ dimensional involutive distribution can be defined as the span of $n-1$ vector fields, over each of which it will then be invariant, and the control policy only gives one vector field, f, there will be more than one family of hypersurfaces which are all invariant under f. The set of all invariant hypersurfaces can be found in erms of $n-1$ functionally independent mappings which form the basis for the desired et of functionals, $\{h_i : i \in I_B\}$. This basis is obtained by solving the characteristic equation

$$\frac{dx_1}{f_1(x)} = \frac{dx_2}{f_2(x)} = \cdots = \frac{dx_n}{f_n(x)} \tag{67}$$

where $f_i(x)$ is the ith element of $f(x)$.

4.2 Controller Design

n previous work using this framework for hybrid control systems, the interface vas assumed to be given and the controller was designed using the given plant and interface; see Section 3 and [6, 8, 9]. In those cases, the plant and interface vere modeled as a discrete event system, called the *DES plant model*, and existing DES controller design techniques were adapted and used to obtain a controller. The drawback was that there was no guarantee that the desired behavior could be achieved with the given plant and interface.

Now, with the generator design technique described in Section 4.1, the controller lesign is anticipated by the design of the interface. This represents an improvement over the previous situation because now there is no question that the control goal can be achieved once the interface has been successfully designed, and furthermore he actual controller design has been largely determined by the interface design.

Once the interface has been designed as described in Section 4.1, the design of he controller involves two steps. The first step is to construct one subautomaton or each control goal. This is the step which is already determined by the interface lesign. The second step is the connection of these subautomata to create a single DES controller. This step will depend upon the order in which the simpler control goals are to be achieved. For example, if a chemical process is to produce a sequence

of different products, then each subautomaton in the controller would be designed to produce one of the products, and these subautomata would be connected to produce the products in the desired sequence.

The hypersurfaces in the generator divide the state space of the plant into a number of *cells*. Two states are in the same cell exactly when they are both on the same side (positive or negative) with respect to each hypersurface. States which lie on a hypersurface are not in any cell.

The first step in creating the controller is the contruction of the subautomata, one for each individual control goal. Each subautomaton is constructed in the following way.

i. Create a controller state to represent each cell.
ii. Place transitions between states which represent adjacent cells.
iii. Label each transition with the plant symbol which is generated by the hypersurface separating the associated cells.

We now have a subautomaton which can follow the progress of the plant state as it moves from cell to cell. Next the controller output function must be designed for each subautomaton.

The controller symbol output by a given controller state depends on which common flow region contains the associated cell. Each common flow region was constructed using a specific control policy, and the control symbol which initiates that control policy should be output by controller states representing cells contained in that common flow region. However, in general, common flow regions will overlap, meaning a given cell can lie in more than one common flow region. In such cases treat the cell as lying within the common flow region which is closest to the target region. Distance, in this case, is the number additional control policies which must be used to reach the target region. If common flow regions are both the same distance, then the choice is arbitrary, though the common flow region which is favored in one case must then be favored in all such cases. States which represent cells not contained in any common flow region or target region will never be visited and can thus be deleted.

Once the individual subautomata have been constructed they must be connected to form a single controller. This can be accomplished by following these steps for each subautomaton.

i. Remove the state(s) which represent cells in the target region as well all transitions emanating from such states.
ii. Connect the dangling transitions to states in the subautomaton which achieves the next desired control goal. The connections will be to the states which represent the same cells as the states which were removed.

In this way, as soon as one control goal is achieved, the system will begin working on the next one. The actual order in which each control goals are pursued is up to the designer.

4.3 Example - Double Integrator

Consider the double integrator example from Section 2.5. Suppose we are given the plant,

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{r}(t), \tag{68}$$

three available control policies,

$$\mathbf{r}(t) \in \{-1, 0, 1\}, \tag{69}$$

and the following control goal: drive the plant state to the interior of the unit circle from any initial point. So the starting set consists of the entire state space and the following control goal: drive the plant state to the interior of the unit circle from any initial point. So the starting set consists of the entire state space, and the target set is

$$T = \{\xi \in \mathbf{X} : \xi_1^2 + \xi_2^2 < 1\}. \tag{70}$$

The target set is bounded by the hypersurface given by

$$h_T(\xi) = \xi_1^2 + \xi_2^2 - 1 \tag{71}$$

The first step is to calculate the invariants which can be used to obtain hypersurfaces. There are three families of invariants, one for each of the three control policies.

$$\pm(\xi_1 + \frac{1}{2}\xi_2^2 + c_1) \tag{72}$$

$$\pm(\xi_1 - \frac{1}{2}\xi_2^2 + c_2) \tag{73}$$

$$\pm(\xi_2 + c_3) \tag{74}$$

The first hypersurface, h_1, is used to identify the target region.

$$h_1(\xi) = h_T(\xi) = \xi_1^2 + \xi_2^2 - 1 \tag{75}$$

A tube entering T under the first control policy, $\mathbf{r}(t) = -1$, is bounded by

$$h_2(\xi) = -\xi_1 - \frac{1}{2}\xi_2^2 - .9 \tag{76}$$

$$h_3(\xi) = \xi_1 + \frac{1}{2}\xi_2^2 - .9 \tag{77}$$

and

$$h_e(\xi) = h_4(\xi) = \xi_2 + .1 \tag{78}$$

These hypersurfaces satisfy Proposition 12. Identify this tube as B_1.

$$B_1 = \{\xi : h_i(\xi) < 0, h_4(\xi) > 0, i \in \{2,3\}\} \tag{79}$$

Likewise, a tube entering T under the third control policy is bounded by

$$h_5(\xi) = -\xi_1 + \frac{1}{2}\xi_2^2 - .9 \tag{80}$$

$$h_6(\xi) = \xi_1 - \frac{1}{2}\xi_2^2 - .9 \tag{81}$$

and

$$h_e(\xi) = h_7(\xi) = -\xi_2 + .1 \tag{82}$$

Identify this tube as B_2.

$$B_2 = \{\xi : h_i(\xi) < 0, h_7(\xi) > 0, i \in \{5,6\}\} \tag{83}$$

Figure 12 illustrates what we have so far. Now the target can be extended to include B_1 or B_2 and more tubes can be obtained. Let the new target be given by $T' = T \cup B_1$. A tube entering T' under the second control policy is bounded by choosing

$$I_{B_3} = \{7\} \tag{84}$$

and $e = 2$. A tube entering $T'' = T' \cup B_3$ under the third control policy is bounded by choosing

$$I_{B_4} = \{6\} \tag{85}$$

and $e = 7$. Figure 13 gives a final picture of the hypersurfaces and regions involved in this example.

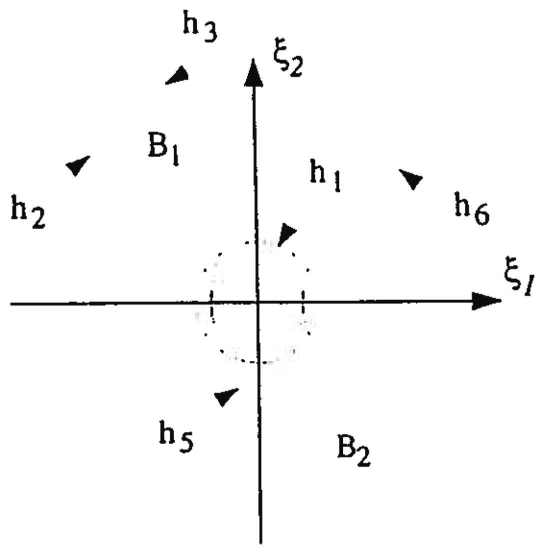


Fig. 12. Target Region and Invariants

There is only one control goal for this example and therefore the entire controll will consist of a single subautomata. Start by creating a controller state $\tilde{s}_T$ whi is associated with the target region. Two tubes, labeled B_1 and B_2, were identifi which lead to the target region. So create two more controller states, $\tilde{s}_1$ and $\tilde{s}_2$. consists of the trajectories which reach the target region under control policy $\tilde{r}_1$ a therefore $\phi(\tilde{s}_1) = \tilde{r}_1$, likewise $\phi(\tilde{s}_2) = \tilde{r}_3$. Connect $\tilde{s}_1$ to $\tilde{s}_T$ with a tranisition label

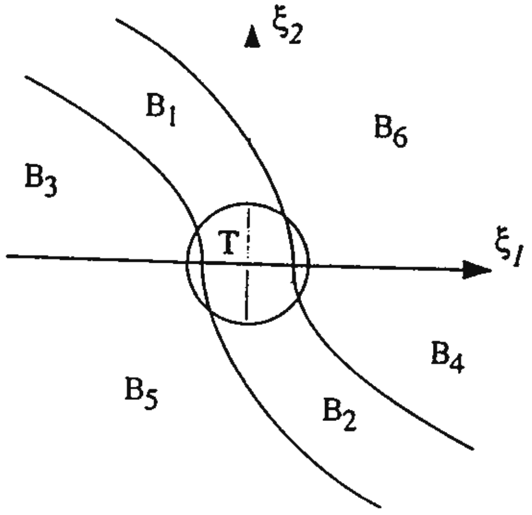


Fig. 13. Target Region and Invariants

$\bar{x}_1$ which is generated when the plant state crosses h_1 to enter the target region. Do the same for $\bar{s}_2$. Next, create $\bar{s}_3$ to go with B_3, and add a transition to $\bar{s}_1$ labeled $\bar{x}_2$. When all the tubes have their associated states and transitions the controller shown in Figure 14 results.

4.4 Example - Triple Integrator

With the double integrator example, it is easy to see how the invariant surfaces are used. The technique can also be used in more complicated cases where it is not so ntuitively obvious. Consider the triple integrator,

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} r(t), \qquad (86)$$

vith the same three available control policies,

$$r(t) \in \{-1, 0, 1\}. \qquad (87)$$

This time the control goal is to drive the state to the unit sphere from any initial state. First find a basis for the invariants by solving the characteristic equation

$$\frac{dx_1}{x_2} = \frac{dx_2}{x_3} = \frac{dx_3}{r} \qquad (88)$$

Two functions are obtained,

$$h_a(\xi) = r\xi_2 - \frac{1}{2}\xi_3^2 + c_1 \qquad (89)$$

$$h_b(\xi) = r^2\xi_1 + \frac{1}{3}\xi_3^3 - r\xi_2\xi_3 - rc_2. \qquad (90)$$

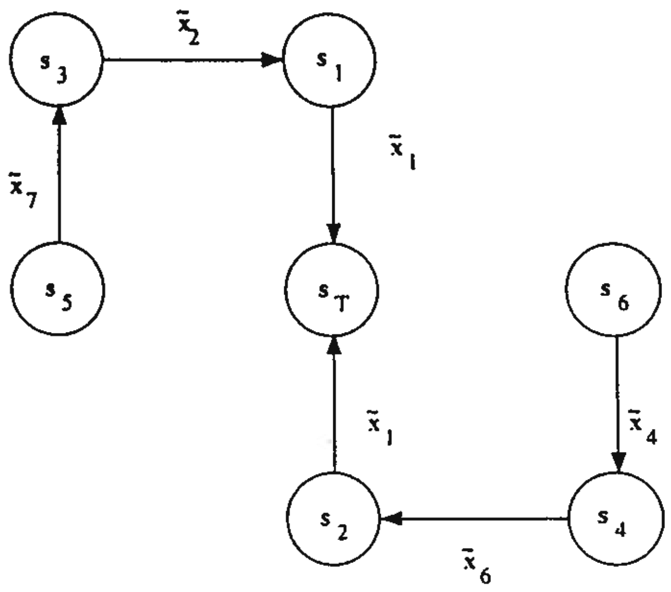


Fig. 14. Controller

where r is the input (control policy) and c_1 and c_2 are arbitrary constants. Examp hypersurfaces for $r = 1$ and $c_1 = c_2 = 0$ are shown in Figures 15 and 16.

The target region is bounded by the hypersurface, h_1, i.e. $I_T = \{1\}$.

$$h_1(\xi) = \xi_1^2 + \xi_2^2 + \xi_3^2 - 1 \qquad (9$$

Now identify a "tube" of trajectories which enters the target region under the inpu $r(t) = 1$. The following hypersurfaces are used.

$$h_2(\xi) = -\xi_2 + \frac{1}{2}\xi_3^2 - \sqrt{2} \qquad (9$$

$$h_3(\xi) = \xi_2 - \frac{1}{2}\xi_3^2 - \sqrt{2} \qquad (9$$

$$h_4(\xi) = \xi_1 + \frac{1}{3}\xi_3^3 - \xi_2\xi_3 - \sqrt{2} \qquad (9$$

$$h_5(\xi) = -\xi_1 - \frac{1}{3}\xi_3^3 + \xi_2\xi_3 - \sqrt{2} \qquad (9$$

The tube runs through the origin, where the target is centered, and the constar values, $\pm\sqrt{2}$, where choosen so that the tube passes through the target.

4.5 Remarks

Preliminary results were presented in [7], where they were discussed in the conte of digital control. A more extensive presentation can be found in [32].

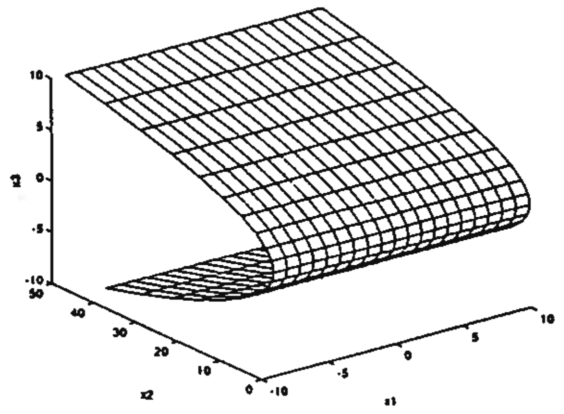


Fig. 15. Invariant for h_a

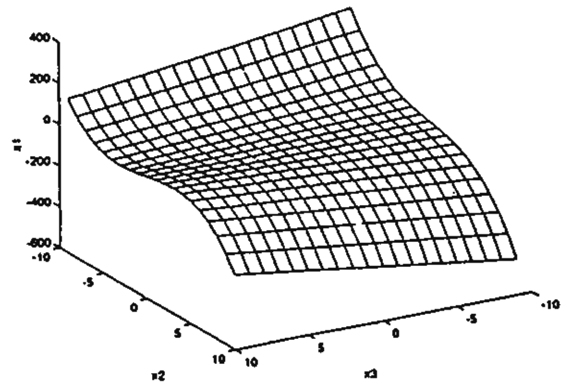


Fig. 16. Invariant for h_b

5 Conclusion

This paper presented two methods to design a hybrid control system. The first method can be used when the plant and interface of the system have been specified and a controller is required. The second method allows the design of the interface (the generator) as well as of the controller.

References

1. P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon, "Hybrid system modeling and autonomous control systems", In *Hybrid Systems*, edited by R. L. Grossman, A. Nerode, A. P. Ravn and H. Rischel, vol. 736 of *Lecture Notes in Computer Science*, pp. 366–392. Springer-Verlag, 1993.

2. 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.
3. 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, Oct. 1991.
4. J. A. Stiver and P. J. Antsaklis, "Modeling and analysis of hybrid control systems", In *Proceedings of the 31st Conference on Decision and Control*, pp. 3748–3751, Tucson, AZ, Dec. 1992.
5. J. A. Stiver and P. J. Antsaklis, "State space partitioning for hybrid control systems", In *Proceedings of the American Control Conference*, pp. 2303–2304, San Francisco, California, June 1993.
6. J. A. Stiver and P. J. Antsaklis, "On the controllability of hybrid control systems", In *Proceedings of the 32nd Conference on Decision and Control*, pp. 3748–3751, San Antonio, TX, Dec. 1993.
7. J. A. Stiver, P. J. Antsaklis, and M. D. Lemmon, "Digital control from a hybrid perspective", In *Proceedings of the 33rd Conference on Decision and Control*, pp. 4241-4246, Lake Buena Vista, FL, Dec. 1994.
8. 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. To appear as a chapter in the IEEE Press book Intelligent Control: Theory and Applications.
9. 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, October 1994.
10. 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, January 1995.
11. 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.
12. W. Kohn, A. Nerode, J. Remmel, and X. Ge, "Multiple agent hybrid control: Carrier manifolds and chattering approximations to optimal control", In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pp. 4221–4227, Lake Buena Vista, FL, Dec. 1994.
13. W. Kohn, J. James, A. Nerode, and N. DeClaris, "A hybrid systems approach to integration of medical models", In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pp. 4247–4252, Lake Buena Vista, FL, Dec. 1994.
14. R. Brockett, "Language driven hybrid systems", In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pp. 4210–4214, Lake Buena Vista, FL, Dec. 1994.
15. P. J. Ramadge, "On the periodicity of symbolic observations of piecewise smooth discrete-time systems", *IEEE Transactions on Automatic Control*, vol. 35, no. 7, pp. 807–812, July 1990.
16. 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*, pp. 455–464, University of Illinois at Urbana-Champaign, Oct. 1991.
17. 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*, pp. 4215–4220, Lake Buena Vista, FL, Dec. 1994.

18. A. Gollu and P. Varaiya. "Hybrid dynamical systems". In *Proceedings of the 28th Conference on Decision and Control*, pp. 2708–2712, Tampa, FL, Dec. 1989.
19. A. Deshpande and P. Varaiya. "Viable control of hybrid systems". In the Ph.D. Dissertation of the first author, June 1994. ftp: eclair.eecs.berkeley.edu.
20. M. Tittus and B. Egardt, "Control-law synthesis for linear hybrid systems". In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pp. 961–966, Lake Buena Vista, FL, Dec. 1994.
21. B. Lennartson, B. Egardt, and M. Tittus. "Hybrid systems in process control". In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pp. 3587–3595, Lake Buena Vista, FL, Dec. 1994.
22. R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors. *Hybrid Systems*, vol. 736 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.
23. 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, Nov. 1985.
24. P. Ramadge and W. M. Wonham. "Supervisory control of a class of discrete event processes". *SIAM Journal of Control and Optimization*, vol. 25, no. 1, pp. 206–230, Jan. 1987.
25. P. Ramadge and W. M. Wonham. "The control of discrete event systems". *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–89, Jan. 1989.
26. 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, Nov. 1983.
27. W. M. Wonham and P. J. Wonham, "On the supremal controllable sublanguage of a given language", *SIAM Journal of Control and Optimization*, vol. 25, no. 3, pp. 637–659, May 1987.
28. X. Yang, P. J. Antsaklis, and M. D. Lemmon, "On the supremal controllable sublanguage in the discrete event model of nondeterministic hybrid control systems". Technical Report of the ISIS Group ISIS-94-004, University of Notre Dame, Notre Dame, IN, March 1994.
29. H. Nijmeijer and A. J. van der Schaft, *Nonlinear Dynamical Control Systems*. Springer-Verlag, New York, 1990.
30. R. Miller and A. Michel. *Ordinary Differential Equations*. Academic Press, New York, NY, 1982.
31. A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, Berlin, 2 edition, 1989.
32. P. J. Antsaklis. "On Intelligent Control: Report of the IEEE CSS Task Force on Intelligent Control". Technical Report of the ISIS Group (Interdisciplinary Studies of Intelligent Systems) ISIS-94-001, University of Notre Dame, January 1994.

176

176
Box 4/95