Supervisory Control of Hybrid Systems

XENOFON D. KOUTSOUKOS, PANOS J. ANTSAKLIS, FELLOW, IEEE, JAMES A. STIVER, AND MICHAEL D. LEMMON, MEMBER, IEEE

Invited Paper

In this paper, the supervisory control of hybrid systems is introduced and discussed at length. Such control systems typically arise in the computer control of continuous processes, for example, in manufacturing and chemical processes, in transportation systems, and in communication networks. A functional architecture of hybrid control systems consisting of a continuous plant, a discrete-event controller, and an interface is used to introduce and describe analysis and synthesis concepts and approaches. Our approach highlights the interaction between the continuous and discrete dynamics, which is the cornerstone of any hybrid system study. Discrete abstractions are used to approximate the continuous plant. Properties of the discrete abstractions to be appropriate representations of the continuous plant are presented, and important concepts such as determinism and controllability are discussed. Supervisory control design methodologies are presented to satisfy control specifications described by formal languages. Several examples are used throughout the paper to illustrate our approach.

Keywords*—Controllability, hybrid systems, language theoretic framework, natural invariants, nondeterministic finite automata, supervisory control, transition stability.*

I. INTRODUCTION

Hybrid dynamical systems are characterized by interacting continuous and discrete dynamics (see, e.g., [11] and [10] and the guest editor's introduction in this special issue). Hybrid control systems typically arise from computer-aided control of continuous processes, e.g., in manufacturing and chemical processes, in transportation systems, and in communication networks. The study of hybrid control systems is essential in designing supervisory controllers for continuous systems, and it is central in designing intelligent control systems with a high degree of autonomy (see, e.g., [14], [12], [13], [3], and [4]). Hybrid system analysis and controller synthesis techniques may provide efficient approaches for the design and verification of complex engineering systems.

In this paper, the supervisory control of hybrid systems is introduced and discussed at length. The simplest example of a supervisory controller is perhaps the thermostat that regulates the temperature in our homes. The controller is really a switching mechanism that interacts with the continuous dynamics of the furnace to counteract the heat losses, so as to keep the temperature within a desirable range. The thermostat is further discussed in Section II-B. The type of supervisory control problems that is of interest here arises whenever a continuous system is to be controlled by a discrete process such as a switching mechanism or a digital computer program. Controllers of this type are being used to control many physical processes. Examples include the operation of chemical plants—start-up and shutdown procedures, fail-safe mechanisms, and control during regular operation by switching to different operating modes. They are also used to coordinate multiple interacting robots, to control manufacturing processes, and to coordinate the operation of autonomous vehicles.

A convenient way to represent such hybrid control systems is shown in Fig. 1. The continuous process to be controlled, together with any continuous controllers, is identified as the "Plant" and is typically described by differential/difference equations. The "Controller" includes a discrete decision process that is typically a discrete-event system described, e.g., by a finite automaton. The "Interface" makes it possible for these different processes to communicate with each other. This control framework is quite flexible and can describe modern engineering systems where a computer process is used to control and coordinate several physical processes over a computer network. It can also describe a switching control system where a continuous plant is controlled by different continuous controllers over a number of operating regions. It should be noted that the

Manuscript received October 29, 1999; revised March 28, 2000. This work was supported in part by the National Science Foundation under Grant ECS95–31485 and by the Army Research Office under Grant DAAG55–98–1–0199.

X. D. Koutsoukos, P. J. Antsaklis, and M. D. Lemmon are with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: xkoutsou@nd.edu; antsaklis.1@nd.edu; lemmon@maddog.ee.nd.edu).

J. A. Stiver is with Harris Corporation, Melbourne, FL 32902 USA (e-mail: jstiver@harris.com).

Publisher Item Identifier S 0018-9219(00)06463-X.

representation of Fig. 1 is a functional one that is convenient for the mathematical study of such hybrid systems. In certain systems, Fig. 1 represents the actual control architecture, and the controller, interface, and plant can be identified in a natural way. In other systems, however, such separation is not so clear, and in that case, this representation is used primarily to study the system and identify its properties rather than to implement any control strategies. There are hybrid systems for which it may not be possible to separate the continuous from the discrete part in a natural way. However, a description as in Fig. 1 may be useful in analysis, and it may lead to a better understanding of important properties related to the interface of continuous and discrete dynamics.

Using the supervisory control framework as illustrated in Fig. 1, it has been possible to design discrete-event controllers for hybrid systems. These controllers are based on discrete abstractions of the continuous dynamics. Applications have been primarily in the chemical process industry and include control of distillation columns and batch processes. The appeal of this approach is that it generalizes well-known concepts from digital control design. Note that this connection to digital control is described in Section IV-C. One of the main characteristics of the supervisory control approach has been the emphasis and explicit identification of the interface issues between the continuous and discrete dynamics. These interface issues are the cornerstone of any hybrid system study. The decision of how detailed a discrete abstraction should be is related to the fundamental question of how much information is needed by the controller to attain particular control goals. Note that the discrete-event plant models derived via discrete abstractions are typically nondeterministic, and this leads to significant difficulties in controlling hybrid systems. Those issues that were originally identified in a supervisory control framework have affected many different approaches to hybrid control systems that are discussed in this issue.

The types of problems that have been addressed by existing methods in supervisory control of hybrid systems are those with control specifications that can be described by formal languages accepted by finite automata, which approximate the continuous plant. Examples include safety problems, where the controller guarantees that the plant will not enter an unsafe operating region, e.g., guaranteeing that two interacting robots will not collide. Another example is reachability problems, where the controller drives the plant from an initial operating region or state to a desired one; this is the case, e.g., in the start-up procedure in a chemical plant. Note that this supervisory control framework is based on a logical approach for hybrid systems and does not directly address time issues. The emphasis is on the logical ordering of sequences of events, e.g., in a start-up procedure rather than on time constraints for an event to take place within a particular time window. However, certain time issues can be addressed using this approach by including a clock to be part of the continuous dynamics of the plant. This is done to describe digital control in a hybrid system framework, and details can be found in Section IV-C.

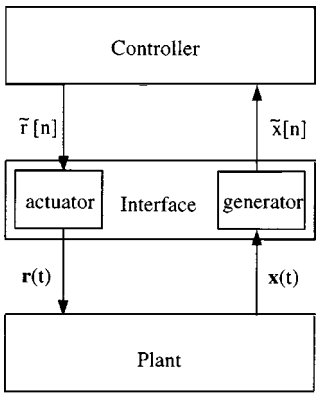

Fig. 1. Hybrid control system.

It should be noted that the supervisory control approach does not intend to address problems that involve continuous controls, as it has been assumed that any continuous control action has already been considered and is included in the plant of Fig. 1. It should also be noted that this approach does not address jumps in the continuous state that may occur when certain state variables are discontinuously reset. However, it includes switches where, typically, there are no jumps in the state value. There is, however, a large class of important engineering problems that can be solved using the supervisory control approach described in this paper.

In general, a great amount of research work has already been done in the hybrid systems area during the past decade (see, e.g., [6], [2], [7], [5], [11], [10]), and in the many papers that have appeared in major decision and control conference proceedings, as well as the papers in this special issue. Several different mathematical paradigms have been used for modeling hybrid systems. In broad terms, the models differ with respect to the emphasis on or the complexity of the continuous and discrete dynamics. On one end of the spectrum there are equational models that include discontinuities such as switchings and jumps. Typically, these models are used in order to extend ideas from continuous systems and study traditional control problems such as stability, robustness, and optimal control. Some examples of such systems are discussed in [45], [42], [70], and in this special issue. Switched systems consisting of a family of continuous subsystems and a rule that orchestrates the switching between them is an important class of hybrid dynamical systems in this category (see [38] for a survey of recent developments regarding the stability and design of switched systems). On the other end of the spectrum, there are computer science models that are mainly used to describe the behavior of real-time embedded systems. Models of this type are collectively known as hybrid automata [1], [41], [24]. Based on the type of the continuous dynamics they can represent, there are several variations of hybrid automata such as timed, rectangular, linear, and nonlinear hybrid automata. Hybrid system models that are based on Petri nets instead of finite automata have also been proposed [23], [31], [21]. There are additional models spanning the rest of the spectrum that combine concepts from continuous control and discrete event systems. A survey of different

models and methodologies can be found in [9]. Finally, a unified hybrid system model and a formal comparison between models has been discussed in [17].

The work described in this paper is based on and represents extensions of developments reported in [60], [15], [63], [62], [64], and [65]. Similar approaches based on approximations of the continuous plant model by a discrete event system have also been proposed in [47], [56], [20], and [40]. Some important characteristics of these approaches are briefly discussed in Section II-D. A related area is hierarchical control of hybrid systems where a hierarchical structure is imposed on the system architecture to reduce complexity [18], [52]. The design is based on the notion of hierarchical consistency, which ensures that the control objectives are satisfied by the precise models at the lower levels, although the design has been carried out at the higher levels of the hierarchy using coarse models. Discrete abstractions of continuous systems in finite quotient spaces have also been used to study formal verification and decidability of hybrid systems in [33] and [34].

This paper is organized as follows. Modeling of hybrid systems in the proposed supervisory control framework is described in Section II. The properties of the discrete abstractions of the continuous plant are discussed in Section III. Methodologies for the design of the interface are presented in Section IV. More specifically, a methodology to design the partition of the continuous state space based on the natural invariants of the plant is presented in Section IV-A. Stability of transitions in the discrete event system (DES) plant with respect to variations in the initial state of the continuous plant is discussed in Section IV-B. The relation of the supervisory control framework of hybrid systems with digital control design is also described in Section IV. A language theoretic framework is used to describe performance specifications for hybrid systems and the problem of supervisory control design for hybrid systems is formulated in Section V-A. The notion of controllability for the languages generated by the abstracting DES plant model is defined and a methodology for supervisory control design is presented in Section V-B. The design methodology is illustrated using several examples, including a distillation column and a robotic manufacturing system. Finally, some concluding remarks are included in Section VI.

II. Modeling Hybrid Systems

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 in a feedback configuration. It is generally assumed that the dynamic behavior of the plant is governed by a set of known nonlinear ordinary differential equations. In the model shown in Fig. 1, the plant contains all continuous components 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, and so on. The controller is an event-driven, asynchronous 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.

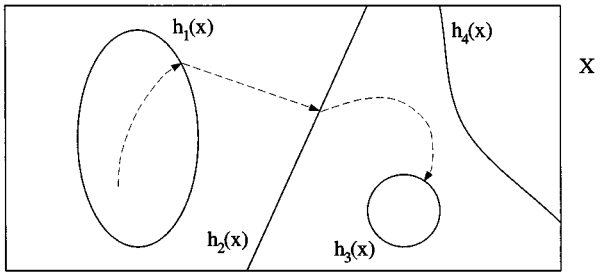

Fig. 2. Partition of the continuous state space.

The interface plays a key role in determining the dynamic behavior of the hybrid control system. Here, the interface has been chosen to be simply a partitioning of the state space (see Fig. 2), 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 as shown in Fig. 3, not unlike the output of a zero-order hold in a digital control system. Note that signals such as ramps, sinusoids, etc., can be generated if desired by including an appropriate continuous system at (the input of) the plant. The simple interface used in the model allows the analysis of properties such as controllability, stability, and determinism. More important, it enables the development of controller design methodologies. The simplicity of the interface with the resulting benefits in identifying central issues and concepts in hybrid control systems is perhaps the main characteristic of the approach.

A. Hybrid System Model

In this section, we present our mathematical model for supervisory hybrid control systems shown in Fig. 1. The description of the interface is done very carefully in order to take into consideration important phenomena such as chattering, delays in switching, etc., so Sections II-A and II-B are rather technical by necessity. The modeling approach is illustrated via two simple examples in this section. A distillation column and a robotic manufacturing example may be found in Section V.

1) Continuous Plant: The plant is in general a nonlinear, time-invariant system represented by a set of ordinary differential equations

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

where $\mathbf{x}(t) \in \mathbf{X}$ and $\mathbf{r}(t) \in \mathbf{R}$ are the state and input vectors, respectively, and $\mathbf{X} \subset \mathbb{R}^n$, $\mathbf{R} \subset \mathbb{R}^m$, with $t \in [a, b)$ some time interval. For each fixed $\mathbf{r}(t) \in \mathbf{R}$, the function $f(\cdot, r(t))$: $\mathbf{X} \to \mathbf{X}$ 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.

The representation of the plant is quite general and can be used to describe a large class of systems that includes

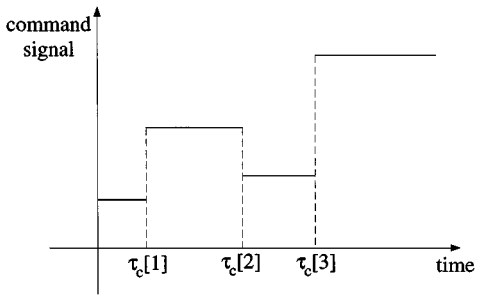

Fig. 3. Command signal issued by the interface.

time-invariant nonlinear systems and switching systems. For example, a linear switching system consisting of m subsystems can be described by

$$\dot{\mathbf{x}}(t) = \sum_{i=1}^{m} r_i(t) A_i \mathbf{x}(t)$$

where $\mathbf{x} \in \mathbb{R}^n$, r_i: $\mathbb{R} \rightarrow \{0, 1\}$, $\sum_{i=1}^{m} r_i(t) = 1$, and $\forall t \in \mathbb{R}$, where $r_i(\tau_1) = 1$ implies that the system A_i is actuated at time τ_1.

2) Controller: The controller (or supervisor) is a discrete event system that is modeled as a deterministic finite automaton [26]. This automaton is specified by $S = (\tilde{S}, \tilde{X}, \tilde{R}, \delta, \phi)$, where $\tilde{S}$ is the set of states, $\tilde{X}$ is the set of *plant symbols*, $\tilde{R}$ is the set of *controller symbols*, δ: $\tilde{S} \times \tilde{X} \rightarrow \tilde{S}$ is the state transition function, and ϕ: $\tilde{S} \rightarrow \tilde{R}$ 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, also called the *alphabet*, 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 $\tilde{X}$.

3) Interface: The controller and plant cannot communicate directly in a hybrid control system because each utilizes different types of signals. Thus, an interface is required that 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 the actuator. The generator issues symbols to the controller and plays the role of a sampler together with a quantizer of the signals analogous to an analog-to-digital converter (sampler) in a digital control system. The actuator injects the appropriate control signal into the plant and is analogous to a digital-to-analog 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.

The *generator* is the subsystem of the interface that 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 that separates the plant's state space into two disjoint sets. 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, $\{h_i$: $\mathbb{R}^n \rightarrow \mathbb{R}, i \in I\}$, 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, $\mathcal{N}(h_i) = \{\xi \in \mathbb{R}^n$: $h_i(\xi) = 0\}$, forms an $n-1$-dimensional smooth hypersurface separating the state space.

Let the sequence of plant events be denoted by e, where $e[n] = i$ means the nth 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 nth plant event and $\tau_e[0] = 0$. A simple way of expressing the conditions for the generation of plant events is by $h_i(\mathbf{x}(t)) = 0$, $(d/dt)h_i(\mathbf{x}(t)) \neq 0$. In this case, an assumption is made that the derivative is nonzero, i.e., $(d/dt)h_i(\mathbf{x}(t)) \neq 0$ at the crossing. Note, however, that these conditions do not take into account the case where the crossing occurs exactly at a point where $(d/dt)h_i(\mathbf{x}(t)) = 0$. In this case, one must use the following conditions:

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

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, (5), contains three conditions: 1) at the time of the plant event the plant state lies on the triggering hypersurface; 2) immediately after the event the plant state lies on the negative (positive) side of the triggering hypersurface; and 3) prior to reaching the triggering hypersurface, the plant state lies on the positive (negative) side. The fourth condition, (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, i.e., the value of i. The generation of plant events is illustrated in Fig. 4.

A plant event will only cause a plant symbol to be generated if the hypersurface is crossed in a defined 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. 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 α_i: $\mathcal{N}(h_i) \rightarrow \tilde{X}$. The sequence of plant symbols can now be defined as

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

where i identifies the hypersurface that was crossed. Alternatively, one could select the interface to generate information bearing symbols when crossed in either direction.

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

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

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. $\tau_c[n]$ is the time of the nth control symbol, which is based on the sequence of plant symbol instants, defined in (5), 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 time 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.

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 nonzero delay is not used, it is possible that the differential equation (1) will exhibit solutions that switch between different control policies infinitely many times in a finite time interval. Such behavior does not occur in physical systems. Systems capable of exhibiting such behavior are referred to as *Zeno* systems. In supervisory hybrid control systems, we want the systems to be non-Zeno. 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 units of time 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.

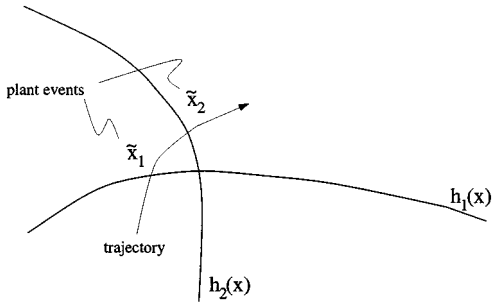

Fig. 4. Generation of plant events.

B. 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 (see Fig. 5). 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 that 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 that is controlled. In the following, we present a simple example of a thermostat/furnace in order to illustrate the approximation of a continuous plant by a DES plant model. A methodology for the extraction of the DES plant is described after the thermostat/furnace example and is illustrated with additional examples.

1) Example—Thermostat/Furnace System: The hybrid system in this example consists of a typical thermostat and furnace. Assuming the thermostat is set at 70°F, the system behaves as follows. If the room temperature falls below 70°, the furnace starts and remains on until the room temperature reaches 75°. At 75°, the furnace shuts off. For simplicity, we will assume that when the furnace is on it produces a constant amount of heat per unit time.

The plant in the thermostat/furnace hybrid control system is made up of the furnace and room. It can be modeled with the following differential equation:

$$\dot{\mathbf{x}} = 0.0042(T_0 - \mathbf{x}) + 0.1\mathbf{r} \tag{10}$$

where the plant state $\mathbf{x}$ is the temperature of the room in degrees Fahrenheit, the input $\mathbf{r}$ is the voltage on the furnace control circuit, and T_0 is the outside temperature. The units for time are minutes. The constants used in this example correspond to particular given data. This model of the furnace is certainly a simplification, but it is adequate for this example.

The thermostat partitions the state space of the plant with two hypersurfaces as follows:

$$h_1(\mathbf{x}) = \mathbf{x} - 75 \tag{11}$$

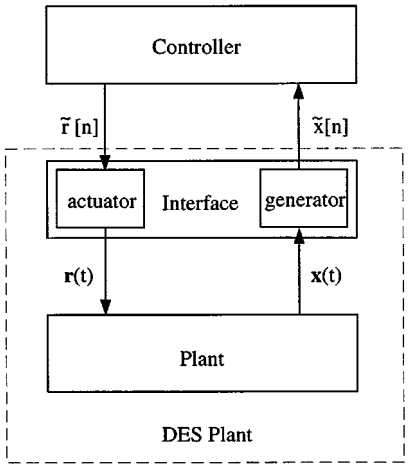

Fig. 5. DES plant model.

$$h_2(\mathbf{x}) = 70 - \mathbf{x}. \tag{12}$$

The first hypersurface detects when the state exceeds 75, and the second detects when the state falls below 70. The associated functions α_1 and α_2 are very simple in this case:

$$\alpha_i(\mathbf{x}) = \tilde{x}_i. \tag{13}$$

So there are two plant symbols: $\tilde{x}_1$ and $\tilde{x}_2$.

The DES controller is shown in Fig. 6. The output function of the controller is defined as

$$\phi(\tilde{s}_1) = \tilde{r}_1 \Leftrightarrow \text{off} \tag{14}$$

$$\phi(\tilde{s}_2) = \tilde{r}_2 \Leftrightarrow \text{on} \tag{15}$$

and the actuator operates as

$$\gamma(\tilde{r}_1) = 0 \tag{16}$$

$$\gamma(\tilde{r}_2) = 12. \tag{17}$$

The thermostat/heater example has a simple DES plant model, which is useful to illustrate how these models work. Fig. 7 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. Note 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's doing nothing, but it serves to keep all the symbolic sequences, $\tilde{s}$, $\tilde{p}$, etc., in phase with each other.

C. Extraction of the DES Plant Model

The *DES plant model* is a nondeterministic finite automaton, which is represented mathematically by $G = (\tilde{P}, \tilde{X}, \tilde{R}, \psi, \lambda)$. $\tilde{P}$ is the set of states, $\tilde{X}$ is the set of plant symbols, and $\tilde{R}$ is the set of control symbols. ψ: $\tilde{P} \times \tilde{R} \rightarrow 2^{\tilde{P}}$ is the state transition function. For a given DES plant state and a given control symbol, it specifies a set of possible new DES plant states. The output function, λ: $\tilde{P} \times \tilde{P} \rightarrow 2^{\tilde{X}}$, maps the previous and current states to a set of plant symbols. Note that the DES plant model G is a nondeterministic automaton and the state transition function

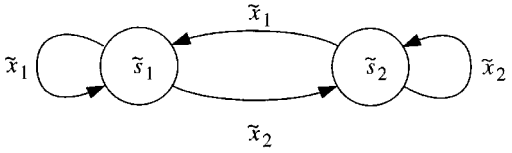

Fig. 6. Controller for the thermostat/furnace system.

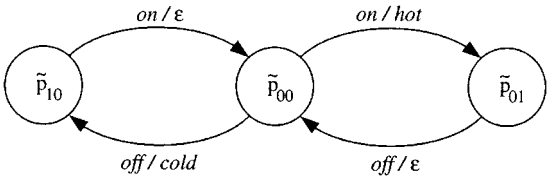

Fig. 7. DES plant for the thermostat/furnace system.

is defined as a mapping from $\tilde{P} \times \tilde{R}$ to the power set $2^{\tilde{P}}$ of $\tilde{P}$, since for a given state and input symbol the next state is not uniquely defined. The output function is defined similarly.

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 $\{\xi \in \mathbb{R}^n: h_i(\xi) \neq 0, i \in I\}$ as follows:

$$\xi_1 \equiv_p \xi_2 \text{ iff } h_i(\xi_1)h_i(\xi_2) > 0, \qquad \forall i \in I. \tag{18}$$

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 ith element of b and $\tilde{p}_b$ is associated with the set $\{\xi \in \mathbb{R}^n: b_i = 1 \Leftrightarrow h_i(\xi) < 0\}$. The equivalence relation is not defined for states that 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.

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

$$\tilde{P} = \{\xi \in \mathbb{R}^n: h_i(\xi) \neq 0, i \in I\} / \equiv_p . \tag{19}$$

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

$$\tilde{p}_b = \{\xi \in \mathbb{R}^n: b_i = 0 \Rightarrow h_i(\xi) > 0 \text{ and } b_i = 1 \Rightarrow h_i(\xi) < 0\}. \tag{20}$$

The *DES plant state* $\tilde{p}[n]$ corresponds to the most recently entered region of the plant state space and is defined as follows:

$$\tilde{p}[n] = \tilde{p}_b \tag{21}$$

where

$$\lim_{\epsilon \rightarrow 0^+} \mathbf{x}(\tau_e[n] + \epsilon) \in \tilde{p}_b. \tag{22}$$

The limit must be used here because at exactly $\tau_e[n]$, the continuous state will be on a boundary. Note that no plant symbol is generated if the state trajectory moves along the boundaries. Note that problems associated with the boundaries of regions that partition the state space and the continuity of

the analog-to-digital maps defined from these partitions have been studied at length in [47] and [16].

The state of the DES plant model 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.

Note that 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 $j \in I$

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

where b_i and c_i are the binary vectors associated with $\tilde{p}_b$ and $\tilde{p}_c$, respectively, and $\overline{\tilde{p}_b}$ 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 for the hypersurfaces defined by h_i, we have that $(d/dt)h_i(\mathbf{x}(t)) \neq 0$ [see comments following (5)].

Proposition 1: Given a hybrid control system, described by (1)–(9), with f and h_i continuously differentiable, then $\tilde{p}_c \in \psi(\tilde{p}_b, \tilde{r}_k)$ iff $i \in I$ and $\xi \in \mathcal{N}(h_i)$ exist such that following conditions are satisfied:

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

Further, if $\alpha_i(\xi) = \tilde{x}_i$, we also have that $\tilde{x}_i \in \lambda(\tilde{p}_b, \tilde{p}_c)$.

The usefulness of this proposition is that it allows the extraction of a DES automaton model of the continuous plant and interface as illustrated in Fig. 8. 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 that 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.

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

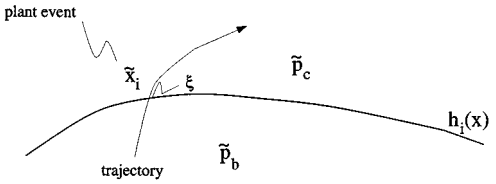

Fig. 8. Extraction of the DES plant model.

a control policy permits a given behavior in the DES plant, there is no guarantee that the behavior will occur in the actual system. This phenomenon is due to the nondeterminism in the DES plant model and is further discussed in Section III.

1) Example—Double Integrator: The system consists of a double integrator plant, which is controlled by a discrete event system. A double integrator plant may used as a simple example for satellite control, modeling the relation between the angular position and velocity and the reaction jets. The plant is given by the differential equation

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{r}(t). \tag{23}$$

The generator recognizes four plant events:

$$h_1(\mathbf{x}) = x_1 \quad h_2(\mathbf{x}) = -x_1 \tag{24}$$

$$h_3(\mathbf{x}) = x_2 \quad h_4(\mathbf{x}) = -x_2. \tag{25}$$

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

$$\alpha_1(\mathbf{x}) = \tilde{x}_1 \quad \alpha_2(\mathbf{x}) = \tilde{x}_1 \tag{26}$$

$$\alpha_3(\mathbf{x}) = \tilde{x}_2 \quad \alpha_4(\mathbf{x}) = \tilde{x}_2. \tag{27}$$

Note 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 that was crossed, as illustrated in Fig. 9.

We assume that there are two controller symbols, $\tilde{R} = \{\tilde{r}_1, \tilde{r}_2\}$, so the actuator provides two possible inputs to the plant

$$\gamma(\tilde{r}) = \begin{cases} -1, & \text{if } \tilde{r} = \tilde{r}_1 \\ 1, & \text{if } \tilde{r} = \tilde{r}_2. \end{cases} \tag{28}$$

Using Proposition 1, we extract the DES plant for this system, which is shown in Fig. 10. 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. Proposition 1 is also 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 $\{\tilde{x}_1\} = \lambda(\tilde{p}_9, \tilde{p}_5)$. A controller that drives the state of the double integrator in clockwise circles is designed in Section V-B.

D. Related Models

There are supervisory approaches to hybrid control in the literature that are related to the approach described above.

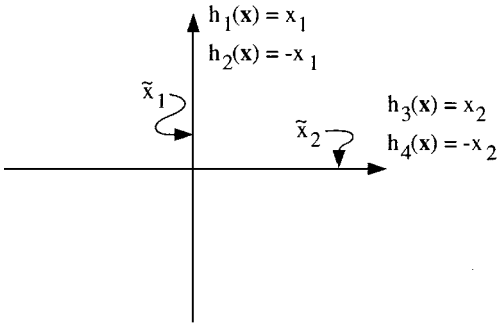

Fig. 9. Generator for the double integrator example.

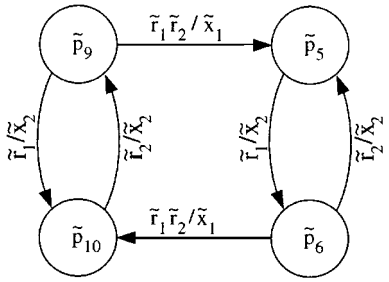

Fig. 10. DES plant for the double integrator.

One is the Nerode–Kohn approach to hybrid control systems that is based on automata theory. The model and related work can be found in [47] and [48]. This approach is similar to the one described here. The main differences are in the way plant events are generated and the explicit introduction of time as a parameter of the control signals. The plant and controller are represented by interacting nondeterministic sequential automata. The *control automaton* is a nondeterministic sequential automaton, which has as an infinite input alphabet consisting of sensor measurements and an infinite output alphabet consisting of control laws. The plant is modeled as a nondeterministic sequential automaton, which represents a system given by time-variant ordinary differential equations. The input alphabet of the plant is the infinite alphabet of control laws and the output alphabet of the plant is the infinite alphabet of sensor measurements. The automata interact at times t_i, where the interval between successive times, $\Delta_i = t_{i+1} - t_i$, can vary. To facilitate analysis, several specializations of the above model are imposed. First, the time between successive plant–controller interactions is required to be constant, i.e., $\Delta_i = \Delta$. Second, the plant is assumed to be time-invariant. Third, the control automaton is decomposed into a finite automaton equipped with an interface. The model is shown in Fig. 11. Here, the control automaton has been separated into three parts: an analog-to-digital converter, an internal control automaton, and a digital-to-analog converter. The internal control automaton is a finite automaton. That means the analog-to-digital converter converts real-number sensor measurements to one of a finite number of input symbols. The digital-to-analog converter maps output symbols to a control law and a duration for which the control law should remain in effect.

The plant automaton represents an underlying system defined by ordinary differential equations. The plant evolves

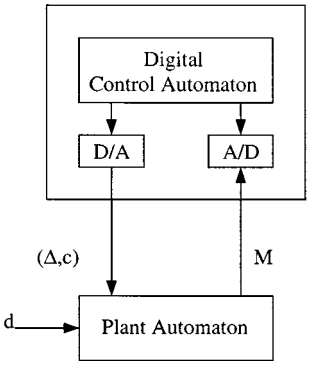

Fig. 11. Hybrid system model.

over the intervals of time between successive plant–controller interactions according to the following equation for the interval $[r, s]$:

$$x'(t) = f(x(t), t, c(t-r), d(t-r)) \tag{29}$$

where $t \in [r, s]$, and $c(t)$ and $d(t)$ are the control function and disturbance function, each with domain $[0, \infty]$. A hybrid system is formed by connecting the plant automaton to the control automaton as follows. The output of the controller consists of pairs (Δ, c) specifying a duration and a control law for that duration. This makes up the corresponding input to the plant automaton. The output of the plant automaton is connected to the input of the controller automaton.

The next two approaches are motivated by chemical process control applications and are directly related to our work. Qualitative modeling of linear systems with quantized state measurements has also been considered by Lunze [39]. The main characteristic of the model is that the partition is predetermined by sensor quantizations. Discrete-time systems of the form

$$x(k+1) = Ax(k), \quad x(0) = x_0 \tag{30}$$

are considered. It is assumed that only quantized state measurements $[x(k)]$, which represent the qualitative states, are available. The state variables x_i are quantized independently from each other with resolution q_{x_i}, according to the equation

$$[x_i(k)] = s_i(k) \tag{31}$$

where $s_i(k)$ satisfies the relation

$$\left(\left(s_i(k) - \tfrac{1}{2}\right) q_{x_i}\right) \le x_i(k) \le \left(\left(s_i(k) + \tfrac{1}{2}\right) q_{x_i}\right). \tag{32}$$

Graphically, the state space $\mathbb{R}^n$ is partitioned into the rectangular blocks whose edges have the direction of the axes and length q_{x_i} as illustrated in Fig. 12. A nondeterministic automaton is used to describe the qualitative behavior similarly to the hybrid control system model presented earlier in the section. The system is also approximated by a stochastic automaton with the assumption that the initial state x_0 is uniformly distributed over the corresponding rectangular region of the state space. More recently, Lunze has considered more

general grids in order to derive deterministic discrete-event representations for linear continuous systems [40].

A similar model where the plant state evolves in $\mathbb{R}^n$ and the control input and measurement signals are symbolic is considered by Raisch [55], [56]. In addition, the state is affected by real-valued unknown but bounded disturbances. More specifically, the plant is modeled as a nonlinear discrete-time system with state transition function

$$x(t_{k+1}) = f(x(t_k), w(t_k), u_d(t_k)) \tag{33}$$

$$y_d(t_k) = q_y(x(t_k)) \tag{34}$$

where $x(t_k) \in \mathbb{R}^n$ is the state at time t_k and $w(t_k) \in \mathbb{R}^r$ is the disturbance. The control and measurement symbols are $u_d(t_k)$ and $y_d(t_k)$, respectively, and q_y is the measurement map that converts the state to the corresponding measurement symbol. A requirement imposed on f is that it can be solved with respect to the first argument, i.e., $x(t_k) = \tilde{f}(x(t_{k+1}), w(t_k), u_d(t_k))$. A contribution of this modeling framework is the formulation of a hierarchy of discrete abstractions with respect to the approximation accuracy. The approximation accuracy of the DES plant model can be improved by including past measurements and control signals. The state of the DES plant model (approximating automaton) is defined by

$$x_d(t_k) = \begin{cases} ([y_d(t_k), \cdots, y_d(t_0)], \\ \quad [u_d(t_k), \cdots, u_d(t_0)]), & \text{if } k = 0, 1, \\ & \cdots, v-1 \\ ([y_d(t_k), \cdots, y_d(t_{k-v})], \\ \quad [u_d(t_k), \cdots, u_d(t_{k-v})]), & \text{if } k \geq v. \end{cases} \tag{35}$$

The approximation accuracy depends on the length v of the measurement and control signals and can be adjusted to the specification requirements. An important characteristic of the approach is that the time needed for a transition, which is $t_{i+1} - t_i$, is retained in the discrete approximation. The approach has been applied to examples from process control in [54].

III. Properties of the DES Plant Model

In this section, we discuss properties for the DES plant model to be a useful representation of the continuous system.

A. DES Plant Model as an Approximation

The DES plant model is an approximation of the actual system and its behavior is an abstraction of the system's behavior. Specifically, the state of the DES plant model is an approximation of the state of the continuous plant. As a result, the future behavior of the actual continuous system 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. Thus, we construct a conservative approximation of the behavior of the continuous plant that includes the behavior of the plant, and can be realized by a finite-state machine. Note that different continuous processes may be represented by a DES model. From a control point of view, this means if undesirable behaviors can be eliminated from the DES plant (through appropriate control policies), then these behaviors will be eliminated from the actual system. On the other hand, just because a control policy permits a given behavior in the DES plant, there is no guarantee that the behavior will occur in the actual system.

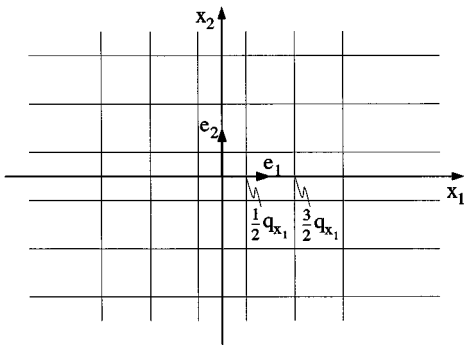

Fig. 12. Partition of the two-dimensional state space into rectangular blocks.

Raisch and coworkers [55], [56] have used a behavioral approach for the representation of dynamical systems in order to formalize the issues related to the approximation accuracy of the discrete abstractions and the effects of the supervisor. We briefly discuss this approach to illustrate the issues related to the approximation of the plant by a DES plant model.

A *dynamical system* Σ can be described as a triple (T, W, B) with $T \subseteq \mathbb{R}$ the *time axis*, W the *signal space*, and $B \subset W^T$ (the set of all functions $f\colon T \to W$) the *behavior* [67]. In our modeling formalism, the behavior of the DES plant model $B_d \subset (\tilde{X} \times \tilde{R})^T$ consists of all the pairs of plant and control symbols that the nondeterministic automaton $(\tilde{P}, \tilde{X}, \tilde{R}, \psi, \lambda)$ can generate. The time axis T represents here the occurrences of events as defined in Section II. A necessary condition for the DES plant model to be a valid approximation of the continuous plant is that the behavior of the continuous plant model B_c is contained in the behavior of the DES plant model, i.e., $B_c \subseteq B_d$.

Since the controller is represented by the automaton $(\tilde{S}, \tilde{X}, \tilde{R}, \delta, \phi)$, its behavior can also be described as $B_s \subset (\tilde{X} \times \tilde{R})^T$. The main objective of the controller is to restrict the behavior of the DES plant model in order to specify the control specifications. The specifications can be described by a behavior $B_{\text{spec}} \subset (\tilde{X} \times \tilde{R})^T$. Supervisory control of hybrid systems is based on the fact that if undesirable behaviors can be eliminated from the DES plant, then these behaviors can likewise be eliminated from the actual system. This is described formally by the relation

$$B_d \cap B_s \subseteq B_{\text{spec}} \Rightarrow B_c \cap B_s \subseteq B_{\text{spec}} \tag{36}$$

and is depicted in Fig. 13. The challenge is to find a discrete abstraction with behavior B_d, which is an approximation of the behavior B_c of the continuous system and for which is possible to design a supervisor in order to guarantee that the behavior of the closed-loop system satisfies the specifications B_{spec}.

A more accurate approximation of the plant's behavior can be obtained by considering a finer partitioning of the state

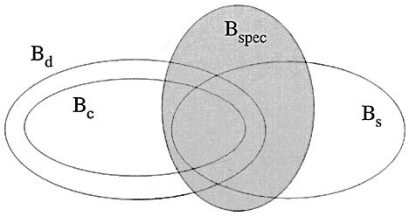

Fig. 13. DES plant model as an approximation.

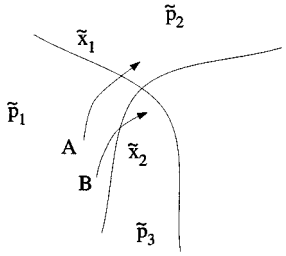

Fig. 14. Nondeterminism of the DES plant model.

space for the extraction of the DES plant. A totally ordered set of discrete abstractions for a given system has been presented in [55].

B. Determinism

An interesting aspect of the DES plant's behavior is that it is distinctly nondeterministic. This fact is illustrated in Fig. 14. The figure shows two different trajectories generated by the same control symbol. Both trajectories originate in the same DES plant state $\tilde{p}_1$. Fig. 14 shows that for a given control symbol, there are at least two possible DES plant states that can be reached from $\tilde{p}_1$. Nondeterminism in the DES plant therefore arises due to uncertainty in the DES states reached under a controlled transition. Transitions within a DES plant will usually be nondeterministic unless the boundaries of the partition sets are invariant manifolds with respect to the vector fields that describe the continuous plant. More details about such partitions are presented in Section IV. The problem of obtaining deterministic discrete-event representations for specific classes of hybrid systems has been considered in [22] and [40].

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 state transitions exists, the sequence of inputs that achieves it may also permit other sequences of state transitions. Unfortunately, given a continuous-time plant, it may be difficult or even impossible to design an interface that leads to a DES plant model that is deterministic. Fortunately, it is not generally necessary to have a deterministic DES plant model in order to control it. The supervisory control problem for hybrid systems can be formulated and solved when the DES plant model is nondeterministic.

IV. Interface

The interface plays a key role in determining the dynamic behavior of the hybrid control system. Many times, the partition of the state space (generator in the interface) is determined by physical constraints and it is fixed and given. Here, we assume that we can select the partition and we focus on specific problems for the interface design. First, we present a methodology to design the partition of the continuous state space based on the natural invariants of the plant. Next, we study stability of transitions in the DES plant with respect to variations in the initial state of the continuous plant. Finally, we present an alternative to the usual quantization technique of digital control based on the interface of hybrid control systems.

A. Generator Design

A methodology is presented [63], [64] to design the interface for a plant described by

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{r}(t)) \tag{37}$$

where certain smoothness assumptions apply. The approach is based on the natural invariants of (37). We assume that the plant is given, the set of available control policies is given, and the control goals are specified as follows. Each control goal for the system is given as a starting set and a target set, each of which is an open subset of the plant state space. To realize the goal, the controller must be able to drive the plant state from anywhere in the starting set to somewhere in the target set using the available control policies. Generally, a system will have multiple control goals.

We propose the following solution to this interface design problem. For a given target region, identify the states that can be driven to that region by the application of a single control policy. If the starting region is contained within this set of states, the control goal is achievable via a single control policy. If not, then this new set of states can be used as a target region and the process can be repeated. 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. Note that similar methods based on backward analysis of the dynamics have been discussed in [53] and [66]. Computational algorithms for the use of the phase-space geometric description of dynamics have been developed in [71].

To describe the regions mentioned above, we use the concept of the flow [49]. Let the flow for the plant (1) be given by F_k: $\mathbf{X} \times \mathbb{R} \rightarrow \mathbf{X}$, where

$$\mathbf{x}(t) = F_k(\mathbf{x}(0), t). \tag{38}$$

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

$F_k^-: \mathbf{X} \to \mathbb{P}(\mathbf{X})$, [$\mathbb{P}(\mathbf{X})$ denotes the power set of X], which are defined as follows:

$$F_k^+(\boldsymbol{\xi}) = \bigcup_{t \geq 0} \{F_k(\boldsymbol{\xi}, t)\} \tag{39}$$

$$F_k^-(\boldsymbol{\xi}) = \bigcup_{t \leq 0} \{F_k(\boldsymbol{\xi}, t)\}. \tag{40}$$

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

$$F_k^+(\mathbf{A}) = \bigcup_{\boldsymbol{\xi} \in \mathbf{A}} \{F_k^+(\boldsymbol{\xi})\} \tag{41}$$

$$F_k^-(\mathbf{A}) = \bigcup_{\boldsymbol{\xi} \in \mathbf{A}} \{F_k^-(\boldsymbol{\xi})\} \tag{42}$$

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 that can be reached with input $\gamma(\tilde{r}_k)$ and an initial state in T. Note that the backward and forward flow functions applied on the set of states correspond to the precondition and postcondition operators used in verification algorithms of hybrid systems [1].

Now, a generator design procedure can be described using the backward flow function. 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 that 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. 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, $\{h_i: \mathbf{X} \to \mathbb{R} | i \in I\}$ and the associated functions, $\{\alpha_i: \mathcal{N}(h_i) \to \tilde{R} | i \in I\}$, described in Section II. These hypersurfaces make up the invariant manifolds and exit boundaries mentioned above, as well as form the boundary for the target region(s). A target region T is specified as

$$T = \{\boldsymbol{\xi} \in \mathbf{X}: \forall i \in I_T,\ h_i(\boldsymbol{\xi}) < 0\} \tag{43}$$

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

$$B = \{\boldsymbol{\xi} \in \mathbf{X}: h_i(\boldsymbol{\xi}) < 0,\ h_e(\boldsymbol{\xi}) > 0,\ \forall i \in I_B\} \tag{44}$$

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. Fig. 15 shows an example of this where $I_T = \{1\}$ and $I_B = \{2, 3\}$. The target region T is surrounded by h_1, and the common flow region lies between h_2 and h_3 above the exit boundary h_e.

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 $\Delta(\mathbf{x})$ is invariant under f if it satisfies

$$[f(\mathbf{x}), \Delta(\mathbf{x})] \subset \Delta(\mathbf{x}) \tag{45}$$

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

$$\delta_1(\mathbf{x}), \delta_2(\mathbf{x}) \in \Delta(\mathbf{x}) \Rightarrow [\delta_1(\mathbf{x}), \delta_2(\mathbf{x})] \in \Delta(\mathbf{x}). \tag{46}$$

Therefore, by identifying the involutive distributions that are invariant under the vector field f, we have identified a set of candidate hypersurfaces. For details of these relationships between vector fields and invariant distributions, see [27]. 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 that are all invariant under f. The set of all invariant hypersurfaces can be found in terms of $n-1$ functionally independent mappings that form the basis for the desired set of functionals $\{h_i: i \in I_B\}$. This basis is obtained by solving the characteristic equation

$$\frac{dx_1}{f_1(\mathbf{x})} = \frac{dx_2}{f_2(\mathbf{x})} =, \cdots, = \frac{dx_n}{f_n(\mathbf{x})} \tag{47}$$

where $f_i(\mathbf{x})$ is the ith element of $f(\mathbf{x})$.

In the following, the approach for the generator design based on natural invariants is illustrated using the double integrator example. More details can be found in [63] and [64], and the application of the method to a simplified model of an autonomous underwater vehicle can be found in [62]. A methodology based on natural invariants has also been used in a Petri net framework for modeling and control of hybrid systems in [31] and [30].

1) Example—Double Integrator: Consider the double integrator example. Suppose we are given the plant

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{r}(t) \tag{48}$$

three available control policies

$$\mathbf{r}(t) \in \{-1, 0, 1\} \tag{49}$$

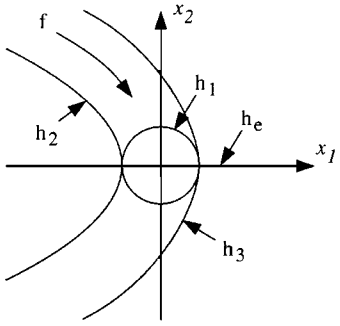

Fig. 15. Target region and invariants.

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

$$T = \{\boldsymbol{\xi} \in \mathbf{X}\colon \xi_1^2 + \xi_2^2 < 1\}. \tag{50}$$

The target set is bounded by the hypersurface given by

$$h_T(\boldsymbol{\xi}) = \xi_1^2 + \xi_2^2 - 1. \tag{51}$$

The first step is to calculate the invariants that can be used to obtain hypersurfaces. There are three families of invariants, one for each of the three control policies, which can be found by solving the characteristic equation (47) for the double integrator

$$\pm\left(\xi_1 + \tfrac{1}{2}\,\xi_2^2 + c_1\right) \tag{52}$$

$$\pm\left(\xi_1 - \tfrac{1}{2}\,\xi_2^2 + c_2\right) \tag{53}$$

$$\pm(\xi_2 + c_3). \tag{54}$$

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

$$h_1(\boldsymbol{\xi}) = h_T(\boldsymbol{\xi}) = \xi_1^2 + \xi_2^2 - 1 \tag{55}$$

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

$$h_2(\boldsymbol{\xi}) = -\xi_1 - \tfrac{1}{2}\,\xi_2^2 - 0.9 \tag{56}$$

$$h_3(\boldsymbol{\xi}) = \xi_1 + \tfrac{1}{2}\,\xi_2^2 - 0.9 \tag{57}$$

and

$$h_e(\boldsymbol{\xi}) = h_4(\boldsymbol{\xi}) = \xi_2 + 0.1. \tag{58}$$

Identify this common flow region as B_1:

$$B_1 = \{\boldsymbol{\xi}\colon h_i(\boldsymbol{\xi}) < 0,\ h_4(\boldsymbol{\xi}) > 0,\ i \in \{2, 3\}\}. \tag{59}$$

Likewise, a common flow region entering T under the third control policy is bounded by

$$h_5(\boldsymbol{\xi}) = -\xi_1 + \tfrac{1}{2}\,\xi_2^2 - 0.9 \tag{60}$$

$$h_6(\boldsymbol{\xi}) = \xi_1 - \tfrac{1}{2}\,\xi_2^2 - 0.9 \tag{61}$$

and

$$h_e(\boldsymbol{\xi}) = h_7(\boldsymbol{\xi}) = -\xi_2 + 0.1. \tag{62}$$

Identify this common flow region as B_2

$$B_2 = \{\boldsymbol{\xi}\colon h_i(\boldsymbol{\xi}) < 0,\ h_7(\boldsymbol{\xi}) > 0,\ i \in \{5, 6\}\}. \tag{63}$$

Fig. 16(a) illustrates what we have so far. Now the target can be extended to include B_1 or B_2 and more common flow regions can be obtained. Let the new target be given by $T' = T \cup B_1$. A common flow region entering T' under the second control policy is bounded by choosing

$$I_{B_3} = \{7\} \tag{64}$$

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

$$I_{B_5} = \{6\} \tag{65}$$

and $e = 7$. Fig. 16(b) gives a final picture of the hypersurfaces and regions involved in this example.

Given the hypersurfaces and the regions of the partition, a controller can be easily designed. Note that our synthesis methodology is described in Section V. Here, we present an intuitive way for the design of the controller. There is only one control goal for this example, and therefore, the entire controller will consist of a single subautomaton. Start by creating a controller state $\tilde{s}_T$, which is associated with the target region. Two common flow regions, labeled B_1 and B_2, were identified that lead to the target region. Then, create two more controller states, $\tilde{s}_1$ and $\tilde{s}_2$. B_1 consists of the trajectories that reach the target region under control policy $\tilde{r}_1$ and 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 transition labeled $\tilde{x}_1$, which is generated when the plant state crosses h_1 to enter the target region. Do the same for $\tilde{s}_2$. Next, create $\tilde{s}_3$ to go with B_3, and add a transition to $\tilde{s}_1$ labeled $\tilde{x}_2$. When all the common flow regions have their associated states and transitions, the controller is shown in Fig. 16(c).

B. Transition Stability

In this section, logical invariance of the DES plant transitions to variations in the initial continuous state is defined and conditions for inferring "stable" DES plant transitions are presented. For more details, the reader is referred to [36]. Ideally, the transitions in the DES plant should be unchanged by small perturbations in the state of the continuous plant. The important property is referred to here as transition stability, and it is related to the notion of "structural stability." This section presents a set of sufficient conditions ensuring the stability of transitions in the DES plant with respect to variations in the initial state of the continuous plant. These conditions are based on the Lyapunov stability theory, and hence, the notion of logical stable transitions is clearly related to conventional stability of continuous systems.

Assume that the partition of the state space is given. Let $\mathcal{B}$ denote the finite collection of q disjoint sets that partition the state space of the continuous plant, and denote the ith element as $B_i, i = 1, \cdots, q$. The desired behavior of the continuous plant can be described as a desired language for the DES plant model as described in Section V. This is a formal specification on how the plant should transition between elements of $\mathcal{B}$. The problem considered here is concerned with conditions that ensure that such transitions occur in a stable manner. DES plant validity can be viewed in terms of the invariance of plant and control event sequences to small perturbations in the state of the continuous plant. An arc of the DES plant represents a transition of the continuous plant's state between two subsets from $\mathcal{B}$. The labeling of that by a plant

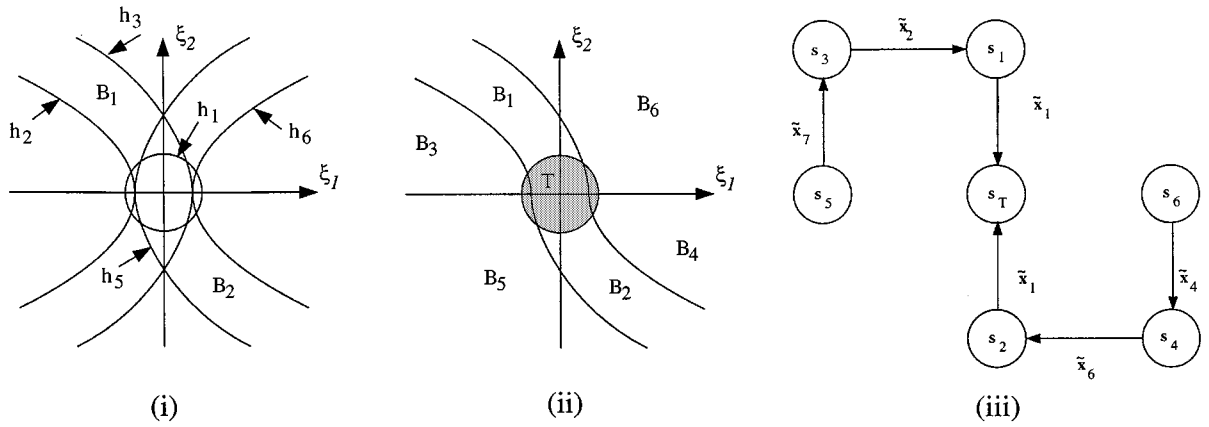

Fig. 16. (a) Target region and invariants. (b) Final regions for double integrator. (c) Controller.

symbol arc represents the symbolic behavior of that transition. A valid DES plant would preserve that labeling under small perturbations of the initial continuous state. This viewpoint is formalized in the following definition of T-*stability*.

Let $G = (\tilde{P}, \tilde{X}, \tilde{R}, \psi, \lambda)$ be a DES plant model for a hybrid dynamical system. Let $\tilde{p}_i$ and $\tilde{p}_j$ be two vertices in $\tilde{P}$ corresponding to the sets B_i and B_j, respectively. Consider the arc $(\tilde{p}_i, \tilde{p}_j)$ labeled with the control symbol $\tilde{r}$ and plant symbol $\tilde{x}$. Let $\mathbf{r} = \gamma(\tilde{r})$ be the control vector associated with control symbol $\tilde{r}$ through the interface actuator mapping γ, and denote by $\Phi_t^{\mathbf{r}}$: $\mathbf{X} \to \mathbf{X}$ the transition operator generated by the differential equation $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{r}(t))$.

The arc $(\tilde{p}_i, \tilde{p}_j)$ is *transition-stable* (or T-*stable*) iff for all $\mathbf{x}_0 \in B_i$, there exists an open neighborhood $N_\epsilon(\mathbf{x}_0)$ centered at $\mathbf{x}_0$ and a finite time $0 < T < \infty$ such that the set

$$N_T = \{\mathbf{x}_T \colon \mathbf{x}_T = \Phi_T^{\mathbf{r}}(\mathbf{x}),\ \mathbf{x} \in N_\epsilon(\mathbf{x}_0)\} \tag{66}$$

is an open subset of B_j and the plant symbol $\tilde{x}$ issued during the transition is identical for all transitions starting in $N_\epsilon(\mathbf{x}_0)$ and ending in N_T.

We consider a special case of nonlinear continuous plant, which is described by the following set of p differential equations:

$$\dot{x}^{(i)} = f_0\left(x^{(i)}\right) + \sum_{j=1}^{m} r_{ji} f_j\left(x^{(i)}\right) \tag{67}$$

where $x^{(i)}, i = 1, \cdots, p$ is the state vector for the ith differential equation, and $r_{ji}, j = 1, \cdots, m$ are the components of the control vector r_i for the ith equation. In this case, the collection of mappings f_j: $\mathbb{R}^n \to \mathbb{R}^n$ for $j = 0, \cdots, m$ represents a set of $m + 1$ *control policies*. The control policies of the ith differential equation are linearly mixed by the components of the control vector r_i. Such a nonlinear system is often referred to as being "affine" in its control vectors. Under suitable assumptions, it includes the class of nonlinear systems that can be linearized through appropriate feedback.

Consider a DES plant model of a hybrid system whose continuous plant is affine in its control policies. Consider an arc $(\tilde{p}_j, \tilde{p}_k)$ of the DES plant with control symbol label $\tilde{r}$, and let $\mathbf{r} = [r_1, \cdots, r_m]^T = \gamma(\tilde{r})$ be its corresponding control vector. Note that a sufficient condition for the transition to be T-stable is that all trajectories starting in B_j are attracted to B_k and are repelled by any other elements of $\mathcal{B}$. This condition is satisfied provided B_k contains a global attractor for the controlled system and all other $B_i, i \neq k$ are repellors. These conditions can easily be established by constructing a Lyapunov functional V_k over the state space such that the system is globally stable to B_k. To ensure that all other sets are repelling, it is sufficient to guarantee that there exists Lyapunov functional $V_i, i \neq k$ for each of these generator sets, which always forces the state trajectory out of the set.

In order to formulate these sufficient conditions, we use the Lie directional derivative of a functional. Let V: $\mathbb{R}^n \to \mathbb{R}$ be a continuous differentiable functional and let f: $\mathbb{R}^n \to \mathbb{R}^n$ be a smooth vector field. Let $\nabla_x V$ denote the gradient vector of V. The Lie (directional) derivative of V, $L_f V$: $\mathbb{R}^n \to \mathbb{R}$ is defined as the inner product of the gradient of V with the vector field f, $L_f V$: $(\nabla_x V)^T f$. The LaSalle invariance principle [35] can be used to establish the following sufficient conditions.

This arc is T-stable if there exists a set of continuously differentiable positive definite functionals V_i: $\mathbb{R}^n \to \mathbb{R}, i = 1, \cdots, q$, which are zero on a closed proper subset of $B_i \in \mathcal{B}, i = 1, \cdots, q$, such that for all $\mathbf{x} \in \mathbf{X}$ and $\mathbf{x} \notin B_k$

$$[L_{f_0} V_k, L_{f_1} V_k, \cdots, L_{f_m} V_k] \begin{bmatrix} 1 \\ r_1 \\ \vdots \\ r_m \end{bmatrix} < 0 \tag{68}$$

and for all $\mathbf{x} \in B_i$, for $i = 1, \cdots, q$ and $i \neq k$

$$[L_{f_0} V_i, L_{f_1} V_i, \cdots, L_{f_m} V_i] \begin{bmatrix} 1 \\ r_1 \\ \vdots \\ r_m \end{bmatrix} > 0. \tag{69}$$

The above conditions are clearly not necessary for T-stability. For many situations, they may only have to hold in a local sense. Nevertheless, these conditions are very valuable. Dynamical systems are always influenced by unpredicted external disturbances, which may force the plant state off the controlled trajectory. When such disturbances occur, it is desirable that the transition remain "stable." One way to ensure

that is to require that the sets of the partition are global attractors and repellors. Therefore, while the above conditions are restrictive, they provide a test that is useful in the face of unmodeled disturbances.

The sufficient conditions pertain to a single transition arc of the DES plant for a given control symbol. These conditions form a system of linear inequality constraints in the control space of the continuous plant. Feasible points satisfying the inequality system are therefore constant control vectors $\mathbf{r} \in \mathbb{R}^m$, which guarantee that the single arc is T-stable. By finding the feasible points for each arc in the DES plant, a set of control vectors $r_i, i = 1, \cdots, p$ associated with the control symbols $\tilde{r}_i$ is obtained. The systematic application of this approach to every arc in a given DES plant can then be used to determine an actuator mapping γ, which T-stabilizes the entire DES plant.

Deciding the T-stability of the entire DES plant can only be done if there exists a numerically efficient method for finding feasible points. One class of algorithms for doing this is the *method of centers* algorithms [46]. Method of center algorithms compute a sequence of convex bodies and their centers in such a way that the computed centers converge to a feasible point. Depending on the analytic form of the convex bodies and the centers, different types of algorithms are obtained. A particularly well-known example is the ellipsoid method [46]. In this algorithm, the convex bodies are ellipsoidal sets containing the set of feasible vectors, and the centers are the geometric centers of these ellipsoids. In [36], the ellipsoid method is used as an inductive learning algorithm for inferring T-stable interfaces.

C. Digital Control in a Hybrid Framework

In a digital control system, a continuous-time plant is controlled by a digital computer. The plant output is sampled and quantized to provide the input to the controller, and the control signal is passed through a zero-order hold to provide the plant input. Here, we show that such a digital control system is a special case of a hybrid control system. The more general framework of hybrid control systems is used to discuss problems encountered in digital control such as chattering and limit cycles and to illustrate the connection between hybrid and digital systems. More details can be found in [61]. Section IV-A presented a method for designing the interface of a hybrid control system using the natural invariants of the system. Here, this method is used as an alternative to the usual quantization technique of digital control. An example illustrates the application of this method to a digital control problem; it also shows the application of a DES controller design approach to digital control.

A digital control system can be viewed as a special case of a hybrid control system in which the plant, interface, and controller obey certain constraints. In a digital control system, the events are triggered at regular (or, if desired, irregular) intervals of time. This gives rise to sampling in time and requires that there must be a clock present in the system. In digital control, this clock is normally not modeled explicitly, but in our hybrid control framework it will appear as part of the plant. A convenient way to do this is to use a two-state oscillator, such as

$$\dot{\mathbf{x}}_c = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix} \mathbf{x}_c \tag{70}$$

where ω is the clock frequency. If required, ω can be made a function of the plant input $\mathbf{r}(t)$ and then the clock frequency can be changed by the controller. To implement the sampling, hypersurfaces of the type

$$h_i(\mathbf{x}_c) = k_1 x_{c1} + k_2 x_{c2} \tag{71}$$

are used, where k_1 and k_2 are real constants. A single hypersurface with $k_1 = 1$ and $k_2 = 0$ would model the typical case where the sampling rate is given by $\omega/2\pi$. In this case, the partition of the state space looks like a grid. By adding more hypersurfaces, each with its own measurement function α_i, multirate sampling can be modeled.

When modeling digital control, the measurement function(s) α_i is a quantizer. The value of each state at the sampling instant is truncated or rounded and restricted to the range of values acceptable to the digital controller. For example, a controller implemented on an 8-bit computer might only have the capability to measure a state to 256 possible values. In that case, we might have a set of plant symbols given by

$$\tilde{X} = \{0, 1, \cdots, 255\} \tag{72}$$

and a measurement function given by

$$\alpha_1(\mathbf{x}(t)) = \text{trunc}(\mathbf{x}(t) + 128) \text{ modulo } 256. \tag{73}$$

Quantization of this type will form a grid-like partition of the state space into regions, each associated with the same plant symbol.

On the other side of the interface, the actuator models a zero-order hold. Controller symbols, representing quantized plant input values, are converted to piecewise constant plant inputs. In an example akin to the one given above, the set of controller symbols is given by

$$\tilde{R} = \{0, 1, \cdots, 255\} \tag{74}$$

and γ is given by

$$\mathbf{r}(\tau_c[n]) = \tilde{r}[n] - 128. \tag{75}$$

Finally, the controller is an automaton that implements the desired control strategy for the digital controller.

The designer is free to choose the sampling rate(s) and the quantization level(s) for each state. These choices determine the dimensions of the n-dimensional boxes that fill up the state space. What the designer cannot do is change the basic shape of the boxes that form the partition; in digital control with sampling in time they will always be rectangular. On the other hand, most hybrid control design possibilities cannot be realized in digital control because of the need for a grid-like partition.

In digital control, the problem of choosing the appropriate control policy is many times handled by approximating an existing continuous control law. Since state-space trajectories are not generally straight lines, they will not flow "neatly" through the grid-like partition of the digital control

system. Trajectories in a given region of the partition will inevitably leave that region through more than one of the region's boundaries. This gives rise to nondeterministic behavior in the DES plant. The problem cannot be solved by changing the size of the region because it will still have the same grid-like shape. In hybrid control, the strategy is to shape the regions according to the system trajectories so as to control which boundaries the trajectories can pass through. This is not possible with a digital control system. Fig. 17 shows an example of the added flexibility afforded by hybrid control over standard digital control.

Since the problems encountered in digital control are largely a result of the quantization, it is reasonable to try to solve them by changing the way the quantization is done. The usual quantization forms a grid-like partition of the state space, forming an array of n-dimensional boxes—each with its own symbol. The problem is that these quantization levels may have no relationship to the state trajectories that flow through them. To reduce this problem, the grid-like partition can be replaced with a partition based on the natural invariants of the system. The following illustrating example explores the use of invariants for quantization.

1) Example—Double Integrator: In this example, we have a double integrator that we wish to stabilize. The interface is designed to quantize based on system invariants and the controller is again designed with methods from hybrid control. We start with a double integrator plant

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{r}. \tag{76}$$

The natural invariants of the system are to be used in the quantization. Three possible inputs are used to control the double integrator $\{-2, 0, 2\}$, and they are used to compute three invariant functions for the system. The invariant function associated with the input -2 is computed as follows. The control policy is

$$f_1(\mathbf{x}) = x_2 \tag{77}$$

$$f_2(\mathbf{x}) = -2. \tag{78}$$

This gives the characteristic equation

$$\frac{dx_1}{x_2} = \frac{dx_2}{-2} \tag{79}$$

with a solution

$$g_1(\mathbf{x}) = x_1 + 0.25x_2^2. \tag{80}$$

The remaining two invariant functions can be computed similarly:

$$g_2(\mathbf{x}) = x_2 \tag{81}$$

$$g_3(\mathbf{x}) = x_1 - 0.25{x_2}^2. \tag{82}$$

The values of these invariant functions are quantized to form the plant symbols

$$\alpha(\mathbf{x}) = 0.25\text{round}(4g(\mathbf{x})). \tag{83}$$

Now, the quantization yields the partition shown in Fig. 18(a).

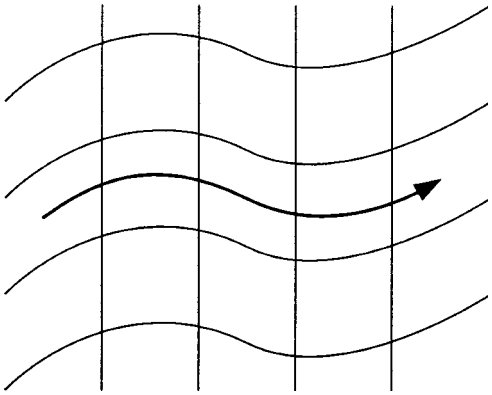

Fig. 17. Nongrid-like partition.

The controller for this case is

$$\tilde{r}[k] = \begin{cases} -2.0, & \text{if } g_3(\mathbf{x}) < 0,\ g_2(\mathbf{x}) < 0 \\ 2.0, & \text{if } g_2(\mathbf{x}) > 0,\ g_3(\mathbf{x}) > 0 \\ 0.0, & \text{otherwise.} \end{cases} \tag{84}$$

Fig. 18(b) shows a state trajectory for this system. Note that chattering has been avoided and the controller easily follows from the interface design by identifying the hypersurfaces that will drive the state to the origin.

V. Supervisory Control Design

In conventional control, theoretic measures of system performance are frequently taken to be norms ("size") of some important signals within the control system. Unfortunately, norm-based performance measures may be inappropriate for supervised systems, since many times the space of interest is not metric and such measures do not exist. A different way to express performance of a system is needed that, for example, may be used to supervise the start-up procedure of a process plant. In the following, we use a language theoretic framework to describe performance specifications for hybrid systems, and we formulate the supervisory control problems for hybrid systems. We also use the language generated by the DES plant to examine the controllability of the hybrid control system, we present a methodology for controller design, and we illustrate the framework using several examples.

Once the DES plant model of a hybrid system has been extracted, a supervisor can be designed using controller synthesis techniques based on discrete-event systems. Our work builds upon the framework of supervisory control theory initiated by Ramadge and Wonham [57], [58], [68]. Here, we adapt several of those results and apply them to the DES plant model obtained from a hybrid control system. The main differences between the Ramadge–Wonham framework and the DES plant models of the hybrid control framework are the nondeterminism of the plant and the inability to disable plant events individually. These differences and an extension of the supervisor synthesis algorithm in [57] to the design of hybrid control systems are discussed in Section V-B. The logical DES approach to the design of hybrid control systems has been described in [65]. Note that more details for supervisory control of nondeterministic discrete-event systems can be found, e.g., in [25], [32], and [50].

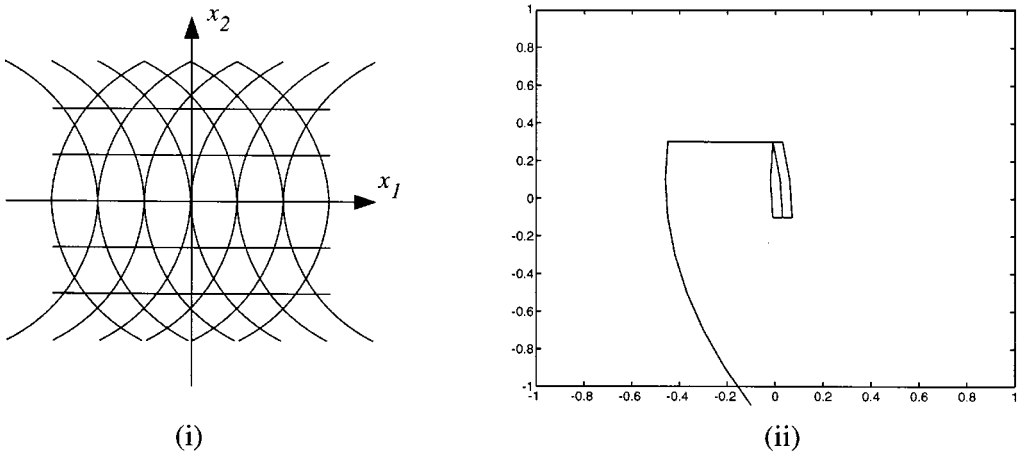

Fig. 18. (a) Quantization. (b) State trajectory.

A. The Supervisory Control Problem for Hybrid Systems

The objective is to develop methodologies that, given the system description and performance specifications, extract discrete-event controllers that supervise the plant to guarantee that these specifications are satisfied. The symbolic behavior of the plant is explicitly seen from the DES controller's perspective. From the controller's perspective, it is the DES plant model that is controlled. The state symbol sequence $\tilde{p}[n]$ represents the symbolic evolution of the continuous plant over the event partition. In order to represent the evolution of the DES plant model, we introduce some notions from the theory of discrete event systems (see, e.g., [58] and [19]) that are necessary for the presentation of the design approach.

Consider the set of all plant symbols $\tilde{X}$, which is also called *the alphabet*. This set consists of all the plant symbols that can possibly be generated by the DES plant model. Denote by $\tilde{X}^*$ the set of all finite strings formed by concatenation of symbols from the alphabet, including the empty string ϵ; the $*$ operation is called the Kleene closure. A *language* is formally defined as a subset of $\tilde{X}^*$. The usual set operations, such as union, intersection, difference, and complement (with respect to $\tilde{X}^*$), are applicable to languages. In addition, the *prefix-closure* of L, denoted by $\overline{L}$, is defined as the set of all prefixes of strings in L. The language L is said to be *prefix-closed* if all the prefixes of the language are also in L, or equivalently, if $L = \overline{L}$.

In our approach, the DES plant is represented as a nondeterministic finite automaton. The behavior of the DES plant G is represented by the language $L \subseteq \tilde{X}^*$. This is the set of all finite sequences of symbols the DES plant can generate. Since the language L is generated by a finite automaton, it is known to be prefix-closed and *regular* [26].

The plant symbols $\tilde{x}$, which are in the language of a given DES plant model, are defined as follows. Given a finite sequence of plant symbols $\tilde{x}$: $\mathbf{N} \rightarrow \tilde{X}$ defined over the set $\mathbf{N} = \{1, \cdots, N\}$, then $\tilde{x} \in L$ if there exists $\tilde{p} \in \tilde{P}^*$ and $\tilde{r} \in \tilde{R}^*$ such that the following conditions hold:

$$\tilde{p}[n+1] \in \psi(\tilde{p}[n], \tilde{r}[n]), \qquad \forall n \in \mathbf{N} \tag{85}$$

$$\tilde{x}[n] \in \lambda(\tilde{p}[n-1], \tilde{p}[n]), \qquad \forall n \in \mathbf{N}. \tag{86}$$

Since each specification can be described by a language consisting of symbols generated by the DES plant model, the formal description of the performance specifications depends directly on the selected extraction of the DES plant.

In the supervisory control paradigm, the objective of the controller is to restrict the behavior of a given uncontrolled DES in order to satisfy prescribed specifications on the languages generated by the system. Performance specifications can be viewed as requiring that certain undesirable sequences of events are not permitted to occur, while at the same time, certain other desirable sequences are permitted. The uncontrolled DES plant model is assumed to generate "illegal behavior" that should be avoided by appropriate control action. Each specification can be described by a language consisting of symbols generated by the DES plant model. The "legal behavior" is characterized as a subset of the DES language after accounting for all the performance specifications that are imposed on the system.

The plant symbols in the DES plant are divided into two sets, those that are *controllable* and those that are *uncontrollable*: $\tilde{X} = \tilde{X}_c \cup \tilde{X}_u$. A plant symbol's being controllable means that the supervisor can prevent it from being issued by the DES plant model. When the supervisor prevents a controllable plant symbol from being issued, the plant symbol is said to be *disabled*. We assume that the state transition function of the DES plant is controlled by an external agent in the sense that the controllable events can be disabled by a supervisor.

Before we present the extension of the logical DES framework [58] to hybrid control systems, the important differences must be discussed. The Ramadge–Wonham (RW) model consists of two interacting DESs, the *generator* and the *supervisor*. The RW generator is analogous to our DES plant, and the RW supervisor corresponds to our DES controller. In the RW framework, the plant symbols can be individually disabled, at any time and in any combination, by a command from the 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. The particular subset of $\tilde{X}$ that is disabled by a given controller symbol depends on the state transition function ψ and output function λ of the DES plant model. In addition, there is no guarantee that any arbitrary subset of $\tilde{X}$ can be disabled while the other plant symbols remain enabled. The inability to disable plant symbols individually is what differentiates the DES plant model from the automata of earlier frameworks.

The DES plant model is connected in the feedback loop to a supervisor S. For each possible string of plant symbols generated by the DES plant model, the supervisor specifies the control symbol $\tilde{r}$ to be applied. Each control symbol, in turn, disables a particular set of plant symbols in order to prevent undesirable sequences of events. Therefore, the supervisor (or controller) S can be described as a function

$$S\colon L \to \tilde{X} := \left\{\Gamma \in 2^{\tilde{X}}\colon \tilde{X}_u \subseteq \Gamma\right\} \tag{87}$$

specifying the control action to be taken for each possible string. Note that the control always contains the set $\tilde{X}_u$, since the supervisor never disables an uncontrollable event. The language generated by the closed-loop system (S, G) is defined recursively as follows:

1) $\epsilon \in L(S, G)$;
2) $s\sigma \in L(S, G)$ iff $(s \in L(S, G))$ and $(s\sigma \in L)$ and $(\sigma \in S(s))$.

Given the DES plant model $G = (\tilde{P}, \tilde{X}, \tilde{R}, \psi, \lambda)$ and desired language K, the objective in the supervisory control problem for hybrid systems is to build a supervisor S such that $L(S, G) \subseteq K$. In addition, it is required that the supervisor is maximally permissive, meaning that the language $L(S, G)$ is as large as possible.

In this paper, we are interested in the case when the languages $L(S, G)$ and K are regular. In this case, the supervisor can be realized as a deterministic finite automaton $S = (\tilde{S}, \tilde{X}, \tilde{R}, \delta, \phi)$ in a straightforward manner, as shown in the examples of Section V-B. Note that this case is of special interest, since the supervisor has an implementable finite representation. For more details on the realization of supervisors, see, e.g., [19].

B. Controllability and Supervisor Design

The basic problem in supervisory control design of hybrid systems is to modify the behavior of the DES plant model so that it satisfies the specifications described by a desired language K. Therefore, the main question is if the language K can be achieved by supervision of the DES plant model. This question is directly related to the *controllability* of the language K. More generally, the concept of controllability of a language is used to identify all the possible closed-loop behaviors $K \subset L$ that can be achieved by control given a DES with language L. Next, we discuss controllability for DES systems [58], and then we present a definition of controllable languages for DES plant models extracted from hybrid dynamical systems.

Given a discrete-event system described by a finite automaton, it is possible to determine whether a desired language can be achieved by supervision; i.e., whether it is possible to design a controller such that the system will be restricted to some target language K. Such a controller can be designed if K is prefix closed and satisfies the following condition:

$$\overline{K}\tilde{X}_u \cap L \subset \overline{K}. \tag{88}$$

When (88) is true, the desired language K is said to be *controllable*, and provided K is prefix-closed, a controller can be designed that will restrict the system to the language K. This condition is very intuitive. It 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. It is clear that if an uncontrollable event σ occurs along a string s in K, then the extended string $s\sigma$ must remain in K.

If the desired language K is not attainable for a given DES, it may be possible to find a more restricted language that is. Since we want the least restricted behavior, it is desirable to find the supremal element of the family of controllable sublanguages of K under the partial order of set inclusion. An algorithm for finding this behavior, which is referred to as the *supremal controllable sublanguage* $K^{\uparrow}$ of the desired language, is described in [68]. The supremal controllable sublanguage is the largest subset of K that can be attained by a controller and can be found via the following iterative procedure:

$$K_0 = K \tag{89}$$

$$K_{i+1} = \left\{w \in \overline{K}_i,\, w\tilde{X}_u \cap L \subset \overline{K}_i\right\} \tag{90}$$

$$K^{\uparrow} = \lim_{i\to\infty} K_i. \tag{91}$$

The basis of the algorithm is a fixed point iteration of a certain operator on languages. The largest fixed point of the iteration is computed by iterative applications of the operator. Each iteration of (91) corresponds to the application of the operator. For finite automata and regular languages, this fixed-point iteration converges in finite steps [58]. It can be shown that $K^{\uparrow}$ is also a regular language, and therefore, it can be realized by a supervisor described by a finite automaton.

Since the DES plant model belongs to a slightly different class of automata than the Ramadge–Wonham framework, we present another definition for controllable language that applies to the DES plant. We assume in this section that all languages are prefix-closed, q_0 is the initial state, and $\tilde{x}$ is a finite sequence of plant symbols, $\tilde{x}\colon \mathbf{N} \to \tilde{X}$, defined over the set $\mathbf{N} = \{1, \cdots, N\}$.

In a hybrid control system, the controller must provide a controller symbol, following the generation of each plant symbol. Furthermore, the effect of the controller symbol on the behavior of the system is revealed, at least partially, by the current state of the DES plant. For this reason, it is desirable to determine the state of the DES plant model from a finite sequence of plant (output) symbols. In order to satisfy this objective, we assume that the current state can be determined uniquely from the previous state and plant symbol. This assumption should not be confused with the nondeterminism of the DES plant model. Note that the DES plant is said to be deterministic if for a given state and control (input)

event, there is only one possible subsequent state. As we have shown, the DES plant model is described, in general, by a nondeterministic finite automaton. Here, our assumption is made with respect to the plant (output) symbols and is similar to the concept of observability [51].

This is a realistic assumption for practical applications of hybrid systems. The plant symbols represent the measurements from the continuous plant. Each plant symbol corresponds to a hypersurface and to a direction of crossing that hypersurface, and it is issued when the continuous state crosses this hypersurface. If the current state is known and a plant symbol is detected, then we can determine the successor state uniquely. Note that this assumption is not inconsistent with nondeterminism in the DES plant model, since in a nondeterministic DES plant model, the successor state cannot be determined uniquely by the current state and the control symbol applied by the controller.

Since the current state can be determined uniquely from the previous state and plant symbol, 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 $\tilde{x}$. This assumption implies the existence of a mapping, obs: $\tilde{P} \times \tilde{X}^* \rightarrow \tilde{P}^*$, which takes an initial state together with a sequence of plant symbols and maps them to the corresponding sequence of states. The nth state in the sequence $\tilde{p}[n]$ can also be written as $obs(q_0, \tilde{x})[n]$, where $q_0 \in \tilde{P}$ was the initial state. The mapping obs is needed for the following definition for controllable languages, which applies to the DES plant.

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

$$\tilde{x}\lambda(q, \psi(q, \rho)) \subset K \tag{92}$$

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 that will cause the string to remain in K. This definition implies the next technical result shown in [65].

Proposition 2: If the language K is controllable, then a controller can be designed that will restrict the given DES plant to the language K.

Since the concept of controllability for the language generated by the DES plant model can be seen as an extension of the Ramadge–Wonham framework to the hybrid system case, the conditions in (92) reduce to those of (88) under appropriate restrictions. These restrictions basically are that the plant symbols fall into a controllable/uncontrollable dichotomy and a control policy exists to disable any combination of controllable plant symbols.

For hybrid control systems, the supremal controllable sublanguage of the DES plant can be found by a similar iterative scheme:

$$K_0 = K \tag{93}$$

$$K_{i+1} = \Big\{ w \in K : \forall \tilde{x} \in \overline{w} \, \exists \rho \in \tilde{R} \text{ such that } \tilde{x}\lambda(q, \psi(q, \rho)) \subset K_i \Big\} \tag{94}$$

$$K^{\uparrow} = \lim_{i \rightarrow \infty} K_i. \tag{95}$$

For regular languages, it can be shown that the above iteration also converges in finite steps and that $K^{\uparrow}$ is regular. From (94), it follows that for any $\tilde{x} \in K^{\uparrow}$, there exists a control symbol $\rho \in \tilde{R}$ such that $\tilde{x}\lambda(q, \psi(q, \rho)) \subset K^{\uparrow}$; therefore, the language $K^{\uparrow}$ is controllable. This result yields the following proposition.

Proposition 3: For a DES plant and language K, $K^{\uparrow}$ is controllable and contains all controllable sublanguages of K.

The supremal controllable sublanguage is regular and can be realized with a supervisor described by a finite automaton as illustrated by the following examples. Related work on the supremal controllable sublanguage in the discrete-event model of nondeterministic hybrid control systems can be found in [69].

1) Example—Double Integrator: The system consists of a double integrator plant, which is controlled by a discrete event system. Consider the double integrator example with the DES plant shown in Fig. 10. 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 $K = \overline{(\tilde{x}_2\tilde{x}_1)^*}$. In this example, it can be shown that the language K is controllable because it satisfies (92). This can also be seen by observing Fig. 10. If the current state is either $\tilde{p}_5$ or $\tilde{p}_{10}$, then the system can evolve in a clockwise direction. If the current state is $\tilde{p}_9$, then the plant symbol $\tilde{x}_2$ can be disabled by selecting the control symbol $\tilde{r}_2$. Similarly, for $\tilde{p}_6$, $\tilde{x}_2$ can be disabled by selecting $\tilde{r}_1$. Therefore, according to Proposition 2, a controller can be designed to achieve the stated control goal. The controller for this example is shown in Fig. 19, and its output function ϕ is as follows:

$$\phi(\tilde{s}_1) = \tilde{r}_1 \quad \phi(\tilde{s}_2) = \tilde{r}_1 \tag{96}$$

$$\phi(\tilde{s}_3) = \tilde{r}_2 \quad \phi(\tilde{s}_4) = \tilde{r}_2. \tag{97}$$

2) Example—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 Fig. 20.

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

$$L_m = (\tilde{x}_2(\tilde{x}_1 + \tilde{x}_4(\tilde{x}_5\tilde{x}_4)^*\tilde{x}_1 + \tilde{x}_3(\tilde{x}_6\tilde{x}_3)^* \cdot (\tilde{x}_1 + \tilde{x}_6\tilde{x}_5\tilde{x}_4(\tilde{x}_5\tilde{x}_4)^*\tilde{x}_1)))^*. \tag{98}$$

A problem that appears very often in hybrid system is to supervise the system so that it will not enter an unsafe region. Suppose we want to control the DES so that it never enters state $\tilde{p}_4$. We simply remove the transitions to $\tilde{p}_4$ and then compute the resulting language. This desired language is therefore

$$K = \overline{(\tilde{x}_2(\tilde{x}_1 + \tilde{x}_4\tilde{x}_1 + \tilde{x}_3(\tilde{x}_6\tilde{x}_3)^*\tilde{x}_1))^*}. \tag{99}$$

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 $\rho \in \tilde{R}$ that will prevent the DES plant from deviating from K by generating $\tilde{x}_5$ and entering state $\tilde{p}_4$. Since

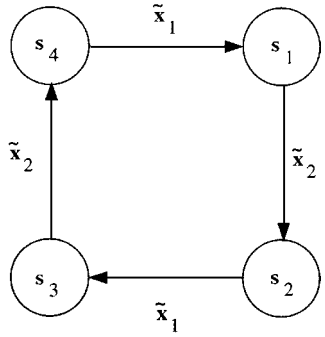

Fig. 19. Controller for the double integrator.

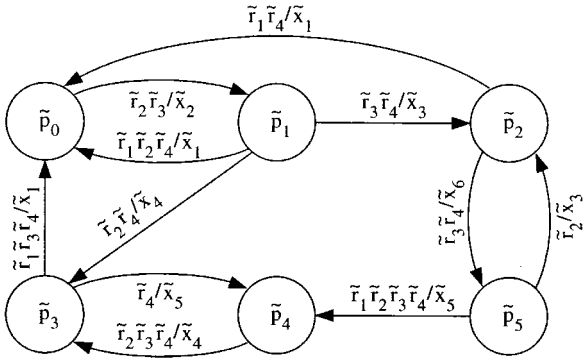

Fig. 20. DES plant model.

K is not controllable, we find the supremal controllable sublanguage of K as defined in (95). The supremal controllable sublanguage is

$$K^{\uparrow} = K_1 = \overline{(\tilde{x}_2(\tilde{x}_1 + \tilde{x}_4\tilde{x}_1 + \tilde{x}_3\tilde{x}_1))^*}. \tag{100}$$

Obtaining a DES controller once the supremal controllable sublanguage has been found is straightforward. The controller is a DES whose language is given by $K^{\uparrow}$. Since the language $K^{\uparrow}$ is regular, the supervisor is implemented by a finite automaton that generates the language $K^{\uparrow}$. Details regarding the equivalence between finite automata and regular languages can be found in [26]. The output of the controller in each state $\phi(\tilde{s})$ is the controller symbol, which enables only transitions that are found in the controller. The existence of such a controller symbol is guaranteed by the fact that $K^{\uparrow}$ is controllable. For this example, the controller is shown in Fig. 21 and its output function ϕ is as follows:

$$\phi(\tilde{s}_1) = \tilde{r}_2 \quad \phi(\tilde{s}_2) = \tilde{r}_4 \tag{101}$$

$$\phi(\tilde{s}_3) = \tilde{r}_1 \quad \phi(\tilde{s}_4) = \tilde{r}_1. \tag{102}$$

3) 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 [44]. Here, a condensed description is given to show the source of the DES plant model and provide insight into the physical meaning of the states and events.

Fig. 22 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

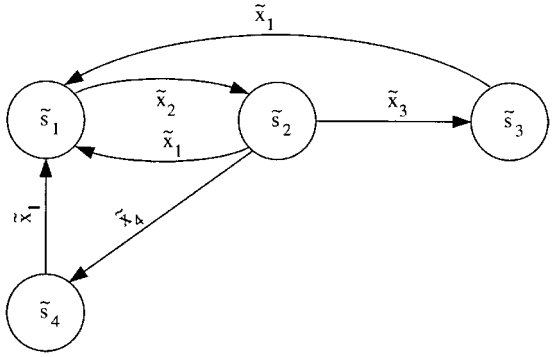

Fig. 21. DES controller.

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 $(x_B \to 0)$.

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:

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

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

$$F_L = 0.6F \quad F_V = F - F_L \quad x_F = (0.5F - F_V y_F)/F_L$$

and the outputs are

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

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:

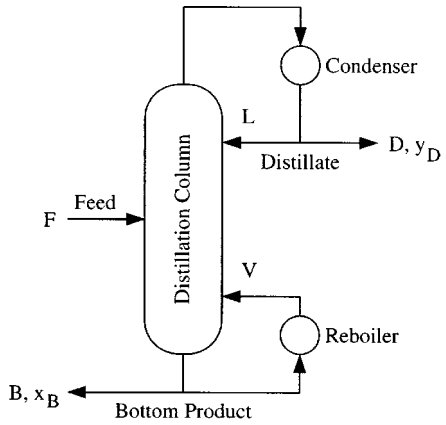

Fig. 22. Distillation column.

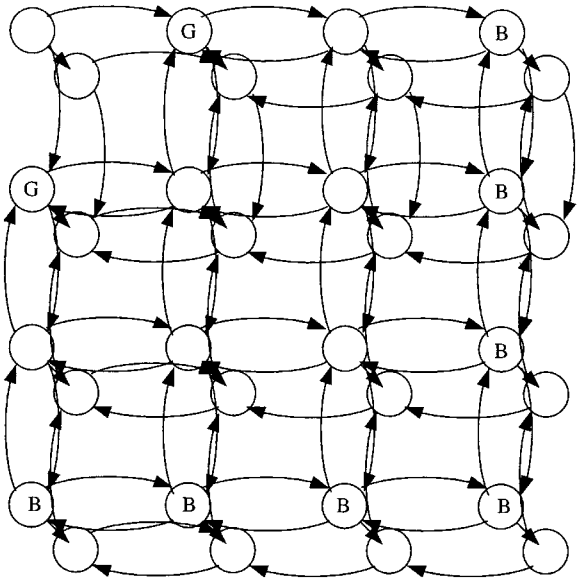

Fig. 23. DES plant for the distillation column.

$\tilde{x}_1$ $B + D$ falls below 2;
$\tilde{x}_2$ $B + D$ exceeds 2;
$\tilde{x}_3$ x_B falls below 0.13;
$\tilde{x}_4$ x_B exceeds 0.13;
$\tilde{x}_5$ x_B falls below 0.12;
$\tilde{x}_6$ x_B exceeds 0.12;
$\tilde{x}_7$ x_B falls below 0.08;
$\tilde{x}_8$ x_B exceeds 0.08;
$\tilde{x}_9$ y_D falls below 0.84;
$\tilde{x}_{10}$ y_D exceeds 0.84;
$\tilde{x}_{11}$ y_D falls below 0.85;
$\tilde{x}_{12}$ y_D exceeds 0.85;
$\tilde{x}_{13}$ y_D falls below 0.95;
$\tilde{x}_{14}$ y_D exceeds 0.95.

We would like to keep x_B below 0.13, y_D above 0.95, 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 twofold: 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. Fig. 23 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.

A controller was obtained by automating the procedure for finding the supremal controllable sublanguage. The controller is shown in Fig. 24. This controller drives the plant from the initial state to a loop containing the two good states. Note that in this figure, the states of the controller have been labeled with the controller symbol that is generated by that state.

4) Example—Robotic Manufacturing System: An example of a free floating robotic vehicle with two articulated arms is presented. The robotic arms shown in Fig. 25 are required to obtain components from a *parts bin* and move these components to *work areas* where assembly operations are to be performed. The tasks of fetching the workpiece, transporting it to the work area, and then returning to the parts bin to fetch another workpiece are performed repeatedly. The introduction of a shared resource generates a *mutual exclusion* constraint on the system. This example is particularly interesting because of the free-floating base, which makes the dynamics quite challenging. Similar problems arise in control and coordination of modern complex engineering applications such as autonomous vehicles and multibatch chemical processes. The robotic manufacturing example described here has been used in [37] to illustrate various concepts in hybrid system theory. A simplified version of the system without the free rotating table has been used in [29] and [28] to illustrate regulatory control of hybrid systems based on discrete abstractions.

The motions of the arms are described by the following ordinary differential equations:

$$\ddot{\theta}_1 = -\dot{\theta}_1 + k(\theta_1 + \theta_b - r_1) \tag{103}$$
$$\ddot{\theta}_2 = -\dot{\theta}_2 + k(\theta_2 + \theta_b - r_2) \tag{104}$$

where θ_1 and θ_2 are the angular positions of arm 1 and arm 2 with respect to the body axis of the robot. For this example, the control law is a proportional feedback law with gain k and with reference inputs r_1 and r_2. These reference inputs represent commands that direct the arm to move to the parts bin or work area. The movement of the arms will induce a body rotation so that the total angular momentum of the system is conserved. Let $\overline{\theta}_1 = \theta_1 + \theta_b$ and $\overline{\theta}_2 = \theta_2 + \theta_b$ denote the inertial angles of the robot arms 1 and 2, respectively. The body angle θ_b with respect to the inertial frame must satisfy

$$J_b\dot{\theta}_b + J_a\dot{\overline{\theta}}_1 + J_a\dot{\overline{\theta}}_2 = 0 \tag{105}$$

where J_b and J_a are the moments of inertia for the body and arms, respectively.

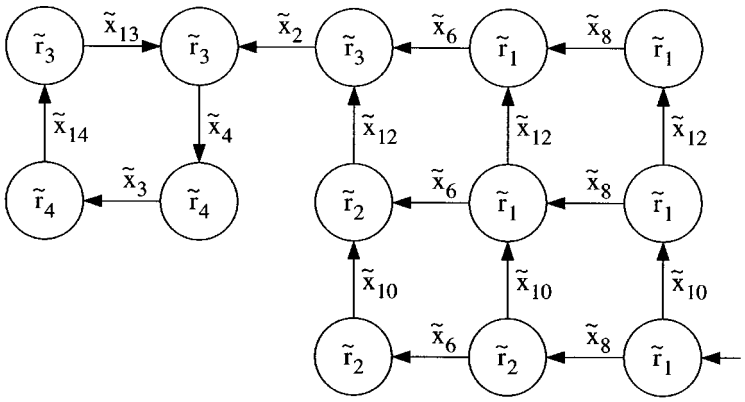

Fig. 24. Sample controller for distillation column.

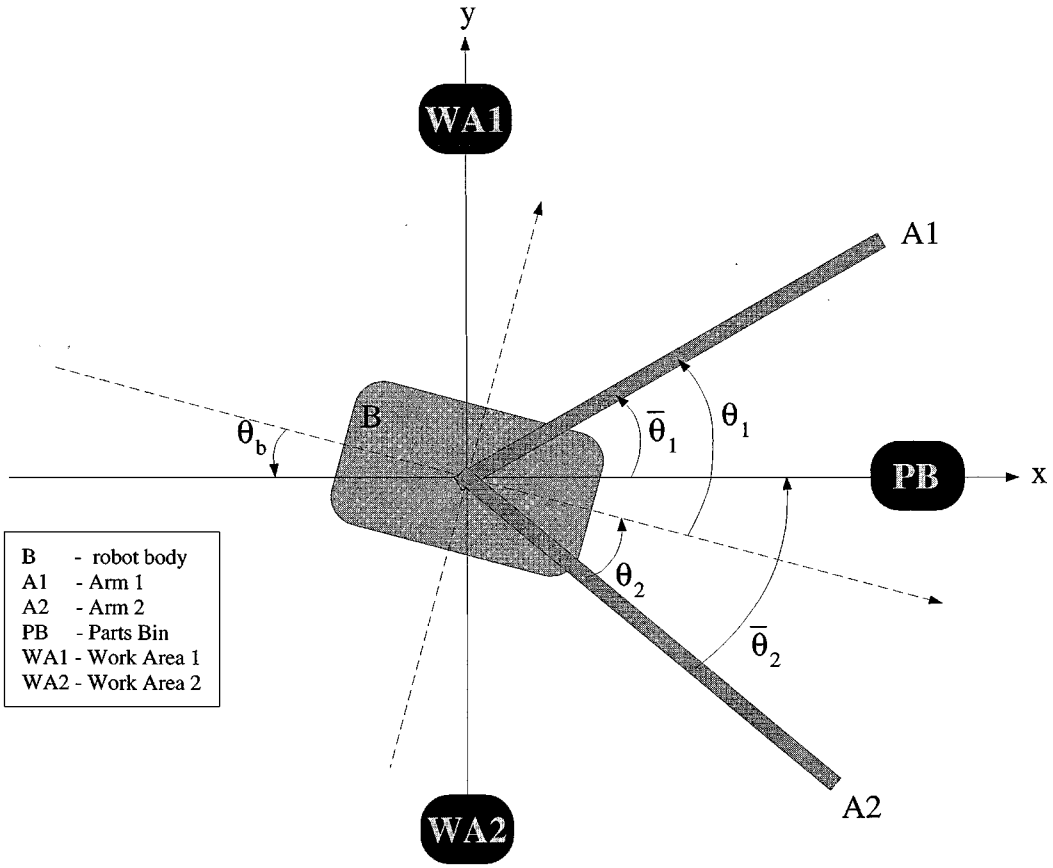

Fig. 25. Robotic manufacturing system on a free rotating platform.

The available control policies for the ith robotic arm are defined as follows:

$\tilde{r}_{i1}$ drive arm i to parts bin;
$\tilde{r}_{i2}$ drive arm i to work area;
$\tilde{r}_{i3}$ stop arm i.

Note that continuous controllers that guarantee that each command signal is executed in a suitable manner may be necessary. As discussed in Section II, it is assumed that these continuous controllers are included in the description of the plant. Next, plant symbols are defined based on events as follows:

$\tilde{x}_1$ Arm 1 approaches the parts bin;
$\tilde{x}_2$ Arm 1 enters the parts bin;
$\tilde{x}_3$ Arm 1 exits the parts bin;
$\tilde{x}_4$ Arm 1 leaves the parts bin;
$\tilde{x}_5$ Arm 2 approaches the parts bin;
$\tilde{x}_6$ Arm 2 enters the parts bin;
$\tilde{x}_7$ Arm 2 exits the parts bin;
$\tilde{x}_8$ Arm 2 leaves the parts bin.

The generator was designed to recognize eight different plant events. This leads to nine different regions in the state space, and therefore, the DES plant model has nine states as shown in Fig. 26.

We want to control the robotic manufacturing system so that it never enters the critical section. Therefore, the control requirement for the DES plant is that it never enters state $\tilde{p}_8$.

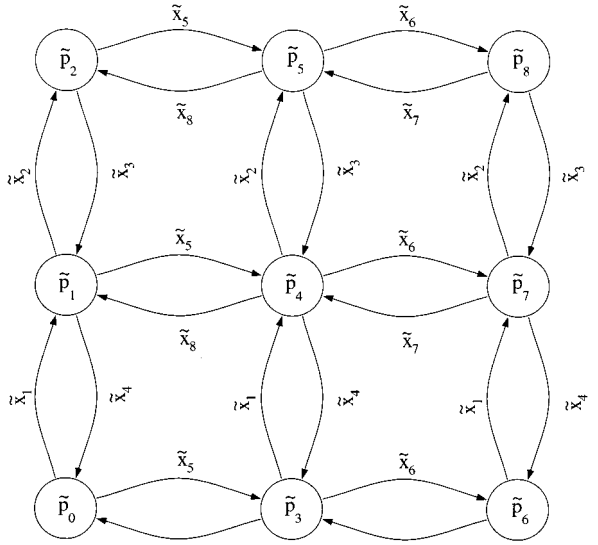

Fig. 26. DES plant model for the free-floating robotic system.

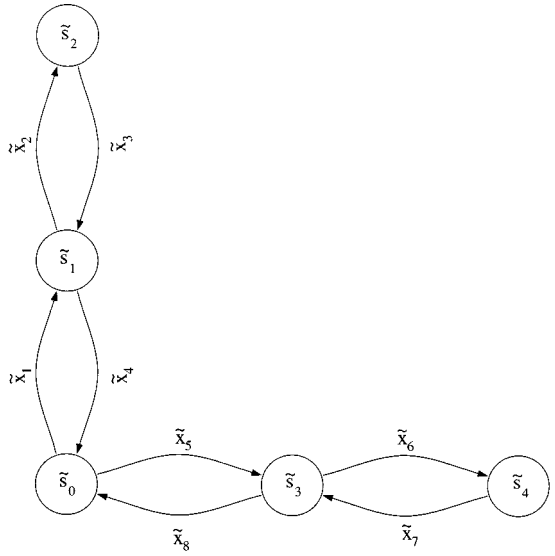

Fig. 27. DES controller for the free-floating robotic system.

The controller shown in Fig. 27 was obtained based on the supremal controllable sublanguage and does not allow the robotic arms to enter the critical section at the same time.

VI. Conclusions

In this paper, the supervisory control of hybrid systems has been introduced and discussed at length. Discrete abstractions that are represented by a DES plant model have been used to approximate the continuous plant. In general, the abstracting DES models are nondeterministic. Properties of the DES plant model to be a valid representation of the continuous plant have been presented. The emphasis has been put on the design of the interface between the continuous plant and the discrete event controller. A methodology to design the partition of the continuous state space based on the natural invariants of the plant has been briefly outlined. The robustness problem of the discrete transitions subject to small variations of the continuous system has also been addressed. Note that robustness to parameter variation is still an open issue in supervisory control of hybrid systems. An alternative methodology to the usual quantization technique of digital control based on the interface of hybrid control systems has been presented. The types of problems that have been addressed are those with control specifications that can be described by formal languages accepted by the DES plant model. The supervisory control problem for hybrid systems has been formulated, and algorithms for supervisory design based on the controllability of the specification language have been presented. Although the approach in this paper was based on a continuous-time model of the plant, similar results have been obtained using discrete-time systems [65], [59]. It should be noted that our coverage is primarily of a tutorial nature, and so many technical details have been just briefly outlined or simply omitted; the reader should consult the references for further details.

In this paper, we focused on the case when finite automata are used to describe both the plant and the controller. Hybrid control approaches based on Petri nets have been reported in the literature (see, e.g., the survey paper [8]). A similar approach to the one described in this paper using Petri nets, which may be computationally more efficient for large concurrent systems, has been reported in [23] and [31]. This approach addresses a particular class of supervisory control problems described by convex constraints on the marking of the Petri nets [43].

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P.-H. Ho, X. Nicollin, A. Oliveiro, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoret. Comput. Sci.*, vol. 138, pp. 3–34, 1995.

[2] R. Alur, T. Henzinger, and E. Sontag, Eds., *Hybrid Systems III, Verification and Control*. Berlin, Germany: Springer-Verlag, 1996, vol. 1066, Lecture Notes in Computer Science.

[3] P. Antsaklis, "Defining intelligent control," *IEEE Contr. Syst.*, pp. 4–5, 58–66, June 1994.

[4] ——, "Intelligent control," in *Encyclopedia of Electrical and Electronics Engineering*. New York: Wiley, 1997.

[5] P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, Eds., *Hybrid Systems V*. Berlin, Germany: Springer-Verlag, 1999, vol. 1567, Lecture Notes in Computer Science.

[6] P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds., *Hybrid Systems II*. Berlin, Germany: Springer-Verlag, 1995, vol. 999, Lecture Notes in Computer Science.

[7] ——, *Hybrid Systems IV*. Berlin, Germany: Springer-Verlag, 1997, vol. 1273, Lecture Notes in Computer Science.

[8] P. Antsaklis and X. Koutsoukos, "On hybrid control of complex systems: A survey," in *3rd Int. Conf. ADMP'98, Automation of Mixed Processes: Dynamic Hybrid Systems*, Reims, France, Mar. 1998, pp. 1–8.

[9] P. Antsaklis, X. Koutsoukos, and J. Zaytoon, "On hybrid control of complex systems: A survey," *Eur. J. Automat.*, vol. 32, no. 9–10, pp. 1023–1045, 1998.

[10] P. Antsaklis and M. Lemmon, "Introduction to the special issue on hybrid systems," *J. Discrete Event Dyn. Syst.*, vol. 8, p. 10, June 1998. (Special Issue on Hybrid Control Systems).

[11] P. Antsaklis and A. Nerode, "Hybrid control systems: An introductory discussion to the special issue," *IEEE Trans. Automat. Contr.*, vol. 43, pp. 457–460, Apr. 1998. (Special Issue on Hybrid Control Systems).

[12] P. Antsaklis and K. Passino, Eds., *An Introduction to Intelligent and Autonomous Control*. Norwell, MA: Kluwer, 1993.

[13] P. Antsaklis and K. Passino, "Introduction to intelligent control systems with high degrees of autonomy," in *An Introduction to Intelligent and Autonomous Control*, P. Antsaklis and K. Passino, Eds. Norwell, MA: Kluwer, 1993, pp. 1–26.
[14] P. Antsaklis, K. Passino, and S. Wang, "Toward intelligent autonomous control systems: Architecture and fundamental issues," *J. Intell. Robot. Syst.*, vol. 1, pp. 315–3423, 1989.
[15] P. Antsaklis, J. Stiver, and M. Lemmon, "Hybrid system modeling and autonomous control systems," in *Hybrid Systems*, R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds. Berlin, Germany: Springer-Verlag, 1993, vol. 736, Lecture Notes in Computer Science, pp. 366–392.
[16] M. Branicky, " Studies in hybrid systems: Modeling, analysis, and control," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, 1995.
[17] M. Branicky, V. Borkar, and S. Mitter, "A unified framework for hybrid control: Model and optimal control theory," *IEEE Trans. Automat. Contr.*, vol. 43, no. 1, pp. 31–45, 1998.
[18] P. Caines and Y.-J. Wei, "Hierarchical hybrid control systems: A lattice formulation," *IEEE Trans. Automat. Contr.*, vol. 43, no. 4, pp. 501–508, 1998.
[19] C. Cassandras, S. Lafortune, and G. Olsder, "Introduction to the modeling, control and optimization of discrete event systems," in *Trends in Control. A European Perspective*, A. Isidori, Ed. Berlin, Germany: Springer-Verlag, 1995, pp. 217–291.
[20] J. Cury, B. Krogh, and T. Niinomi, "Synthesis of supervisory controllers for hybrid systems based on approximating automata," *IEEE Trans. Automat. Contr.*, vol. 43, no. 4, pp. 564–568, 1998.
[21] I. Demongodin and N. Koussoulas, "Differential Petri nets: Representing continuous systems in a discrete-event world," *IEEE Trans. Automat. Contr.*, vol. 43, no. 4, pp. 573–579, 1998.
[22] S. DiGennaro, C. Horn, S. Kulkarni, and P. Ramadge, "Reduction of timed hybrid systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 8, no. 4, pp. 343–351, 1998.
[23] K. He and M. Lemmon, "Modeling hybrid control systems using programmable timed Petri nets," *Eur. J. Automat.*, vol. 32, no. 9–10, pp. 1187–1208, 1998.
[24] T. Henzinger, "The theory of hybrid automata," in *Proc. 11th Annu. Symp. Logic in Computer Science*, 1996, pp. 278–292.
[25] M. Heymann and F. Lin, "Discrete-event control of nondeterministic systems," *IEEE Trans. Automat. Contr.*, vol. 43, no. 1, pp. 3–17, 1998.
[26] J. E. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison-Wesley, 1979.
[27] A. Isidori, *Nonlinear Control Systems*, 2nd ed. Berlin, Germany: Springer-Verlag, 1996.
[28] X. Koutsoukos and P. Antsaklis, "Design of hybrid system regulators," in *Proc. 38th IEEE Conf. Decision and Control*, Phoenix, AZ, Dec. 1999, pp. 3990–3995.
[29] ——, "Hybrid control of a robotic manufacturing system," in *Proc. 7th IEEE Mediterranean Conf. Control and Automation*, Haifa, Israel, June 1999, pp. 144–159.
[30] X. Koutsoukos, P. Antsaklis, K. He, and M. Lemmon, "Programmable timed Petri nets in the analysis and design of hybrid control systems," in *Proc. 37th IEEE Conf. Decision and Control*, Tampa, FL, Dec. 1998, pp. 1617–1622.
[31] X. Koutsoukos, K. He, M. Lemmon, and P. Antsaklis, "Timed Petri nets in hybrid systems: Stability and supervisory control," *J. Discrete Event Dyn. Syst.*, vol. 8, no. 2, pp. 137–173, 1998.
[32] R. Kumar and M. Shayman, "Non-blocking supervisory control of nondeterministic systems under partial observation and decentralization," *IEEE Trans. Automat. Contr.*, vol. 41, no. 8, pp. 1160–1175, 1996.
[33] G. Lafferriere, G. Pappas, and S. Sastry, "Reachability analysis of hybrid systems using bisimulations," in *Proc. 37th IEEE Conf. Decision and Control*, Tampa, FL, 1998, pp. 1623–1628.
[34] ——, "Hybrid systems with finite bisimulations," in *Hybrid Systems V*, P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, Eds. Berlin, Germany: Springer-Verlag, 1999, vol. 1567, Lecture Notes in Computer Science, pp. 186–203.
[35] J. LaSalle and S. Lefschetz, *Stability by Lyapunov's Direct Method*. New York: Academic, 1961.
[36] M. Lemmon and P. Antsaklis, "Inductively inferring valid logical models of continuous-state dynamical systems," *Theoret. Comput. Sci.*, vol. 138, pp. 201–210, 1995.
[37] M. Lemmon, K. He, and I. Markovsky, "Supervisory hybrid systems," *IEEE Contr. Syst. Mag.*, vol. 19, pp. 42–55, Aug. 1999.
[38] D. Liberzon and A. Morse, "Basic problems in stability and design of switched systems," *IEEE Contr. Syst. Mag.*, vol. 19, no. 5, pp. 59–70, Oct. 1999.
[39] J. Lunze, "Qualitative modeling of linear dynamical systems with quantised state measurements," *Automatica*, vol. 30, no. 3, pp. 417–431, 1994.
[40] J. Lunze, B. Nixdorf, and J. Schroder, "Deterministic discrete-event representations of linear continuous-variable systems," *Automatica*, vol. 35, no. 3, pp. 396–406, 1999.
[41] N. Lynch, R. Segala, F. Vaandrager, and H. Weinberg, "Hybrid I/O automata," in *Hybrid Systems III, Verification and Control*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds. Berlin, Germany: Springer-Verlag, 1996, vol. 1066, pp. 496–510.
[42] N. McClamroch, C. Rui, I. Kolmanovsky, and M. Reyhanoglu, "Hybrid closed loop systems: A nonlinear control perspective," in *Proc. 36th IEEE Conf. Decision and Control*, 1997, pp. 114–119.
[43] J. Moody and P. Antsaklis, *Supervisory Control of Discrete Event Systems using Petri Nets*. Norwell, MA: Kluwer Academic, 1998.
[44] M. Morari and E. Zafiriou, *Robust Process Control*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[45] A. Morse, "Supervisory control of families of linear set-point controllers—Part 1: Exact matching," *IEEE Trans. Automat. Contr.*, vol. 41, pp. 1271–1281, 1996.
[46] A. Nemirovsky and D. Yudin, *Problem Complexity and Method Efficiency in Optimization*. New York: Wiley, 1983.
[47] A. Nerode and W. Kohn, "Models for hybrid systems: Automata, topologies, controllability, observability," in *Hybrid Systems*, R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds. Berlin, Germany: Springer-Verlag, 1993, vol. 736, Lecture Notes in Computer Science, pp. 317–356.
[48] ——, "Multiple agent hybrid control architecture," in *Hybrid Systems*, R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds. Berlin, Germany: Springer-Verlag, 1993, vol. 736, Lecture Notes in Computer Science, pp. 297–316.
[49] H. Nijmeijer and A. van der Schaft, *Nonlinear Dynamical Control Systems*. Berlin, Germany: Springer-Verlag, 1990.
[50] A. Overkamp, "Supervisory control using failure semantics and partial observations," *IEEE Trans. Automat. Contr.*, vol. 42, no. 4, pp. 498–510, 1997.
[51] C. Özveren and A. Willsky, "Observability of discrete event dynamic systems," *IEEE Trans. Automat. Contr.*, vol. 35, no. 7, pp. 797–806, 1990.
[52] G. Pappas, G. Lafferriere, and S. Sastry, "Hierarchically consistent control systems," in *Proc. 37th IEEE Conf. Decision and Control*, Tampa, FL, 1998, pp. 4336–4341.
[53] A. Puri, V. Borkar, and P. Varaiya, "ϵ-approximation of differential inclusions," in *Hybrid Systems III, Verification and Control*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds. Berlin, Germany: Springer-Verlag, 1996, vol. 1066, Lecture Notes in Computer Science, pp. 362–376.
[54] J. Raisch, E. Klein, S. O'Young, C. Meder, and A. Itigin, "Approximating automata and discrete control for continuous systems—Two examples from process control," in *Hybrid Systems V*, P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, Eds. Berlin, Germany: Springer-Verlag, 1999, vol. 1567, Lecture Notes in Computer Science, pp. 279–303.
[55] J. Raisch and S. O'Young, "A totally ordered set of discrete abstractions for a given hybrid system," in *Hybrid Systems IV*, P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds. Berlin, Germany: Springer-Verlag, 1997, vol. 1273, Lecture Notes in Computer Science, pp. 342–360.
[56] J. Raisch and S. O'Young, "Discrete approximation and supervisory control of continuous systems," *IEEE Trans. Automat. Contr.*, vol. 43, no. 4, pp. 568–573, 1998.
[57] P. Ramadge and W. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Contr. Optim.*, vol. 25, no. 1, pp. 206–230, Jan. 1987.
[58] ——, "The control of discrete event systems," *Proc. IEEE*, vol. 77, pp. 81–89, Jan. 1989.
[59] J. Stiver, "Analysis and design of hybrid control systems," Ph.D. dissertation, Dept. Elect. Eng., Univ. Notre Dame, Notre Dame, IN, 1995.
[60] J. Stiver and P. Antsaklis, "A novel discrete event system approach to modeling and analysis of hybrid control systems," in *Proc. 29th Annu. Allerton Conf. Communication, Control and Computing*, Oct. 2–4, 1991.

[61] J. Stiver, P. Antsaklis, and M. Lemmon, "Digital control from a hybrid perspective," in *Proc. 33rd IEEE Conf. Decision and Control*, Lake Buena Vista, FL, Dec. 1994, pp. 4241–4246.
[62] ——, "Hybrid control system design based on natural invariants," in *Proc. 34th IEEE Conf. Decision and Control*, New Orleans, LA, Dec. 1995, pp. 1455–1460.
[63] ——, "Interface and controller design for hybrid control systems," in *Hybrid Systems II*, P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds. Berlin, Germany: Springer-Verlag, 1995, vol. 999, Lecture Notes in Computer Science, pp. 462–492.
[64] ——, "An invariant based approach to the design of hybrid control systems," in *IFAC 13th Triennial World Congr.*, vol. J, San Francisco, CA, 1996, pp. 467–472.
[65] ——, "A logical DES approach to the design of hybrid control systems," *Math. Comput. Modeling*, vol. 23, no. 11/12, pp. 55–76, 1996.
[66] M. Tittus and B. Egardt, "Control design for integrator hybrid system," *IEEE Trans. Automat. Contr.*, vol. 43, no. 4, pp. 491–500, 1998.
[67] J. Willems, "Paradigms and puzzles in the theory of dynamical systems," *IEEE Trans. Automat. Contr.*, vol. 36, no. 3, pp. 259–294, 1991.
[68] W. Wonham and P. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Contr. Optim.*, vol. 25, pp. 637–659, May 1987.
[69] X. Yang, M. Lemmon, and P. Antsaklis, "On the supremal controllable sublanguage in the discrete-event model of nondeterministic hybrid control systems," *IEEE Trans. Automat. Contr.*, vol. 40, no. 12, pp. 2098–2103, 1995.
[70] H. Ye, A. Michel, and L. Hou, "Stability theory for hybrid dynamical systems," *IEEE Trans. Automat. Contr.*, vol. 43, no. 4, pp. 461–474, 1998.
[71] F. Zhao, "Extracting and representing qualitative behaviors of complex systems in phase spaces," *Artif. Intell.*, vol. 369, no. 1–2, pp. 51–92, 1994.

Xenofon D. Koutsoukos was born in Athens, Greece, in 1969. He received the Dipl. degree in electrical and computer engineering from the National Technical University of Athens in 1993. He received the M.S. degrees in electrical engineering and applied mathematics in 1998, and the Ph.D degree in electrical engineering, all from the University of Notre Dame, Notre Dame, IN.

From 1993 to 1995, he was with the National Center for Space Applications, Hellenic Ministry of National Defense, Athens, as a Computer Engineer in the areas of image processing and remote sensing. He was a graduate student fellow of the Center of Applied Mathematics, University of Notre Dame, for the academic year 1997–98. His research interests include hybrid systems, discrete event systems, and intelligent control systems.

Panos J. Antsaklis (Fellow, IEEE) received the undergraduate degree from the National Technical University of Athens (NTUA), Greece, and the M.S. and Ph.D. degrees from Brown University, Providence, RI.

He is Professor of electrical engineering and Director of the Center for Applied Mathematics at the University of Notre Dame, Notre Dame, IN. He has held faculty positions at Brown University, Rice University, and Imperial College of the University of London. During sabbatical leaves, he has lectured and conducted research at the Massachusetts Institute of Technology, Imperial College, NTUA, and the Technical University of Crete, Greece. His research interests are in the area of systems and control, with emphasis on hybrid and discrete event systems, and on autonomous, intelligent, and learning control systems. He has authored a number of publications in journals, conference proceedings, and books, and he has edited four books on intelligent autonomous control and on hybrid systems. He co-authored, with J. Moody, the research monograph *Supervisory Control of Discrete Event Systems Using Petri Nets* (Norwell, MA: Kluwer, 1998) and, with A. N. Michel, the graduate textbook *Linear Systems* (New York: McGraw-Hill, 1997). He serves on the editorial boards of several journals, and has been the Guest Editor of *Journal of Discrete Event Dynamic Systems* (1998). He has served as Program Chair and General Chair of major systems and control conferences, and was the 1997 President of the IEEE Control Systems Society (CSS).

Dr. Antsaklis is an IEEE Third Millennium Medal recipient. He has been Guest Editor of special issues on Neural Networks (*IEEE Control Systems Magazine*; 1990 and 1992), on Intelligence and Learning (*IEEE Control Systems Magazine*; 1995), and on Hybrid Control Systems (IEEE TRANSACTIONS ON AUTOMATIC CONTROL; 1998).

James A. Stiver received the B.S and Ph.D. degrees in electrical engineering from the University of Notre Dame, Notre Dame, IN.

He is currently employed with Harris Corporation, Melbourne, FL, where he is involved in the analysis and design of satellite communications systems and serves as the Leader of the Controls Analysis Group. His research interests are in the area of mixed discrete and continuous control systems.

Michael D. Lemmon (Member, IEEE) received the B.S. degree in electrical engineering from Stanford University, Stanford, CA, in 1979, and the M.S. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1987 and 1990, respectively.

Since 1990, he has been affiliated with the University of Notre Dame, Notre Dame, IN, where he is currently Associate Professor of electrical engineering. His research has promoted the use of hybrid dynamical systems theory as a foundation for the study of intelligent control systems. He chaired the IEEE Working Group on Hybrid Dynamical Systems from 1996 to 2000. He was program chair for the 5th International Workshop on Hybrid Systems (1997) and the 1999 International Symposium on Intelligent Control.

Dr. Lemmon is a former Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS and is currently an Associate Editor for the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY.