Hybrid System Control

Panos J. Antsaklis
University of Notre Dame

Xenofon D. Koutsoukos
Xerox Palo Alto Research Center

GLOSSARY

Discrete event systems Dynamical systems where the variables of interest are described by sequences of discrete events. Discrete event system models may be used, for example, to describe sequences of operations on parts moving among different machines in a manufacturing process. Discrete event systems may be represented by finite automata or Petri net mathematical models

Hybrid Heterogeneous in nature or composition.

Hybrid automata A general modeling formalism for hybrid systems that describes the discrete dynamics using finite automata and the continuous dynamics using differential/difference equations.

Hybrid systems Dynamical systems with behavior defined by entities or processes of distinct characteristics and, in particular, by processes that contain continuous dynamics and processes that contain discrete dynamics.

Supervisory control The process by which a discrete dynamical system's behavior is controlled not by forcing events but by restricting or disabling certain events from taking place.

HYBRID SYSTEMS are dynamical systems that consist of subsystems with continuous dynamics and subsystems with discrete dynamics that interact with each other. Such hybrid systems arise in varied contexts in manufacturing, communication networks, autopilot design, automotive engine control, computer synchronization, traffic control, and chemical processes, among others. Hybrid systems have a central role in embedded control systems that interact with the physical world. They also arise from the hierarchical organization of complex systems and from the interaction of discrete planning algorithms and continuous control algorithms in autonomous, intelligent systems. There has been significant research activity in the area of hybrid systems in the past decade, and significant progress in theory and algorithm development has been reported. Good software tools for the simulation, analysis, and design of hybrid systems, which by their nature are complex systems, are very important. Researchers have recognized this need and several software packages have been developed. In this chapter, a brief introduction to the theory and applications of hybrid systems is presented. Additional details may be found in the references included in the Bibliography.

I. HYBRID SYSTEM MODELS

A hybrid control system is a system in which the behavior of interest is determined by interacting processes of distinct characteristics, in particular, interacting continuous and discrete dynamics. Hybrid systems typically generate mixed signals that consist of combinations of continuous and discrete-valued signals. Some of these signals take values from a continuous set (e.g., the set of real numbers) and others take values from a discrete, typically finite set (e.g., the set of symbols $\{a, b, c\}$). Furthermore, these continuous or discrete-valued signals depend on independent variables such as time, which may also be continuous or discrete-valued. Another distinction that can be made is that some of the signals can be time-driven, while others can be event-driven in an asynchronous manner.

The dynamic behavior of such hybrid systems is captured in hybrid models. In a manufacturing process, for example, parts may be processed in a particular machine, but only the arrival of a part triggers the process; that is, the manufacturing process is composed of the event-driven dynamics of the parts moving among different machines and the time-driven dynamics of the processes within particular machines. Frequently in hybrid systems in the past, the event-driven dynamics were studied separately from the time-driven dynamics, the former via automata or Petri net models (also via PLC, logic expressions, etc.) and the latter via differential or difference equations. To understand fully the system's behavior and meet high-performance specifications, one needs to model all dynamics together with their interactions. Only then may problems such as optimization of the whole manufacturing process be addressed in a meaningful manner. There are, of course, cases where the time-driven and event-driven dynamics are not tightly coupled or the demands on the system performance are not difficult to meet, and in those cases considering simpler separate models for the distinct phenomena may be adequate. However, hybrid models must be used when there is significant interaction between the continuous and the discrete parts and high-performance specifications are to be met by the system.

Hybrid models may be used to significant advantage, for example, in automotive engine control, where there is a need for control algorithms with guaranteed properties, implemented via embedded controllers, that can substantially reduce emissions and gas consumption while maintaining the performance of the car. Note that an accurate model of a four-stroke gasoline engine has a natural hybrid representation, because from the engine control point of view, on one hand, the power train and air dynamics are continuous-time processes, while, on the other hand, the pistons have four modes of operation that correspond to the stroke they are in and so their behavior is represented as a discrete-event process represented, say, via a finite-state machine model. These processes interact tightly, as the timing of the transitions between two phases of the pistons is determined by the continuous motion of the power train, which, in turn, depends on the torque produced by each piston. Note that in the past the practice has been to convert the discrete part of the engine behavior into a more familiar and easier-to-handle continuous model, where only the average values of the appropriate physical quantities are modeled. Using hybrid models one may represent time- and event-based behaviors more accurately so as to meet challenging design requirements in the design of control systems for problems such as cutoff control and idle speed control of the engine. For similar reasons, that is, tight interaction of continuous and discrete dynamics and demands of high performance or the system, hybrid models are important in chemical processes, robotic manufacturing systems, transportation systems, and air traffic control systems among many other applications.

There are other ways in which hybrid systems may arise. Hybrid systems arise from the interaction of discrete planning algorithms and continuous processes, and as such, they provide the basic framework and methodology for the analysis and synthesis of autonomous and intelligent systems. In fact, the study of hybrid systems is essential in designing sequential supervisory controllers for continuous systems, and it is central in designing intelligent control systems with a high degree of autonomy. Another important way in which hybrid systems arise is from the hierarchical organization of complex systems. In these systems, a hierarchical organization helps manage complexity, and higher levels in the hierarchy require less detailed models (discrete abstractions) of the functioning of the lower levels, necessitating the interaction of discrete and continuous components. Examples of such systems include flexible manufacturing and chemical process control systems, interconnected power systems, intelligent highway systems, air traffic management systems, and computer and communication networks.

In the control systems area, a very well-known instance of a hybrid system is a sampled-data or digital control system. Therein, a system described by differential equations, which involve continuous-valued variables that depend on continuous time, is controlled by a discrete-time controller described by difference equations, which involve continuous-valued variables that depend on discrete time;

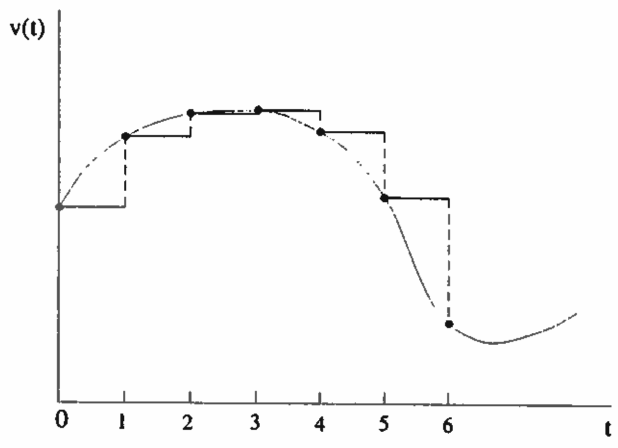

FIGURE 1 A signal, its samples, and its approximation by a zero-order hold.

see, for example, Fig. 1. If one also considers quantization of the continuous-valued variables or signals, then the hybrid system contains not only continuous-valued variables that are driven by continuous and discrete times, but also discrete-valued signals. Another example of a hybrid control system is a switching system, where the dynamic behavior of interest can be adequately described by a finite number of dynamical models, which are typically sets of differential or difference equations, together with a set of rules for switching among these models. These switching rules are described by logic expressions or a discrete-event system with a finite automaton or a Petri net representation.

A familiar simple example of a practical hybrid control system is the heating and cooling system of a typical home. The furnace and air-conditioner, along with the heat flow characteristics of the home, form a continuous-time system, which is to be controlled. The thermostat is a simple asynchronous discrete-event system (DES), which basically handles the symbols {*too hot, too cold*} and {*normal*}. The temperature of the room is translated into these representations in the thermostat and the thermostat's response is translated back to electrical currents, which control the furnace, air-conditioner, blower, etc.

There are several reasons for using hybrid models to represent the dynamic behavior of interest in addition to the ones already mentioned. Reducing complexity was and still is an important reason for dealing with hybrid systems. This is accomplished in hybrid systems by incorporating models of dynamic processes at different levels of abstraction; for example, the thermostat in the above example is a very simple model, but adequate for the task at hand, of the complex heat flow dynamics. For another example, to avoid dealing directly with a set of nonlinear equations, one may choose to work with sets of simpler equations (e.g., linear) and switch among these simpler models. This is a rather common approach in modeling physical phenomena. In control, switching among simpler dynamical systems has been used successfully in practice for many decades. Recent efforts in hybrid system research along these lines typically concentrate on the analysis of the dynamic behaviors and aim to design controllers with guaranteed stability and performance.

Hybrid systems have been important for a long time. The recent interest and activity in hybrid systems have been motivated in part by the development of research results on the control of DESs that occurred in the 1980s and on adaptive control in the 1970s and 1980s and by the renewed interest in optimal control formulations in sampled-data systems and digital control. In parallel developments, there has been growing interest in hybrid systems among computer scientists and logicians, with an emphasis on verification of the design of computer software. Whenever the behavior of a computer program depends on values of continuous variables within that program (e.g., continuous time clocks), one needs hybrid system methodologies to guarantee the correctness of the program. In fact the verification of such digital computer programs has been one of the main goals of several serious research efforts in the hybrid system literature. Note that efficient verification methodologies are essential for complex hybrid systems to be useful in applications. The advent of digital machines has made hybrid systems very common indeed. Whenever a digital device interacts with the continuous world, the behavior involves hybrid phenomena that need to be analyzed and understood. It should be noted that certain classes of hybrid systems have been studied in related research areas such as variable structure control, sliding mode control, and bang-bang control.

Hybrid systems represent a highly challenging area of research that encompasses a variety of challenging problems that may be approached at varied levels of detail and sophistication. Modeling of hybrid systems is very important, as modeling is in every scientific and engineering discipline. Different types of models are used, from detailed models that may include equations and lookup tables that are excellent for simulation but not easily amenable to analysis, to models that are also good for analysis but not easily amenable to synthesis, models for control, models for verification, and so on.

II. APPROACHES TO THE ANALYSIS AND DESIGN OF HYBRID SYSTEMS

Current approaches to hybrid systems differ with respect to the emphasis on or the complexity of the continuous and discrete dynamics and in whether they emphasize analysis and synthesis results, or analysis only, or simulation only. On one end of the spectrum there are approaches to hybrid systems that represent extensions of system theoretic ideas for systems (with continuous-valued variables and continuous time) that are described by ordinary differential equations to include discrete time and variables that exhibit jumps or extend results to switching systems. Typically these approaches are able to deal with complex continuous dynamics. Their main emphasis has been on the stability of systems with discontinuities. On the other end of the spectrum there are approaches to hybrid systems embedded in computer science models and methods that represent extensions of verification methodologies from discrete systems to hybrid systems. Typically these approaches are able to deal with discrete dynamics described by finite automata and emphasize analysis results (verification) and simulation methodologies. There are additional methodologies spanning the rest of the spectrum that combine concepts from continuous control systems described by linear and nonlinear differential/difference equations, and from supervisory control of DESs that are described by finite automata and Petri nets to derive, with varying success, analysis and synthesis results.

Several approaches to the modeling, analysis, and synthesis of hybrid systems are described in this chapter. They are organized into three sections. In Section III, the supervisory control approach to the analysis and design of hybrid control systems is described. Then, in Section IV, approaches that emphasize stability are presented. Finally, approaches based on hybrid automata models are discussed in Section V.

III. SUPERVISORY CONTROL OF HYBRID SYSTEMS

In the 1980s systems with discrete dynamics such as manufacturing systems attracted the attention of the control research community, and models such as finite automata were used to describe such discrete-event dynamical systems. Important system properties such as controllability, observability, and stability were defined and studied for discrete event systems (DESs) and methodologies for supervisory control design were developed. In related developments, the relation between inherently discrete planning systems and continuous-feedback control systems attracted attention. In addition to finite automata, other modeling paradigms such as Petri nets gained the attention of control and automation system researchers in the last decade, primarily in Europe. Petri nets have been used in the supervisory control of discrete-event dynamic systems as an attractive alternative to methodologies based on finite automata.

In this section, we review the supervisory control framework for hybrid systems. One of the main characteristics

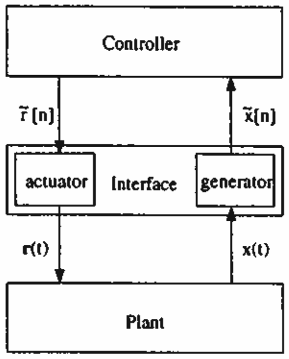

FIGURE 2 Hybrid system model in the supervisory control framework.

of this approach is that the system to be controlled is approximated by a DES and the design is carried out in the discrete domain. The hybrid control systems in the supervisory control framework 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 as shown in Fig. 2. It is generally assumed that the dynamic behavior of the plant is governed by a set of known nonlinear ordinary differential equations,

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

where $x \in \Re^n$ is the continuous state of the system and $r \in \Re^m$ is the continuous control input. In the model shown in Fig. 2, the plant contains all continuous components of the hybrid control system, such as any conventional continuous controllers 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 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.

The interface consists of the generator and the actuator as shown in Fig. 2. The generator has been chosen to be a partitioning of the state space (see Fig. 3). The piecewise continuous command signal issued by the actuator is a

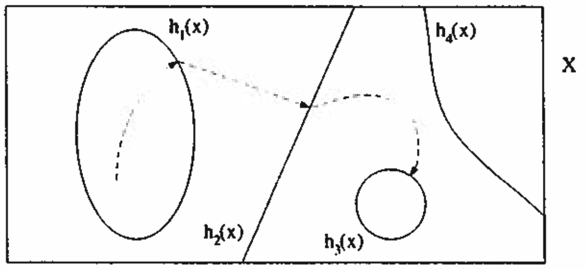

FIGURE 3 Partition of the continuous state space.

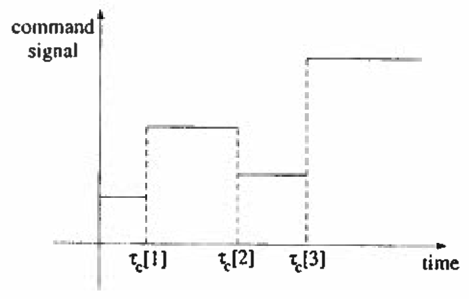

FIGURE 4 Command signal issued by the interface.

staircase signal as shown in Fig. 4, not unlike the output of a zero-order hold in a digital control system. The interface plays a key role in determining the dynamic behavior of the hybrid control system. Many times the partition of the state space is determined by physical constraints and it is fixed and given. Methodologies for the computation of the partition based on the specifications have also been developed.

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

The DES plant model is an approximation of the actual system and its behavior is an abstraction of the system's behavior. As a result, the future behavior of the actual continuous system cannot be determined uniquely, in general, from knowledge of the DES plant state and input. The approach taken in the supervisory control framework is to incorporate all the possible future behaviors of the continuous plant into the DES plant model. A conservative approximation of the behavior of the continuous plant is constructed and realized by a finite state machine. 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 will be eliminated from the actual system. On the other hand, just because a control policy permits a given behavior in the DES plant is no guarantee that the behavior will occur in the actual system.

We briefly discuss 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 $T \subseteq \Re$ the *time axis*, W the *signal space*, and $B \subset W^T$ (denoting the set of all functions $f : T \rightarrow W$) the *behavior*. The behavior of the DES plant model consists of all the pairs of plant

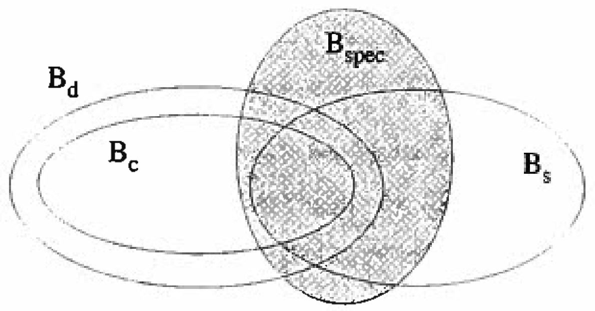

FIGURE 5 The DES plant model as an approximation.

and control symbols that it can generate. The time axis T represents here the occurrences of events. 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., $B_c \subseteq B_d$.

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_{spec}. 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

$$B_d \cap B_s \subseteq B_{spec} \Rightarrow B_c \cap B_s \subseteq B_{spec} \qquad (2)$$

and is depicted in Fig. 5. The challenge is to find a discrete abstraction with behavior B_d which is a approximation of the behavior B_c of the continuous plant and for which it is possible to design a supervisor 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 space for the extraction of the DES plant.

An interesting aspect of the DES plant's behavior is that it is distinctly nondeterministic. This fact is illustrated in Fig. 6, which shows two trajectories generated by the same

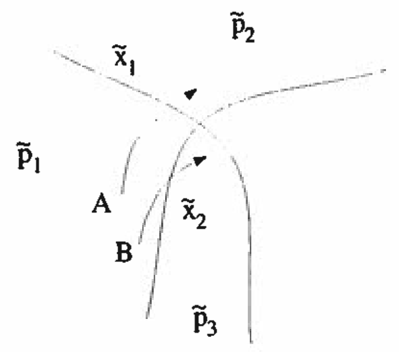

FIGURE 6 Nondeterminism of the DES plant model.

control symbol. Both trajectories originate in the same DES plant state $\tilde{p}_1$. Figure 6 shows that for a given control symbol, there are at least two possible DES plant states that can be reached from $\tilde{p}_1$. 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.

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 which 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 which 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.

A language theoretic framework to describe performance specifications for hybrid systems and to formulate the supervisory control problem has been developed. Once the DES plant model of a hybrid system has been extracted, a supervisor can be designed using control synthesis techniques based on DESs. The main differences are that the DES plant models of the hybrid control framework are nondeterministic and that the plant events cannot be disabled individually.

Example. The hybrid system in this example consists of a typical thermostat and furnace. Assuming that the thermostat is set at 70°F, the system behaves as follows. If the room temperature falls below 70°F, the furnace starts and remains on until the room temperature reaches 72°F. At 72°F, the furnace shuts off. For simplicity, we 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:

$$\dot{x} = 0.0042(T_0 - x) + 0.1r, \tag{3}$$

where the plant state, x, is the temperature of the room in degrees Fahrenheit, the input, r, is the voltage on the furnace control circuit, and T_0 is the outside temperature. This model of the furnace is certainly a simplification, but it is adequate for this example.

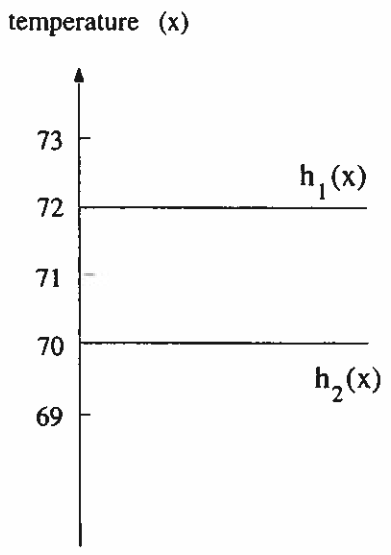

FIGURE 7 Partition for the thermostat/furnace system.

The thermostat partitions the state space of the plant with the following hypersurfaces as shown in Fig. 7.

$$h_1(x) = x - 72, \tag{4}$$

$$h_2(x) = 70 - x. \tag{5}$$

The first hypersurface detects when the state exceeds 72°F, and the second detects when the state falls below 70°F. The associated generator functions, α_1 and α_2, are very simple in this case.

$$\alpha_i(x) = \tilde{x}_i. \tag{6}$$

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

The DES controller is shown in Fig. 8. The output function of the controller is defined as

$$\phi(\tilde{s}_1) = \tilde{r}_1 \Leftrightarrow \text{off}, \tag{7}$$

$$\phi(\tilde{s}_2) = \tilde{r}_2 \Leftrightarrow \text{on}, \tag{8}$$

and the actuator operates as

$$\gamma(\tilde{r}_1) = 0, \tag{9}$$

$$\gamma(\tilde{r}_2) = 12, \tag{10}$$

where the constants for the control inputs correspond to particular given data.

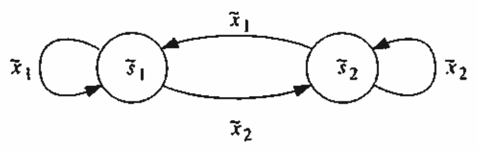

FIGURE 8 Controller for the thermostat/furnace system.

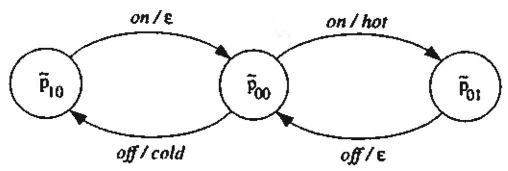

FIGURE 9 DES plant for the thermostat/furnace system.

The thermostat/heater example has a simple DES plant model which is useful to illustrate how these models work. Figure 9 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 slash and then the plant symbols which can be generated by the transition. Notice that two of the transitions are labeled with null symbols, ϵ. This reflects the fact that nothing actually happens in the system at these transitions. When the controller receives a null symbol it remains in the same state and reissues the current controller symbol. This is equivalent to the controller doing nothing, but it serves to keep all the symbolic sequences, $\tilde{s}$, $\tilde{p}$, etc., in phase with each other.

IV. STABILITY AND DESIGN OF HYBRID SYSTEMS

In the area of control systems, powerful methodologies for analysis of properties such as stability and systematic methodologies for the design of controllers have been developed over the years. Some of the methodologies have been extended to hybrid systems, primarily to switched systems. Switched systems are hybrid dynamical systems that consist of a family of continuous or discrete-time subsystems and a rule that determines the switching between them. The switching behavior of these systems may be generated by the changing dynamics at different operating regions. Hybrid dynamical systems also arise when switching controllers are used to achieve stability and improve performance as shown in Fig. 10. Typical examples of such systems are computer disk drives, constrained mechanical systems, switching power converters, and automotive powertrain applications.

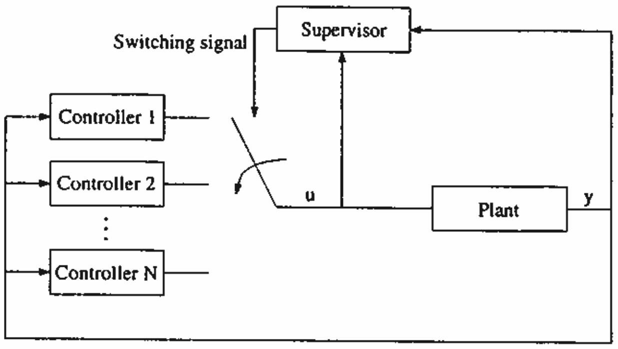

FIGURE 10 Switching controller feedback architecture.

Mathematically, such hybrid systems can be modeled by the equations

$$\dot{x} = f(x(t), q(t), u(t)), \tag{11}$$

$$q(t^+) = \delta(x(t), q(t)), \tag{12}$$

where $x(t) \in \Re^n$ is the continuous state, $q(t) \in \{1, 2, \ldots, N\}$ is the discrete state that indexes the subsystems $f_{q(t)}$, $u(t)$ can be a continuous control input or an external (reference or disturbance) signal to the continuous part, and δ is the switching law that describes the logical and/or DES dynamics.

Example. This example describes a simplified model of a car with an automatic transmission. Let m denote the mass of the car and v the velocity on a road inclined at an angle α. The simplified dynamics of the car are described by

$$\dot{v} = -\frac{k}{m} v^2 \operatorname{sign}(v) - g \sin\alpha + \frac{G_{q(t)}}{m} T, \tag{13}$$

$$\omega = G_{q(t)} v, \tag{14}$$

where G_i, $i = 1, 2, 3, 4$, are the transmission gear ratios normalized by the wheel radius R, k is an appropriate constant, ω is the angular velocity of the motor, and T is the torque generated by the engine. The dynamic behavior of the car is indexed by the discrete state q. The discrete-state transition function which determines the switching between the gears is

$$q(t^+) = \begin{cases} i+1, & \text{if } q(t) = i \neq 4 \quad \text{and} \quad v = \frac{1}{G_i}\omega_{\text{high}}, \\ i-1 & \text{if } q(t) = i \geq 2 \quad \text{and} \quad v = \frac{1}{G_i}\omega_{\text{low}}, \end{cases} \tag{15}$$

where ω_{high} and ω_{low} are prescribed angular velocities of the engine. The discrete-state transition can be described for the finite automaton in Fig. 11. □

Hybrid system stability analysis relies for the most part on classical Lyapunov stability theory. For conventional control systems, demonstrating stability depends on the existence of a continuous and differentiable Lyapunov (energy) function. In the hybrid system case, stability analysis is carried out using multiple Lyapunov functions (MLFs) to compose a single piecewise continuous and piecewise differentiable Lyapunov function that can be used to demonstrate stability. To illustrate the use of MLFs, we consider the autonomous form $[u(t) = 0]$ of the hybrid system model

$$\dot{x}(t) = f(x(t), q(t)) = f_{q(t)}(x(t)), \tag{16}$$

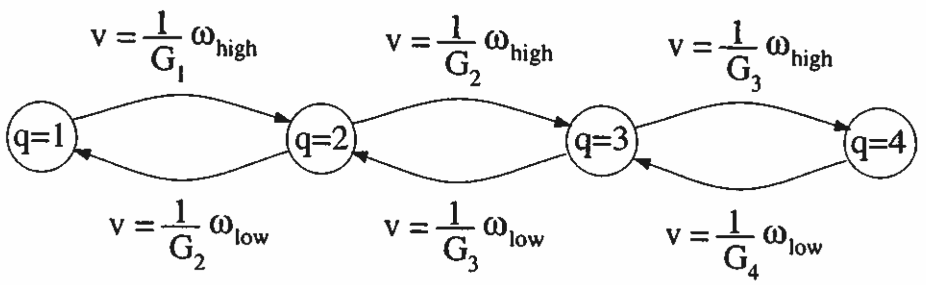
v = $\frac{1}{G_1}\omega_{high}$
v = $\frac{1}{G_2}\omega_{high}$
v = $\frac{1}{G_3}\omega_{high}$
q=1
q=2
q=3
q=4
v = $\frac{1}{G_2}\omega_{low}$
v = $\frac{1}{G_3}\omega_{low}$
v = $\frac{1}{G_4}\omega_{low}$

FIGURE 11 Finite automaton illustrating the switching of the gears.

where $q(t) \in \{1, 2, \ldots, N\}$. In addition, it is assumed that there are only a finite number of switchings in a bounded time interval. It should be noted that hybrid systems that exhibit infinitely many switchings in a finite interval are called *Zeno* systems.

Consider the family of Lyapunov-like functions $\{V_i, i = 1, 2, \ldots, N\}$, where each V_i is associated with the subsystem $f_i(x)$. A *Lyapunov-like* function for the system $\dot{x} = f_i(x)$ and equilibrium point $\bar{x} \in \Omega_i \subset \Re^n$ is a real-valued function $V_i(x)$ defined over the region Ω_i which is *positive definite* [$V_i(\bar{x}) = 0$ and $V_i(x) > 0$ for $x \neq \bar{x}$, $x \in \Omega_i$] and has *negative semidefinite derivative* (for $x \in \Omega_i$, $\dot{V}_i(x) \leq 0$).

Given system (16), suppose that each subsystem f_i has an associate Lyapunov-like function V_i in the region Ω_i, each with equilibrium $\bar{x} = 0$, and suppose that $\bigcup_i \Omega_i = \Re^n$. Let $q(t)$ be a given switching sequence such that $q(t)$ can take on the value i only if $x(t) \in \Omega_i$, and in addition,

$$V_i(x(t_{i,k}) \leq V_i(x(t_{i,k-1}) \qquad (17)$$

where $t_{i,k}$ denotes the kth time the subsystem f_i is "switched in." Then system (16) is stable in the sense of Lyapunov. The stability condition (17) is illustrated in Fig. 12. At every instant the subsystem i becomes active, the corresponding energy function V_i decreases from the value it had the last time the subsystem i was switched in.

The general result presented in Fig. 12 gives sufficient conditions for stability. Implicitly, this result provides a methodology for switching between subsystems to achieve a stable trajectory. One strategy that may stabilize a hybrid system is to pick that subsystem that causes maximal descent of a particular energy function. Another strategy is to select the subsystem according to the Lyapunov function with the smallest value.

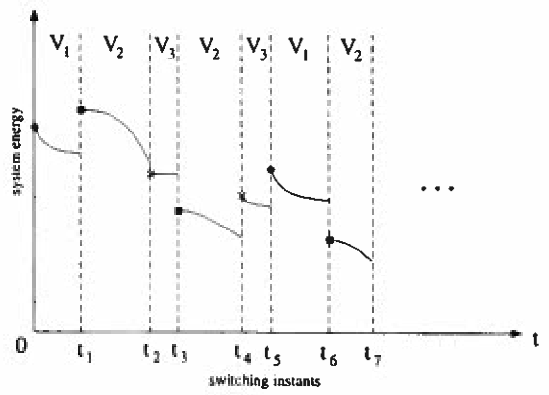

FIGURE 12 Stability condition.

In the following, the emphasis is put on linear switched systems described by

$$\dot{x} = A_{q(t)}x(t), \qquad (18)$$

$$q(t^+) = \delta(x(t), q(t)), \qquad (19)$$

where $q(t) \in \{1, 2, \ldots, N\}$ and $A_{q(t)} \in \Re^{n \times n}$. For this restricted class of hybrid systems, stronger results and systematic methodologies to construct multiple Lyapunov functions have been developed. An important observation is that it is possible for a linear switched system to be unstable even when all the subsystems are stable as illustrated in the following example. On the other hand, it is possible to stabilize a linear switched system even when all the subsystems are unstable.

Example. Consider the hybrid systems $\dot{x}(t) = A_{q(t)}x(t)$, where $x \in \Re^2$, $q \in \{1, 2\}$, and

$$A_1 = \begin{bmatrix} -1 & -100 \\ 10 & -1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} -1 & 10 \\ -100 & -1 \end{bmatrix}.$$

Both A_1 and A_2 have eigenvalues $\lambda_{1,2} = -1 \pm j\sqrt{1000}$ and therefore are stable. But the switched system using A_1 in the second and fourth quadrants and A_2 in the first and third quadrants is unstable as shown in Fig. 13. □

An important problem is to find conditions that guarantee that the switched system $\dot{x}(t) = A_{q(t)}x(t)$ is stable for any switching signal. This situation is of importance when a given plant is being controlled by switching among a family of stabilizing controllers, each of which is designed for a specific task. A supervisor determines which controller is to be connected in closed loop with the plant at each instant of time. Stability of the switched system can usually be ensured by keeping each controller in the loop long enough to allow the transient effects to dissipate. Another approach that can be used to demonstrate

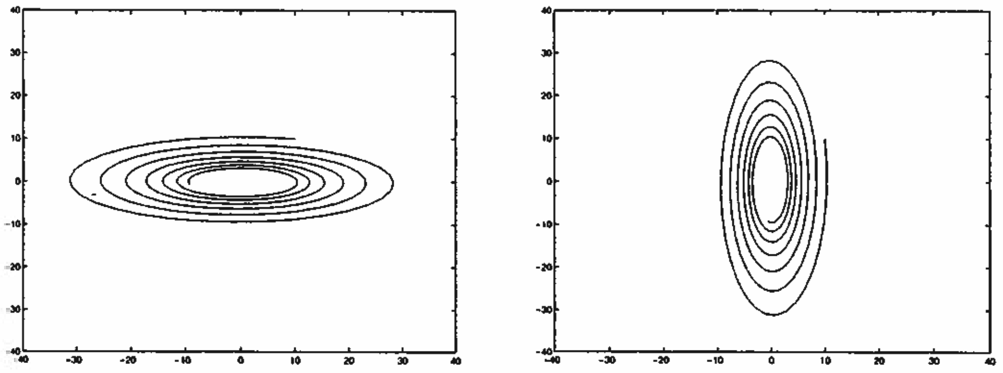

FIGURE 13 Unstable trajectory of a switched system consisting of stable subsystems.

stability for any switching signals is to guarantee that the matrices A_i share a common quadratic Lyapunov function $V(x) = x^T P x$, such that $\dot{V}(x) \leq -x^T Qx$, $Q > 0$ [Q is positive definite ($Q > 0$) when $x^T QX > 0$ for any $x \neq 0$]. These conditions on $V(x)$ are equivalent to finding matrices P and Q that satisfy the inequalities $A_i^T P + PA_i + Q \leq 0$ for all i. Note that the existence of a common Lyapunov function, although sufficient, is not necessary for stability.

The application of the theoretical results to practical hybrid systems is accomplished usually using a linear matrix inequality (LMI) problem formulation for constructing a set of quadratic Lyapunov-like functions. The existence of a solution to the LMI problem is a sufficient condition and guarantees that the hybrid system is stable.

The methodology begins with a partitioning of the state space into Ω-regions that are defined by quadratic forms. Physical insight, a good understanding of the LMI problem, and brute force are often required to choose an acceptable partitioning. Let Ω_i denote a region where one searches for a quadratic Lyapunov function $V_i = x^T P_i x$, $x \in \Omega_i$, that satisfies the condition

$$\dot{V}_i(x) = \left[\frac{\partial}{\partial x} V_i(x)\right] A_i x = x^T (A_i^T P_i + P_i A_i) x \leq 0. \tag{20}$$

The goal is to find matrices $P_i > 0$ that satisfy the above conditions. To constrain the stability conditions to local regions, two steps are involved. First, the region Ω_i must be expressed by the quadratic form $x^T Q_i x \geq 0$. Second, a technique called the S-procedure is applied to replace the constrained stability condition by a condition without constraints. By introducing a new unknown variable $\xi \geq 0$, the relaxed problem takes the unconstrained form

$$A_i^T P_i + P_i A + \xi Q_i \leq 0, \tag{21}$$

which can be solved using standard LMI software tools. A solution to the relaxed problem (21) is also a solution to the constrained problem (20). It should be noted that, in general, several subsystems A_i can be used in each Ω-region.

In addition, the LMI formulation requires that whenever there is movement to an adjacent region Ω_j with corresponding Lyapunov function V_j, then $V_j(x) \leq V_i(x)$. Using local quadratic Lyapunov-like functions this condition can be written $x^T P_j x \leq x^T P_i x$. The states where this condition must be satisfied also have to be expressed by quadratic forms. The S-procedure is used to replace the constrained condition with an unconstrained LMI problem that can be solved very efficiently.

V. HYBRID AUTOMATA

Hybrid automata were introduced in the study of hybrid systems in the early 1990s. Hybrid automata provide a general modeling formalism for the formal specification and algorithmic analysis of hybrid systems. They are typically used to model dynamical systems that consist of both discrete and analog components which arise when computer programs interact with the physical world in real time. In the following, we review the hybrid automaton model and related approaches for analysis, verification, and synthesis of hybrid systems.

A hybrid automaton is a finite state machine equipped with a set of real-valued variables. The state of the automaton changes either instantaneously through a discrete transition or through a continuous activity. The hybrid automaton in Fig. 14 describes a thermostat and is used to introduce the modeling framework.

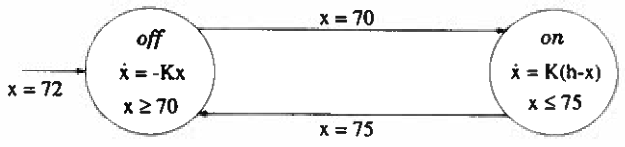

FIGURE 14 Hybrid automaton describing a thermostat.

Example. The hybrid automaton in Fig. 14 models a thermostat controlling the temperature of a room by turning a heater on and off. The real-valued variable x denotes the temperature. The system has two control modes, *off* and *on*. When the heater is off the temperature of the room falls according to the differential equation $\dot{x} = -Kx$. When the heater is on (control mode *on*) the temperature of the system rises according to the equation $\dot{x} = K(h - x)$, where h is a constant. Initially, the temperature is $x = 72$ and the heater is off. The heater will go on as soon as the falling temperature reaches 70°F; the discrete part of the state will then be in the position *on* (Fig. 14) and the continuous part of the state will start at $x = 70$. When the heater is on the temperature rises until it reaches 75°F. Then the heater will go off and the temperature will start falling again. This control policy guarantees that the temperature of the room will remain at between 70 and 75°F. □

A hybrid automaton consists of a finite set $X = \{x_1, \ldots, x_n\}$ of real-valued variables and a labeled directed graph (V, E). V is a finite set of vertices and E is a set of directed arcs or edges between vertices. The directed graph models the discrete (event) portion of the hybrid system. Directed graphs have a very convenient graphical representation. A circle is used to represent each vertex of the graph. An arrow starting at vertex v_i and terminating at vertex v_j represents the directed arc (v_i, v_j). The graph shown in Fig. 14 consists of two vertices and two edges. Note that the arc labeled $x = 72$ is used for the initialization of the system.

The dynamics of the hybrid automaton are defined by labeling the vertices V and edges E of the graph with appropriate mathematical expressions involving the real-values variables $X = \{x_1, \ldots, x_n\}$. The vertices represent continuous activities and they are labeled with constraints on the derivatives of the variables in X. More specifically, a vertex $v \in V$, which is also called a *(control) mode* or *location*, is equipped with the following labeling functions.

- A *flow condition* described by a differential equation in the variables in X. While the hybrid automaton is in control mode v, the variables x_i change according to the flow condition. For example, in the thermostat automaton, the flow condition $\dot{x} = K(h - x)$ of the control mode *on* ensures that the temperature is rising while the heater is on.
- An *invariant condition* $\text{inv}(v) \in \Re^n$ that assigns to each control mode a region of $\Re^n$. The hybrid automaton may reside in control mode v only while the invariant condition $\text{inv}(v)$ is true. For example, in the thermostat automaton, the invariant condition $x \leq 75$ of the control mode *on* ensures that the heater must go off when the temperature rises to 75°F.

An edge $e \in E$ is also called a *control switch* or *transition* and is labeled with an assignment of the variables in X called a guard. A transition is enabled when the associated guard is true and its execution modifies the values of the variables according to the assignment. For example, the thermostat automaton has two control switches. The control switch from control mode *on* to *off* is described by the condition $x = 75$.

A *state* $\sigma = (v, x)$ of the hybrid automaton consists of a mode (control location) $v \in V$ and a particular value $x \in \Re^n$ of the variables in X. The state can change either by a discrete and instantaneous transition or by a time delay through the continuous flow. A discrete transition changes both the control location and the real-valued variables, while a time delay changes only the values of the variables in X according to the flow condition. A *run* of a hybrid automaton H is a finite or infinite sequence,

$$\rho: \sigma_0 \rightarrow^{t_0}_{f_0} \sigma_1 \rightarrow^{t_1}_{f_1} \sigma_2 \rightarrow^{t_2}_{f_2} \ldots,$$

where $\sigma_i = (v_i, x_i)$ are the state of H and f_i is the flow condition for the vertex v_i such that (i) $f_i(0) = x_i$, (ii) $f_i(t) \in \text{inv}(v_i)$ for all $t \in \Re: 0 \leq t \leq t_i$, and (iii) σ_{i+1} is a transition successor of $\sigma'_i = (v_i, f_i(t_i))$ and σ'_i is a time successor of σ_i.

A hybrid automaton is said to be *nonzeno* when only finitely many transitions can be executed in every bounded time interval. Nonzenoness is an important notion for the realizability of the hybrid automaton.

Another labeling function assigns to each transition an event from a finite set of events Σ. The event labels are used to define the parallel composition of hybrid automata. Complex systems can be modeled by using the parallel composition of simple hybrid automata. The basic rule for the parallel composition is that two interacting hybrid automata synchronize the execution of transitions labeled with common events.

Example. A train–gate–controller system is used to illustrate modeling of hybrid systems using hybrid automata. The system consists of three components, the train, the gate, and the gate controller as shown in Fig. 15.

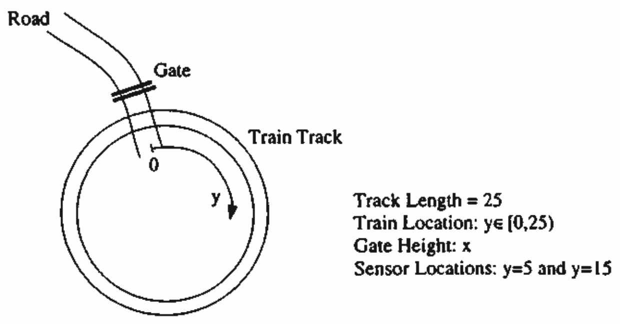

FIGURE 15 Train–gate–controller system.

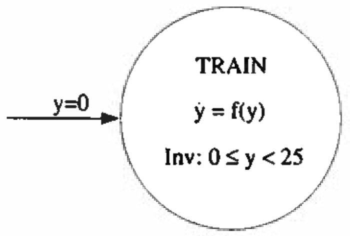

FIGURE 16 Hybrid automaton modeling the train.

A road crosses the train track and it is guarded by a gate which must be lowered to stop the traffic when the train approaches and raised after the train has passed the road. The gate controller gets information from sensors located on the track and lowers or raises the gate.

The train moves clockwise on a circular track. The length of the track is $L = 25$. The location of the train is indicated by the state variable y, where $0 \leq y < 25$. The velocity of the train is described by the differential equation $\dot{y} = f(y)$, where $f(y)$ is an appropriate function of y. The gate is located at $y = 0$ on the train track, while the sensors are at $y = 5$ and $y = 15$. The train is modeled by a hybrid automaton with one control mode as shown in Fig. 16.

The height of the gate is represented by the state variable x. When the gate is lowered the height of the gate decreases according to the equation $\dot{x} = (1 - x)/2$. When the gate is raised the height increases according to $\dot{x} = (10 - x)/2$. The hybrid automaton in Fig. 17 is used to model the dynamic behavior of the gate. The automaton has two control modes, *RAISE* and *LOWER*. The transitions of the automaton are labeled with the events *UP* and *DOWN*, which are generated by the controller. The controller is also modeled as a hybrid automaton as shown in Fig. 18. The controller receives information from the sensors and detects when the train reaches or moves away from the crossing. The controller automaton has two control locations, *DOWN* and *UP*, which trigger the transitions of the gate automaton. The hybrid automaton of the overall system is obtained by parallel composition and is shown in Fig. 19. □

The modeling formalism of hybrid automata is particularly useful in the case when the flow conditions, the invariants, and the transition relations are described by linear expressions in the variables in X. A hybrid automaton is *linear* if its flow conditions, invariants, and transition relations can be defined by linear expressions over the

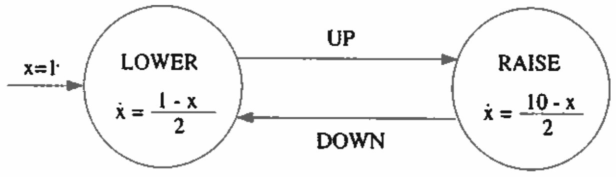

FIGURE 17 Hybrid automaton modeling the gate.

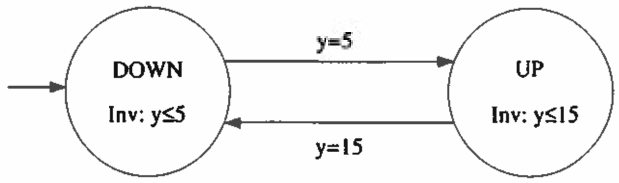

FIGURE 18 Hybrid automaton modeling the controller.

set X of variables. Note the special interpretation of the term linear in this context. More specifically, for the control modes the flow condition is defined by a differential equation of the form $\dot{x} = k$, where k is a constant, one for each variable in X, and the invariant inv(v) is defined by a linear equalities and inequalities (which corresponds to a convex polyhedron) in X. Also, for each transition the set of guarded assignments consists of linear formulas in X, one for each variable. Note that the run of a linear hybrid automaton can be described by a piecewise linear function whose values at the points of first-order discontinuity are finite sequences of discrete changes. An interesting special case of a linear hybrid automaton is a *timed automaton*. In a timed automaton each continuous variable increases uniformly with time (with slope 1) and can be considered a *clock*. A discrete transition either resets the clock or leaves it unchanged.

Another interesting case of a linear hybrid automaton is a rectangular automaton. A hybrid automaton is rectangular if the flow conditions are independent of the control modes and the variables are pairwise independent. In a rectangular automaton, the flow condition has the form $\dot{x} = [a, b]$ for each variable $x \in X$. The invariant condition and the transition relation are described by linear predicates that also correspond to n-dimensional rectangles. Rectangular automata are interesting because they characterize an exact boundary between the decidability and the undecidability of verification problems of hybrid automata. A problem is decidable if there exists an algorithm that has as output the correct answer for every possible input. A problem is undecidable if there is no algorithm that takes as input an instance of the problem and determines whether the answer to that instance is "yes" or "no." *Semidecidable* procedures are often proposed to deal with undecidable problems. These algorithms produce the correct answer if they terminate, but their termination is not guaranteed.

The main decision problem concerning the analysis and verification of hybrid systems is the *reachability problem*, which is formulated as follows. Let σ and σ' be two states in the infinite state space S of a hybrid automaton H. Then, σ' is reachable from σ if there exists a run of H that starts in σ and ends in σ'.

While the reachability problem is undecidable even for very restricted classes of hybrid automata, two semidecision procedures, forward and backward analysis, have

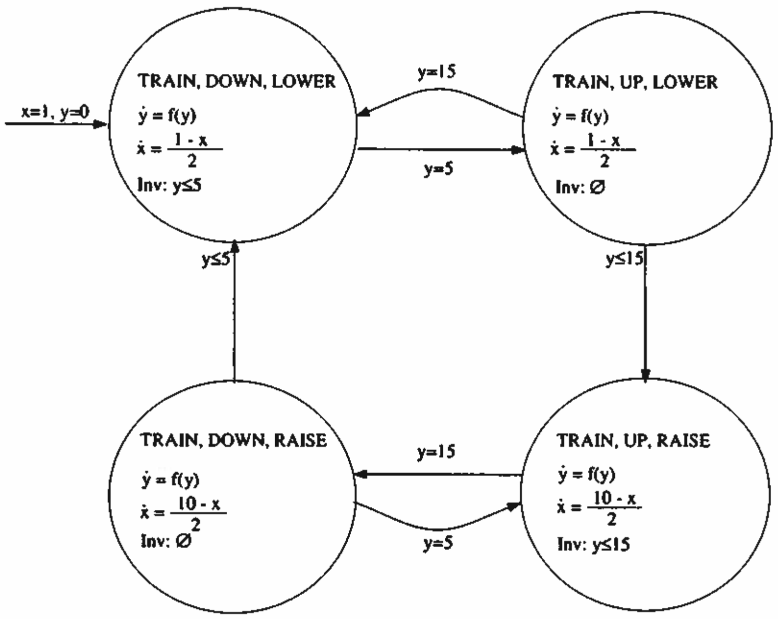

FIGURE 19 Hybrid automaton for the train–gate–controller system.

been proposed for the verification of safety specifications of linear hybrid automata. A *data region* R_v is a finite union of convex polyhedra in $\Re^n$. A *region* $R = (v, R_v)$ consists of a location $v \in V$ and a data region R_v and is a set of states of the linear hybrid automaton. Given a region R, the precondition of R, denoted pre(R), is the set of all states σ such that R can be reached from σ. The postcondition of R, denoted post(R), is the set of all the reachable states from R. For linear hybrid automata both pre(R) and post(R) are regions, i.e., the corresponding data region is a finite union of convex polyhedra. Given a linear hybrid automaton H, an initial region R, and a target region T, the reachability problem is concerned with the existence of a run of H that drives a state from R to a state in T. Two approaches for solving the reachability problem have been proposed. The first one computes the region post*(R) of all states that can be reached from the initial state R and checks if post*$(R) \cap T = \emptyset$ (forward reachability analysis). The second approach computes the region pre*(T) of the states from which one may reach T and checks if pre*$(T) \cap R = \emptyset$ (backward reachability analysis). Since the reachability problem for linear hybrid automata is undecidable, these procedures may not terminate (semidecision procedures). They terminate with a positive answer if T is reachable from R, while they terminate with a negative answer if no new states can be added and T is not reachable from R. The crucial step in these approaches is the computation of the precondition or postcondition of a region.

The reachability problem is central to the verification of hybrid systems. The train–gate–controller example is used to illustrate the verification approach using hybrid automata.

Example. For the train–gate–controller example, the specification is that the gate must be lowered ($x < 5$) whenever the train reaches the crossing. This is a safety specification that can be encoded as $y = 0 \Rightarrow x < 5$. This safety specification corresponds to a set S of safe states of the hybrid automaton shown in Fig. 19 which consists of all four control locations and the region of $\Re^2$ expressed by the set $\{(x, y): x < 5 \wedge y = 0\}$. To verify that the system satisfies the safety specification we compute the set of all states R that can be reached from the initial conditions. If the reachable set R is contained in the set of safe states $R \subset S$, then the gate is always down when the train reaches the crossing. □

The undecidability of the reachability problem is a fundamental obstacle in the analysis and controller synthesis for linear hybrid automata. Nevertheless, considerable research effort has been focused on developing systematic procedures for synthesizing controllers for large classes of problems.

Control design algorithms have been developed for a class of hybrid systems with continuous dynamics described by pure integrators. Although this class of hybrid systems is rather limited, these models are important for several applications including the control of batch processes. Note that even in the case where the continuous dynamics of the physical system are more complicated, it is sometimes useful to use low-level continuous controllers to impose linear ramp-like behavior around a setpoint. More specifically, in this case the continuous dynamics are described by differential equations of the form $\dot{x}(t) = k_v$, where k_v is a constant vector associated with the control mode v of the hybrid automaton. The control specifications are represented by data regions $R_v = \{x \in \Re^n : A_v x + b_v \leq 0\}$ and by a set Q_f of *forbidden control modes* or *forbidden control switches*. Controllability of hybrid integrator systems is defined with respect to a pair of regions of the hybrid state space. A hybrid system is controllable with respect to (R_1, R_2) if there exists an acceptable trajectory that drives the state (v, x) from R_1 to R_2. An acceptable trajectory is a trajectory of the hybrid system that satisfies the control specifications. For example, no forbidden control mode $v \in Q_f$ is visited, and for every legal control mode v the continuous state x lies in R_v. Based on the definition of controllability, a semidecidable algorithm has been developed that uses backward reachability analysis. This algorithm analyzes these integrator hybrid systems with respect to controllability and, as a by-product, generates a set of correct control laws that switch the system among a predefined number of control modes. The semidecidability of the algorithm is due to the undecidability of the reachability problem of linear hybrid automata.

Motivated by problems in aircraft conflict resolution, methodologies for synthesizing controllers for nonlinear hybrid automata based on a game theoretical framework have also been developed. In this approach, the continuous dynamics are described by nonlinear differential equations (that satisfy appropriate conditions for the existence and uniqueness of solutions). The regions of the hybrid state space consist of arbitrary invariant conditions for the control modes and regions of the form $G = \{x \in \Re^n : l(x) < 0\}$, where $l : \Re^n \rightarrow \Re$ is a differentiable function. The control specifications are expressed as acceptance conditions on the system's state. The controller synthesis problem is formulated as a dynamic game between the controller and the environment. The goal is to construct the largest set of states for which the control can guarantee that the acceptance condition is met despite the action of the disturbance. The problem is solved by iterating two appropriate predecessor operators. Consider a region K of the hybrid state space. The *controllable predecessor* of K contains all states in K for which the controllable actions can force the state to remain in K for at least one discrete step. The *uncontrollable predecessor* contains all states in K^c (the complement of K) and all states in K from which the uncontrollable actions may be able to force the state to move outside K. The computation of the predecessor operators is carried out using an appropriate Hamilton–Jacobi–Bellman equation. The computational efficiency of the synthesis procedure depends on the ability to solve this Hamilton–Jacobi–Bellman equation efficiently.

Another approach uses bisimulations to study the decidability of verification algorithms. Bisimulations are quotient systems that preserve the reachability properties of the original hybrid system, and therefore, problems related to the reachability of the original system can be solved by studying the quotient system. Quotient systems are simplified systems derived from the original system by aggregating the states in an appropriate manner. The idea of using finite bisimulations for the analysis and synthesis of hybrid systems is similar to the approximation of the continuous dynamics with DESs.

In summary, recent research efforts toward controller synthesis results have shown that there are classes of hybrid systems for which computationally tractable procedures can be applied. Although many important problems related to hybrid automata are intrinsically difficult, there are efficient algorithms for large classes of systems. Many practical applications can be modeled accurately enough by simple hybrid models. Again, the choice of such models depends on their suitability for studying specific problems.

SEE ALSO THE FOLLOWING ARTICLES

CELLULAR AUTOMATA • CONTROLS, ADAPTIVE SYSTEMS • CONTROLS, BILINEAR SYSTEMS • CONTROLS, LARGER-SCALE SYSTEMS • CONTROL SYSTEMS, IDENTIFICATION • CYBERNETICS AND SECOND ORDER CYBERNETICS • GAME THEORY • ROBOTICS, HYBRID SYSTEMS FOR

BIBLIOGRAPHY

Alur, R., Henzinger, T., and Sontag, E. (eds.) (1996). "Hybrid Systems III—Verification and Control," Lecture Notes in Computer Science, Vol. 1066, Springer-Verlag, New York.

Antsaklis, P. (guest ed.) (2000). Special Issue on Hybrid Control Systems. *Proc. IEEE* **88**, No. 7.

Antsaklis, P. J., and Lemmon, M. D. (guest eds.) (1998). Special Issue on Hybrid Control Systems. *J. Discrete Event Dynam. Syst. Theory Appl.* **8**, No. 2.

Antsaklis, P. J., and Nerode, A. (guest eds.) (1998). Special Issue on Hybrid Control Systems. *IEEE Trans. Auto. Control* **43**, No. 4.

Antsaklis, P., Kohn, W., Nerode, A., and Sastry, S. (eds.) (1995). "Hybrid Systems II," Lecture Notes in Computer Science, Vol. 999, Springer-Verlag, New York.

Antsaklis, P., Kohn, W., Nerode, A., and Sastry, S. (eds.) (1997). "Hybrid Systems IV," Lecture Notes in Computer Science, Vol. 1273, Springer-Verlag, New York.

Antsaklis, P., Kohn, W., Lemmon, M., Nerode, A., and Sastry, S. (eds.) (1999). "Hybrid Systems V," Lecture Notes in Computer Science, Vol. 1567, Springer-Verlag, New York.

Evans R., and Savkin, A. V. (guest eds.) (1999). Special Issue on Hybrid Control Systems. *Syst. Control Lett.* **38,** No. 3.

Grossman, R. L., Nerode, A, Ravn, A. P., and Rischel, H. (eds.) (1993). "Hybrid Systems," Lecture Notes in Computer Science, Vol. 736, Springer-Verlag, New York.

Henzinger, T., and Sastry, S. (eds.) (1998). "Hybrid Systems: Computation and Control," Lecture Notes in Computer Science, Vol. 1386, Springer-Verlag, New York.

Krogh, B., and Lynch, N. (eds.) (2000). "Hybrid Systems: Computation and Control," Lecture Notes in Computer Science, Vol. 1790, Springer-Verlag, New York.

Maler, O. (ed.) (1997). "Hybrid and Real-Time Systems," Lecture Notes in Computer Science, Vol. 1201, Springer-Verlag, New York.

Morse, A. S. (ed.) (1996). "Control Using Logic-Based Switching," Lecture Notes in Control and Information Systems, Vol. 222, Springer-Verlag, New York.

Morse, A., Pantelides, C., Sastry, S., and Schumacher, J. (guest eds.) (1999). Special Issue on Hybrid Control Systems. *Automatica* **35,** No. 3.

Sifakis, J., and Pnueli, A. (guest eds.) (1995). Special Issue on Hybrid Control Systems. *J. Theor. Comput. Sci.* **138.**

Vaandrager, F., and van Schuppen, J. (eds.) (1999). "Hybrid Systems: Computation and Control," Lecture Notes in Computer Science, Vol. 1569, Springer-Verlag, New York.

256