Hybrid Systems: Review and Recent Progress

Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556
antsaklis.1@nd.edu

Xenofon D. Koutsoukos
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
koutsouk@parc.com

Editors' Summary

The last part of this volume focuses on hybrid dynamical systems, an area of research that has developed as a result of the increasing interaction, over the last several years, between the control engineering and computer science communities. This chapter provides a broad-based introduction to hybrid systems and discusses a number of topics of ongoing research.

Hybrid systems are systems that exhibit both continuous-time and discrete-event dynamics. In the former case, the dynamics can be defined by differential or difference equations. For the latter, common representations include finite state machines and Petri nets. A good example of hybrid dynamics is multi-mode behavior. For example, different continuous-time models may be useful for capturing the dynamics of an aircraft in takeoff, landing, cruise, and other operational modes. Analogously, the control laws are generally very different in these cases as well-both plant models and controllers can be hybrid systems. Given the compounded complexity of hybrid systems, new methods are needed for their design and analysis. Over the last few years, research in this area has matured to a point where a number of tools have been developed.

A formalism for hybrid systems that has proven productive is hybrid automata. A hybrid automaton is a finite state machine extended with real-valued variables and differential equations that are defined differently for each (discrete) state of the FSM. An important special case is a linear hybrid automaton, in which the rates of change of the continuous variables are constant. Hybrid automata (especially in their simplified versions) have proven useful in addressing problems of verification in hybrid systems.

The stability of hybrid systems is another research area that has attracted interest. Hybrid system stability analysis relies for the most part on Lyapunov theory, extending the concept of a continuous and differentiable Lyapunov energy function to multiple Lyapunov functions. The final topic discussed in the chapter is the approximation of a continuous-time system by a discrete-event system (DES). The controller can then also be a DES. This approximation scheme has a number of interesting aspects, such as the inherent nondeterminism of the DES plant model.

1 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 described, 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 [18]. For similar reasons, that is, tight interaction of continuous and discrete dynamics and demands for high performance of the system, hybrid models are important in chemical processes [21], robotic manufacturing systems [49, 46], transportation systems [24], air traffic control systems [32] 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 [16]. 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 [14].

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. 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 [38, 39]. 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.

2 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.

It is very important to have good software tools for the simulation, analysis, and design of hybrid systems, which by their nature are complex systems. This need has been recognized by the hybrid system community and several software tools have been developed. Here, we list some of the available software tools. It should be noted that the list of software tools dynamically changes to accommodate the progress in hybrid system research. Tools that have been traditionally used by engineers for simulation and design of continuous systems have been extended to provide various functionalities for hybrid systems. Modeling and simulation tools require the development of sophisticated algorithms that address various problems that arise by interfacing continuous and discrete dynamics. These issues have been studied in the literature; see for example [52, 40, 31].

The Matlab/Simulink/Stateflow software environment [9] provides tools for visual modeling and simulation of hybrid systems that may include continuous-time, discrete-time, and event-driven dynamics. Ptolemy II [5] is a set of software packages supporting heterogeneous, concurrent modeling and design. It supports several models of computation including finite state machines, discrete event systems, and continuous-time systems as well as the appropriate interfaces that enable hybrid system modeling and simulation by orchestrating the interaction of these domains. It should be noted that Ptolemy II provides additional capabilities not necessarily related to hybrid systems; for details see [28]. The object oriented language Modelica [4] has been developed for modeling physical systems and provides modeling paradigms for hybrid systems. Simulation engines and tools such as Dymola [1] and MathModelica [3] that support Modelica can be used for simulation of physical systems that exhibit hybrid phenomena. HCC [2] is a concurrent constrained-based object oriented language that supports hybrid dynamics. An HCC compiler was initially developed at Xerox PARC and then extended at NASA Ames, where it has been used in various space applications. The SHIFT programming language [11] was developed for describing dynamic net-

works of hybrid automata. Although, the primary motivation was the specification and analysis of automotive applications, SHIFT has been used in various application domains. OmSim is a software environment for modeling and simulation based on Omola, an object oriented language for representing continuous-time and discrete-event dynamical systems [10]. Charon is a language for hierarchical and modular modeling of hybrid systems [6]. HyTech is a verification tool for hybrid systems based on the theory of linear hybrid automata [7]. Kronos [8] and UPPAAL [12] are verification tools for real-time systems modeled by timed automata. Hybrid system software tools have been developed in the chemical processing industry; for more details see [21]. In addition, many research groups are developing software tools for supporting their work.

Several approaches to modeling, analysis and synthesis of hybrid systems are described in this article. They are organized into three sections. First, approaches based on hybrid automata models are discussed. Then approaches that emphasize stability are presented. Finally, the supervisory control approach to the analysis and design of hybrid control systems is described. It should be noted that considerable research efforts have addressed additional topics such as optimal control, controllability, observability, and diagnosis. We believe that the research threads that were selected represent the most important topics with mature techical results.

3 Hybrid Automata

Hybrid automata were introduced in the study of hybrid systems in the early 1990s [13]. 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 [13, 22, 35, 29].

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 Figure 1 describes a thermostat and is used to introduce the modeling framework.

Example. The hybrid automaton in Figure 1 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* (Figure 1) 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. □

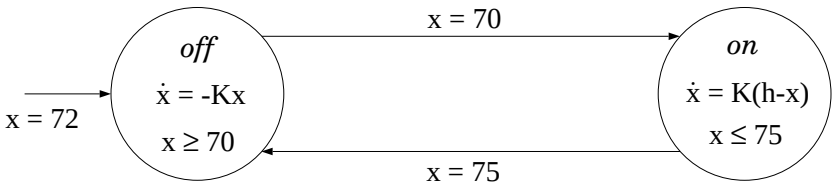

Figure 1: Hybrid automaton describing a thermostat.

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 Figure 1 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* $inv(v) \in \Re^n$ 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 $inv(v)$ 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

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

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) $f_i(t) \in inv(v_i)$ for all $t \in \Re : \ 0 \leq t \leq t_i$, and (iii) σ_{i+1} is a transition successor of $\sigma_i' = (v_i, f_i(t_i))$, where 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 Figure 2. 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.

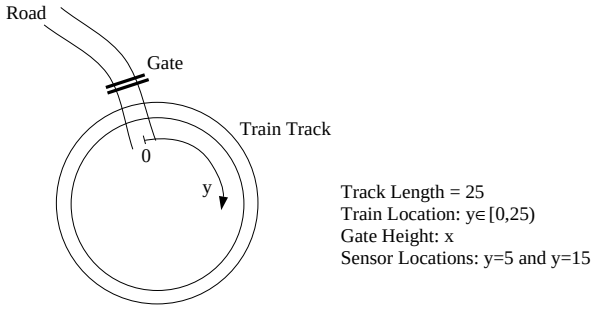

Figure 2: Train-Gate-Controller system.

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 Figure 3.

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 Figure 3 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 Figure 3. 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 Figure 3. □

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 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

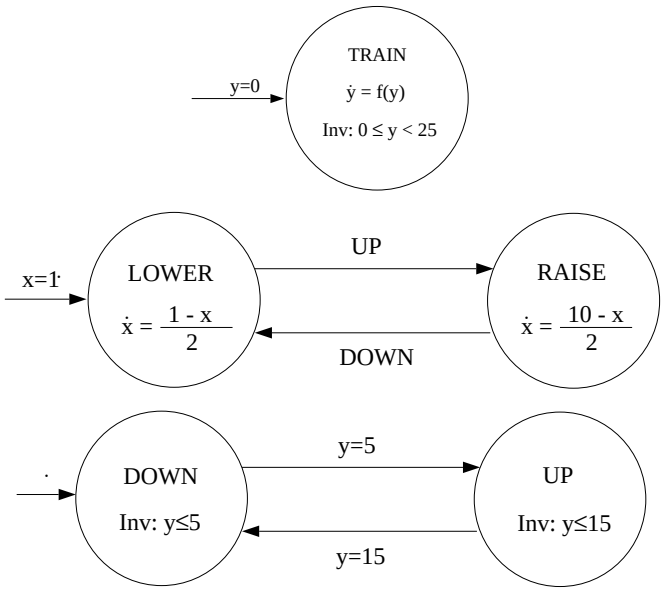

Figure 3: Hybrid automata modeling the train, gate, and controller.

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 the reachability and the controller synthesis problems are decidable under the assumption that the mode transitions occur using sampling at integer points in time [23]. 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

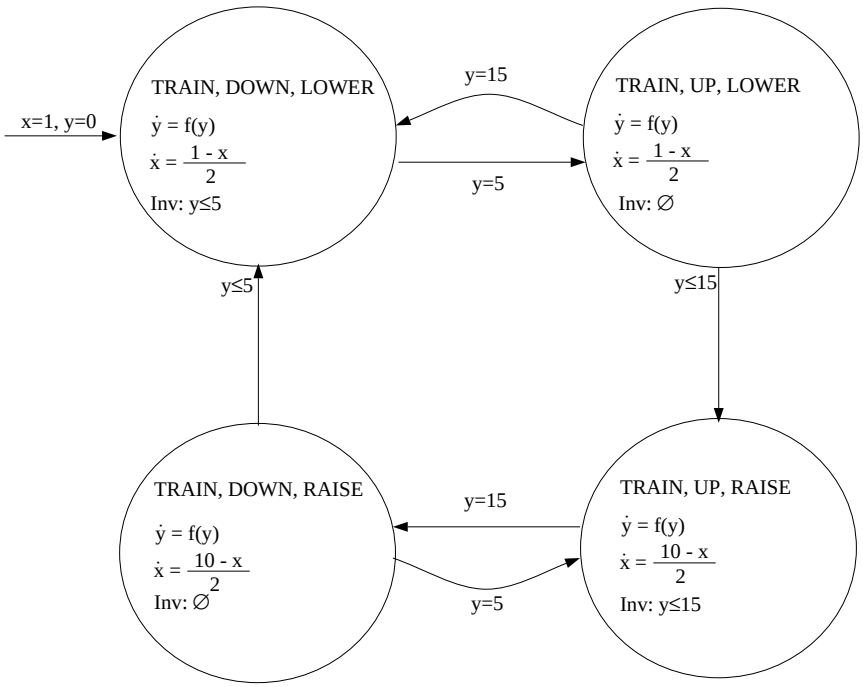

Figure 4: Hybrid automaton for the train-gate-controller system.

problem and determines whether the answer to that instance is "yes" or "no." *Semi-decidable* 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 semi-decision procedures, forward and backward analysis, have 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 $\Re^n$ that can be defined using a logical formula with linear predicates [13]. 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 (semi-decision 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 Figure 4 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 [14].

Control design algorithms have been developed for a class of hybrid systems with continuous dynamics described by pure integrators [53]. 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 set-point.

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 [54]. Another approach uses bisimulations to study the decidability of verification algorithms [14]. 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 (see discusion in Section 5).

4 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 [19, 56, 37], see also [30, 36, 20] and the references therein. 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 Figure 5. Typical examples of such systems are computer disk drives, constrained mechanical systems, switching power converters, and automotive power-train applications.

Mathematically, such hybrid systems can be modeled by the equations

$$\begin{aligned} \dot{x} &= f(x(t), q(t), u(t)), \\ q(t^+) &= \delta(x(t), q(t)), \end{aligned}$$

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 discrete-event dynamics.

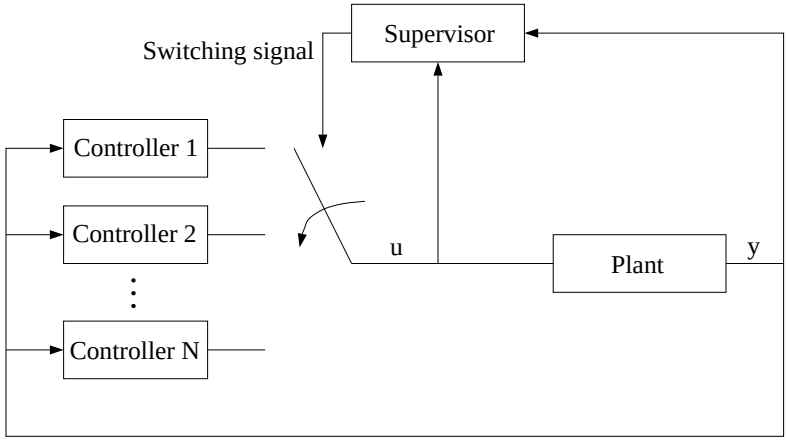

Figure 5: Switching controller feedback architecture.

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

$$\begin{aligned} \dot{v} &= -\frac{k}{m}v^2 sign(v) - g\sin\alpha + \frac{G_{q(t)}}{m}T, \\ \omega &= G_{q(t)}v, \end{aligned}$$

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

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

where ω_{high} and ω_{low} are prescribed angular velocities of the engine. □

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

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

where $q(t) \in \{1, 2, \ldots, N\}$. It is also 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. Further, it is assumed that the switchings occur instantaneously and they do not excite unmodeled high-frequency dynamics.

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 (1), 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,

$$V_i\left(x(t_{i,k}\right) \leq V_i\left(x(t_{i,k-1}\right) \tag{2}$$

where $t_{i,k}$ denotes the kth time the subsystem f_i is "switched in." Then system (1) is stable in the sense of Lyapunov. The stability condition (2) is illustrated in Figure 6. At every time 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.

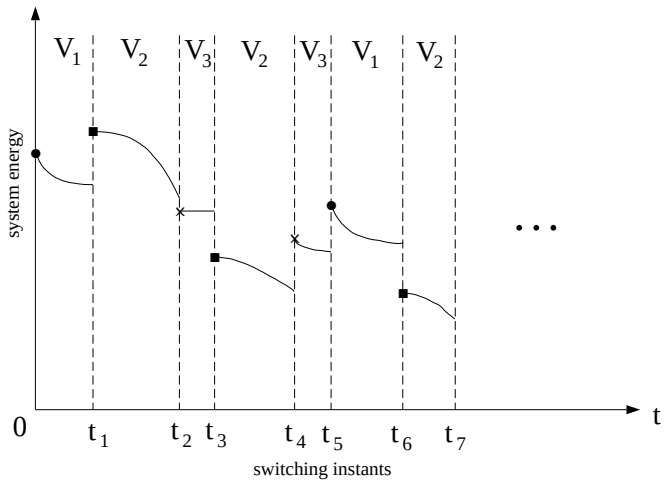

Figure 6: Stability condition.

The general result presented above 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.

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

$$\begin{aligned} \dot{x} &= A_{q(t)}x(t), \\ q(t^+) &= \delta(x(t), q(t)), \end{aligned}$$

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.

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 stability for any switching signals is to guarantee that the matrices A_i share a common quadratic Lyapunov function $V(x) = x^T Px$, 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 [25]. 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

$$\dot{V}_i(x) = \left[\frac{\partial}{\partial x} V_i(x)\right] A_i x = x^T (A_i^T P_i + P_i A_i) x \leq 0. \tag{3}$$

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

$$A_i^T P_i + P_i A + \xi Q_i \leq 0, \tag{4}$$

which can be solved using standard LMI software tools. A solution to the relaxed problem (4) is also a solution to the constrained problem (3). 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.

Non-quadratic Lyapunov functions have also been used for studying the stability of switched linear systems [26]. For example, Lyapunov functions defined by the infinity norm result in polyhedral partitions of the state space that can be used very efficiently for the stability analysis and verification of linear switched systems.

5 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 and methodologies for supervisory control design were developed [48, 43, 44, 45]. In related developments, the relation between inherently discrete planning systems and continuous feedback control systems attracted

attention [15]. 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 DESs as an attractive alternative to methodologies based on finite automata [41, 55].

In this section, we review the supervisory control framework for hybrid systems [42, 17, 33, 50, 47, 34, 27]. One of the main characteristics of the supervisory control 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 Figure 7. It is generally assumed that the dynamic behavior of the plant is governed by a set of known nonlinear ordinary differential equations,

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

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 Figure 7, 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.

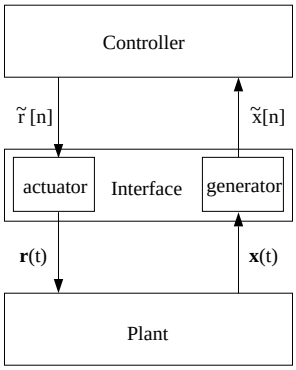

Figure 7: Hybrid system model in the supervisory control framework.

The interface consists of the generator and the actuator as shown in Figure 7. The generator has been chosen to be a partitioning of the state space (see Figure 8). The piecewise continuous

command signal issued by the actuator is a staircase signal 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.

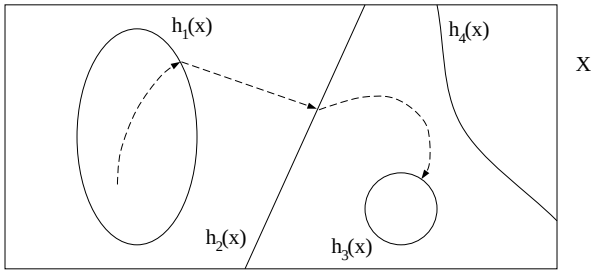

Figure 8: Partition of the continuous state space.

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 $f : T \to W$) the *behavior*. The behavior of

the DES plant model consists of all the pairs of plant 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

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

and is depicted in Figure 9. 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.

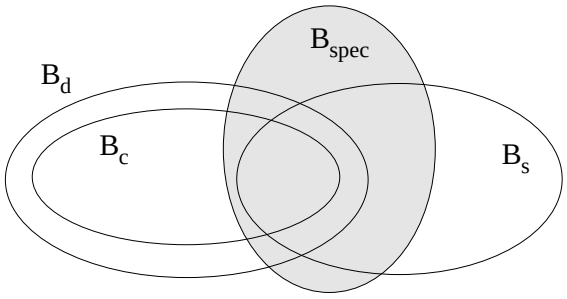

Figure 9: The DES plant model as an approximation.

An interesting aspect of the DES plant's behavior is that it is distinctly nondeterministic. This fact is illustrated in Figure 10 which shows two trajectories generated by the same control symbol. Both trajectories originate in the same DES plant state $\tilde{p}_1$. Figure 10 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 [51].

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

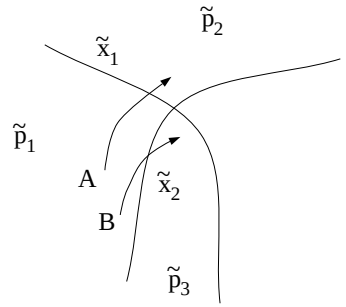

Figure 10: Nondeterminism of the DES plant model.

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 DES. 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 75°F. At 75°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

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

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.

The thermostat partitions the state space of the plant with the following hypersurfaces

$$h_1(x) = x - 75,$$
$$h_2(x) = 70 - x.$$

The first hypersurface detects when the state exceeds 75°F, and the second detects when the state falls below 70°F. The associated functions with the interface that generate the plant events, α_1 and α_2, are very simple in this case.

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

So there are two plant symbols, $\tilde{x}_1$ and $\tilde{x}_2$. The DES controller is shown in Figure 11. The output function of the controller is defined as

$$\phi(\tilde{s}_1) = \tilde{r}_1 \Leftrightarrow \text{off},$$
$$\phi(\tilde{s}_2) = \tilde{r}_2 \Leftrightarrow \text{on},$$

and the actuator operates as

$$\begin{aligned} \gamma(\tilde{r}_1) &= 0, \\ \gamma(\tilde{r}_2) &= 12, \end{aligned}$$

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

The thermostat/heater example has a simple DES plant model which is useful to illustrate how these models work. Figure 11 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. □

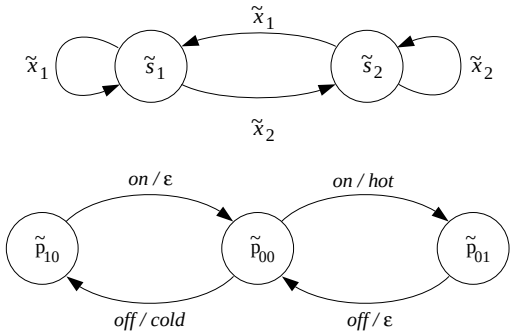

Figure 11: Controller and DES plant for the thermostat/furnace system.

6 Conclusions

This article presents a brief overview of three research threads in hybrid systems, namely hybrid automata based modeling and verification, stability analysis, and supervisory control. It should be noted that considerable progress has been achieved in related areas such analysis and synthesis of piecewise linear systems, optimal control of switched systems, and hybrid system diagnosis among others. The research efforts in the area of hybrid dynamical systems address many important challenges, such as real-time control reconfiguration, mode switching, safety, reachability, and liveness among others, and provide the theoretical background and the principles for the development of software-enabled control technologies. Although, many important research developments were omitted, it is hoped that this paper will provide a useful and representative description of main approaches to hybrid systems together with references, and as such it will be a useful resource to researchers. In summary, although many important problems related to hybrid systems are intrinsically difficult, there are efficient simulation, analysis, and synthesis algorithms for large classes of systems. Recent research efforts towards hybrid system design have shown that there are classes of hybrid systems for which computationally tractable procedures can be applied. Many practical applications can be modeled accurately enough by suitable hybrid models. Again, the choice of such models depends on their suitability for studying specific problems.

References

[1] Dymola, Dynasim AB. Homepage: http://www.dynasim.se/.

[2] Hybrid CC, hybrid automata, and program verification. Homepage: http://www.parc.xerox.com/spl/projects/mbc/languages.html.

[3] Mathmodelica, MathCore. Homepage: http://www.mathcore.com.

[4] Modelica. Homepage: http://www.modelica.org.

[5] Ptolemy II, Department of EECS, UC Berkeley. Homepage: http://ptolemy.eecs.berkeley.edu/ptolemyII/.

[6] Charon, Deparment of Computer and Information Science, University of Pennsylvania. Homepage: http://www.cis.upenn.edu/mobies/charon/.

[7] HyTech. Homepage: http://wwww-cad.eecs.berkeley.edu/ tah/HyTech/.

[8] Kronos, VERIMAG. Homepage: http://www-verimag.imag.fr/TEMPORISE/kronos/.

[9] Matlab, The Mathworks, Inc. Homepage: http://www.mathworks.com.

[10] OmSim, Department of Automatic Control, Lund Institue of Technology. Homepage: http://wwww.control.lth.se/ cace/omsim.html.

[11] Shift, California PATH, UC Berkeley. Homepage: http://wwww.path.berkeley.edu/shift/.

[12] UPPAAL. Homepage: http://www.docs.uu.se/docs/rtmv/uppaal/.

[13] 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. *Theoretical and Computer Science*, 138:3–34, 1995.

[14] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proceedings of IEEE*, 88(7):971–984, July 2000.

[15] P. Antsaklis and K. Passino, editors. *An Introduction to Intelligent and Autonomous Control*. Kluwer Academic Publishers, 1993.

[16] P. Antsaklis, K. Passino, and S. Wang. Towards intelligent autonomous control systems: Architecture and fundamental issues. *Journal of Intelligent and Robotic Systems*, 1:315–342, 1989.

[17] P. Antsaklis, J. Stiver, and M. Lemmon. Hybrid system modeling and autonomous control systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 366–392. Springer-Verlag, 1993.

[18] A. Balluchi, L. Benvenuti, M. D. Benedetto, C. Pinello, and A. Sangiovanni-Vincentelli. Automotive engine control and hybrid systems: Challenges and opportunities. *Proceedings of IEEE*, 88(7):888–912, July 2000.

[19] M. Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):475–482, 1998.

[20] R. DeCarlo, M. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of IEEE*, 88(7):1069–1082, July 2000.

[21] S. Engell, S. Kowalewski, C. Schulz, and O. Stursberg. Continuous-discrete interactions in chemical processing plants. *Proceedings of IEEE*, 88(7):1050–1068, July 2000.

[22] T. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HyTech. In *First Workshop on Tools and Algorithms for the Construction and Analysis of Systems, TACAS95*, volume 1019 of *Lecture Notes in Computer Science*, pages 41–71. Springer-Verlag, 1995.

[23] T. Henzinger and P. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221:369–392, 1999.

[24] R. Horowitz and P. Varaiya. Control design of an automated highway system. *Proceedings of IEEE*, 88(7):913–925, July 2000.

[25] M. Johansson and A. Rantzer. Computation of piecewise quadratic Lyapunov functions for hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):555–559, 1998.

[26] X. Koutsoukos and P. Antsaklis. Characterizing of switching stabilizing sequences in switched linear systems using piecewise linear Lyapunov functions. In M. D. Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems—Computation and Control 2001*, volume 2034 of *Lecture Notes in Computer Science*, pages 347–360. Springer-Verlag, 2001.

[27] X. Koutsoukos, P. Antsaklis, J. Stiver, and M. Lemmon. Supervisory control of hybrid systems. *Proceedings of IEEE*, 88(7):1026–1049, July 2000.

[28] E. Lee. Overview of the ptolemy project. Technical Memorandum UCB/ERL M01/11, University of California, Berkeley, March 6, 2001.

[29] M. Lemmon, K. He, and I. Markovsky. Supervisory hybrid systems. *Control Systems Magazine*, 19(4):42–55, August 1999.

[30] D. Liberzon and A. Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems Magazine*, 19(5):59–70, October 1999.

[31] J. Liu, X. Liu, T. Koo, J. Sinopoli, S. Sastry, and E. Lee. A hierarchical hybrid system model and its simulation. In *Proceedings of the 38th IEEE Conference on Decision and Control*, pages 3508–3513, Phoenix, AZ, December 1999.

[32] C. Livadas, J. Lygeros, and N. Lynch. High-level modeling and analysis of the traffic alert and collision avoidance system. *Proceedings of IEEE*, 88(7):926–948, July 2000.

[33] J. Lunze. Qualitative modelling of linear dynamical systems with quantised state measurements. *Automatica*, 30(3):417–431, 1994.

[34] J. Lunze, B. Nixdorf, and J. Schroder. Deterministic discrete-event representations of linear continuous-variable systems. *Automatica*, 35(3):396–406, 1999.

[35] N. Lynch, R. Segala, F. Vaandrager, and H.Weinberg. Hybrid I/O automata. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III, Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 496–510. Springer, 1996.

[36] A. Michel. Recent trends in the stability analysis of hybrid dynamical systems. *IEEE Transactions on Circuits and Systems I*, 46(1):120–134, 1999.

[37] A. Michel and B. Hu. Towards a stability theory of general hybrid dynamical systems. *Automatica*, 35(3):371–384, 1999.

[38] A. Morse. Logic-based switching and control. In B.A.Francis and A.R.Tannenbaum, editors, *Feedback Control, Nonlinear Systems, and Complexity*, pages 173–195. Springer-Verlag, 1995.

[39] A. S. Morse, editor. *Control using logic-based switching*, volume 222 of *Lecture Notes in Control and Information Sciences*. Springer, 1997.

[40] P. Mosterman. An overview of hybrid simulation phenomena and their support by simulation packages. In *HSCC 99: Hybrid Systems—Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 165–177. Springer, 1999.

[41] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of IEEE*, 77(4):541–580, 1989.

[42] A. Nerode and W. Kohn. Models for hybrid systems: Automata, topologies, controllability, observability. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 317–356. Springer-Verlag, 1993.

[43] C. Özveren and A. Willsky. Observability of discrete event dynamic systems. *IEEE Transactions on Automatic Control*, 35(7):797–806, 1990.

[44] C. Özveren, A. Willsky, and P. Antsaklis. Stability and stabilizability of discrete event dynamic systems. *Journal of the ACM*, 38(3):730–752, 1991.

[45] K. Passino, A. Michel, and P. Antsaklis. Lyapunov stability of a class of discrete event systems. *IEEE Transactions on Automatic Control*, 39(2):269–279, 1994.

[46] D. Pepyne and C. Cassandras. Optimal control of hybrid systems in manufacturing. *Proceedings of IEEE*, 88(7):1108–1123, July 2000.

[47] J. Raisch and S. O'Young. Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control*, 43(4):568–573, 1998.

[48] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–89, January 1989.

[49] M. Song, T.-J. Tarn, and N. Xi. Integration of task scheduling, action planning and control in robotic manufacturing systems. *Proceedings of IEEE*, 88(7):1097–1107, July 2000.

[50] J. Stiver, P. Antsaklis, and M. Lemmon. A logical DES approach to the design of hybrid control systems. *Mathl. Comput. Modelling*, 23(11/12):55–76, 1996.

[51] J. Stiver, X. Koutsoukos, and P. Antsaklis. An invariant based approach to the design of hybrid control systems. *International Journal of Robust and Nonlinear Control*, 11(5):453–478, 2001.

[52] J. Taylor and D. Kebede. Modeling and simulation of hybrid systems in MATLAB. In *IFAC 13th Triennial World Congress*, volume J, pages 275–280, San Francisco, CA, 1996.

[53] M. Tittus and B. Egardt. Control design for integrator hybrid system. *IEEE Transactions on Automatic Control*, 43(4):491–500, 1998.

[54] C. Tomlin, J. Lygeros, and S. Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of IEEE*, 88(7):949–970, July 2000.

[55] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis. Feedback control of Petri nets based on place invariants. *Automatica*, 32(1):15–28, 1996.

[56] H. Ye, A. Michel, and L. Hou. Stability theory for hybrid dynamical systems. *IEEE Transactions on Automatic Control*, 43(4):461–474, 1998.