# Hybrid Systems Control

Panos J. Antsaklis and Xenofon D. Koutsoukos
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

**Abstract**

The hybrid systems of interest contain two distinct types of components, 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, auto-pilot 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. In this article, a brief introduction to the theory and applications of hybrid systems is presented.

**Keywords** Hybrid Systems, Hybrid Control, Continuous and Discrete Event Dynamical Systems, Embedded Systems

## 1 INTRODUCTION

### 1.1 Hybrid Systems

Hybrid means, in general, heterogeneous in nature or composition and the term hybrid systems means systems with behavior defined by entities or processes of distinct characteristics. Here the term hybrid refers to combinations or compositions of continuous and discrete parts and a hybrid dynamical system is understood to mean a dynamical system where the behavior of interest is determined by interacting continuous and discrete dynamics. Hybrid dynamical systems generate variables or signals, that are mixed signals consisting of combinations of continuous or discrete value or time signals, and through them interaction with other systems and the environment occurs. More specifically, 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. Another distinction that could be made is that some of the signals could be time-driven while others could be event-driven in an asynchronous manner. The investigation of hybrid systems is creating a new and fascinating discipline bridging control engineering, mathematics and computer science. There has been significant research activity in the area of hybrid systems in the past decade involving researchers from several areas.

## 1.2 Applications and Background

When the continuous and discrete dynamics coexist and interact with each other it is important to develop models that accurately describe the dynamic behavior of such hybrid systems. Only in this way it may be possible to develop designs that fully take into consideration the relations and interaction of the continuous and discrete parts of the system. Many times it is not only desirable but also natural to use hybrid models to describe the dynamic behavior of systems. 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 fully understand the system's behavior and meet high performance specifications one needs to model all dynamics together with their interactions, and this is most important when there are strong interactions among the parts of the system. Only then problems such as optimization of the whole manufacturing process may be addressed in a more 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 discrete parts and high performance specifications are to be met by the system.

Hybrid models may also be used to significant advantage for example in automotive engine control, where there is need of 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 to meet challenging design requirements in the design of control systems for problems such as cut-off control and idle speed control of the engine. For similar reasons, that is tight interaction of continuous and discrete dynamics and demands for high performance for the system, hybrid models are important in chemical processes, robotic manufacturing systems, transportation systems, air traffic control systems among many other applications.

There are other ways 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, 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. There, 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; see for example Figure 1. If one also considers quantization of the continuous-valued variables or signals, then the hybrid systems contains not only continuous-valued variables that are driven by continuous and discrete times, but also discrete-valued signals as well. 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, that 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.

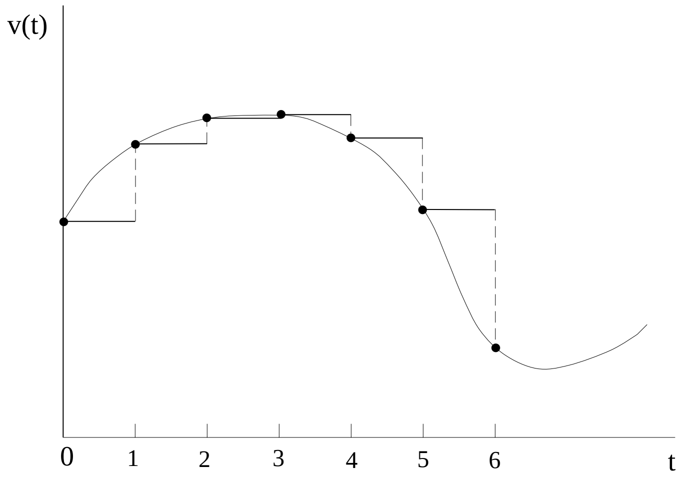

Figure 1: A signal, its samples, and its approximation by a zero order hold.

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, 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 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 sees a very simple, but adequate for the task in hand, model of the complex heat flow dynamics. For another example, in order 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 systems 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 has been motivated in part by the development of research results in the control of discrete event systems (DES) that occurred in the 80's and of adaptive control in the 70's and 80's and of 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 emphasis on verification of 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 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 hybrid systems 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 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. There are different types of models used, from detailed models that may include equations and look up 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. Below a brief introduction to modeling signals and systems is presented.

## 2 MODELING SIGNALS AND SYSTEMS

### 2.1 Modeling Signals

Continuous and discrete-time signals, where a signal $f(t)$ takes on a value from the set of real numbers for each value of the independent variable or time $t$, are certainly familiar to all. In continuous variables or analog signals, both the values of $f(t)$ and the time $t$ are real numbers; that is $f(t)$ is defined for any real $t$ in some interval and it may take on any real value. Examples include voltages and currents in a $RLC$ circuit. Discrete signals are defined only for discrete values of time $t$ and not for any real value $t$. For example a voltage may be measured every tenth of a second, but not in between. This is the case for example when the value of a signal, that could be representing some temperature or pressure in an engine, is known only from periodic measurements or samples. Such discrete signals are typical in sampling continuous signals in a periodic or non periodic manner. Every electrical engineer has studied continuous and discrete-time signals in the time domain or in transform domain using Fourier, Laplace and $z$ transforms. The relation between a continuous-time signal $f(t)$ and its sampled version $\{f(t_k)\}$ has been of great interest in several fields such as signal processing and numerical analysis. For example the celebrated Sampling Theorem prescribes the sampling rate so to be able to reconstruct the original (frequency band limited) signal $f(t)$. Now the value of a discrete-time signal may be obtained or stored using a digital device and because of the finite word length it is only approximated with accuracy depending primarily on the finite word length of the device. That is, a discrete-time signal becomes a digital one by quantization and in this case $f(t)$ takes on values from a discrete set. Such quantized, discrete-valued signals typically are not studied together with the continuous-valued ones primarily because of the mathematical difficulties. Instead, some probabilistic analysis of the quantization effects is frequently performed separately to validate the design. It should be noted also that today's digital devices tend to use longer word lengths and so the use of continuous-valued signals instead of discrete-valued in the analysis is adequate for many practical purposes.

Is the world after all analog, is it digital or is it both? This is certainly a rather challenging question to answer! But some thoughts are perhaps of some use to the reader. Certainly there are many examples of discrete decision making, cases where phenomena are inherently discrete and cases where physical quantities are sampled and represented via discrete values. Analog signals contain a continuum of real values, such a voltage $v(t)$ . Does an analog signal really exist or is it just a convenient way to represent signals? Recall that real numbers are such that between any two there is always a third real number. Does it make sense then to talk about values of voltages or distances represented perhaps by an infinite number of decimals in view of the fact that our measurements provide us only with a finite number of decimal digits? One could say of course that real numbers do not really exist in nature, but they represent an idealization that has helped us understand phenomena ranging from the motion of planets to the behavior of atoms. This may very well be true, however real numbers have retained their usefulness on scales smaller than one hundredth of the classical diameter of subatomic particles (electron, proton) and are possibly valid down to the quantum gravity scale, twenty orders of magnitude smaller than such a particle. It appears then that real numbers and continuous variables and signals are here to stay.

## 2.2 Modeling Dynamical Systems

The dynamical behavior of systems can be understood by studying their mathematical descriptions. The flight path of an airplane subject to certain engine thrust, rudder and elevator angles and under particular wind conditions, or the behavior of an automobile on cruise control when climbing a hill of certain elevation, the evolution in time of a production system in manufacturing can be predicted using mathematical descriptions of the behavior of interest. Mathematical relations, that typically involve differential or difference equations or finite automata and Petri nets, are used to describe the behavior of processes and predict their response when certain inputs are applied. Although computer simulation is an excellent tool for verifying predicted behavior and thus for enhancing our understanding of processes, it is certainly not an adequate substitute in analysis or design for generating the information captured in a mathematical model, when of course such a model is available.

Over the centuries, a great deal of progress has been made in developing mathematical descriptions of physical phenomena. In doing so, laws or principles of physics, chemistry, biology, economics, etc., are invoked to derive mathematical expressions (usually equations) which characterize the evolution in time of the variables that are of interest. The availability of such mathematical descriptions enables us to make use of the vast resources offered by the many areas of applied and pure mathematics to conduct qualitative and quantitative studies of the behavior of processes. A given model of a physical process may give rise to several different mathematical descriptions. For example, when applying Kirchhoff's voltage and current laws to a low frequency transistor model, one can derive a set of differential and algebraic equations, or a set consisting only of differential equations, or, a set of integro-differential equations, and so forth. The process of mathematical modeling, from a physical phenomenon to a model to a mathematical description, is essential in science and engineering. To capture phenomena of interest accurately and in tractable mathematical form is a demanding task, as can be imagined, and requires a thorough understanding of the process involved. In most nontrivial cases, this type of modeling process is close to an art form, since a good mathematical description must be detailed enough to accurately describe the phenomena of interest and at the same time simple enough to be amenable to analysis. Depending on the applications on hand, a given mathematical description of a process may be further simplified before it is used in analysis and especially in design procedures. A point which cannot be overemphasized is that mathematical descriptions characterize processes only approximately. Most often, this is the case because the complexity of physical systems defies exact mathematical formulation. In many other cases, however, it is our own choice that a mathematical description of a given process approximates the actual phenomena only by a certain desired degree of accuracy for simplicity. For example, in the description of $RLC$ circuits, one could use nonlinear differential equations which take into consideration parasitic effects in the capacitors. Most often however it suffices to use linear ordinary differential equations with constant coefficients to describe the voltage-current relations of such circuits, since typically such a description provides an adequate approximation and since it is much easier to work with linear rather than nonlinear differential equations.

There are of course many examples of systems that cannot be conveniently described by continuous models and differential equations. Such systems include production lines in manufacturing, computer networks, traffic systems etc. where their evolution in time depends on complex interactions of the timing of various discrete events. Such discrete event dynamical systems are modeled by discrete models, such as finite automata. Since many of these systems are man-made, the models tend to be easier to construct and be more accurate (although they tend to grow very large in the number of states) than in the case of modeling physical systems, however the same modeling considerations as the ones discussed above still apply.

The behavior of a hybrid dynamic system may be described via different models the detail and nature of which depends on what the intended use of the model is. There are hybrid models in the literature, that are more appropriate for simulation than for analysis or design. For some early mathematical models for hybrid systems and a comparison between models. In this special issue we are primarily interested in models that have been shown to be useful in the analysis of properties and the synthesis of controllers for hybrid systems.

## 3 APPROACHES TO THE ANALYSIS AND DESIGN OF HYBRID SYSTEMS

Systems that integrate continuous and discrete dynamics were of interest in the control systems and computer science literature in the past. For instance, in system theory in the 60s researchers were discussing mathematical frameworks so to study systems with continuous and discrete dynamics. Current approaches to hybrid systems differ with respect to the emphasis on or the complexity of the continuous and discrete dynamics, and on 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 discrete event systems that are described by finite automata and Petri nets to derive, with varying success, analysis and synthesis results. Several approaches to modeling, analysis and synthesis of hybrid systems are represented in this special issue.

## 3.1 Supervisory Control of Hybrid Systems

In the 80s 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 and observability and stability were defined and studied for discrete event systems 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 of this approach is that the plant is approximated by a discrete-event system 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 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 $x\ in\mathbb{R}^n$ is the continuous state of the system and $r \in \mathbb{R}^m$ is the continuous control input. In the model shown in Figure 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 discrete event system (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 Figure 2. The generator has been chosen to be a partitioning of the state space (see Figure 3). The piecewise continuous command signal issued by the actuator is a staircase signal as shown in Figure 4, not unlike the output of a zero-order hold in a digital control system. The interface plays a key role in determining

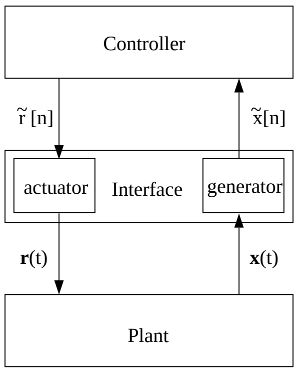

Figure 2: Hybrid system model in the supervisory control framework.

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.

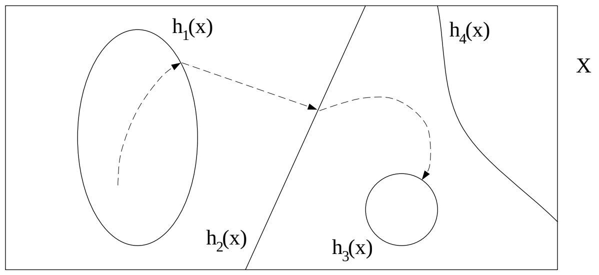

Figure 3: Partition of the continuous state space.

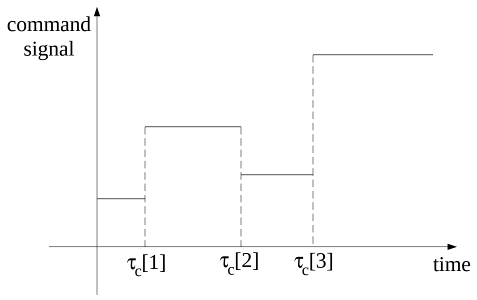

Figure 4: Command signal issued by the interface.

In such a hybrid control system, the plant taken together with the actuator and generator, behaves like a discrete event system; it accepts symbolic inputs via the actuator and produces symbolic outputs via the generator. This situation is somewhat analogous to the way a continuous-time plant, equipped with a zero-order hold and a sampler, “looks” like a discrete-time plant. 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 that 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* $\Sigma$ can be described as a triple $(T, W, B)$ with $T \subseteq \mathbb{R}$ the *time axis*, $W$ the *signal space*, and $B \subset W^T$ (the set of all functions $f : T \to W$) 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

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

and is depicted in Figure 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 is possible to design a supervisor in order to guarantee that the behavior of the closed loop system satisfies the specifications $B_{spec}$. A more accurate approximation of the plant's behavior can be obtained by considering a finer partitioning of the state space for the extraction of the DES plant.

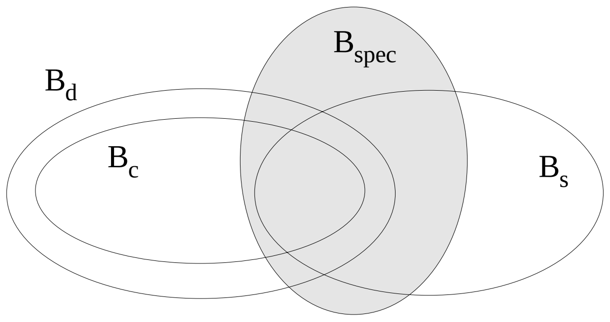

Figure 5: The DES plant model as an approximation.

An interesting aspect of the DES plant's behavior is that it is distinctly nondeterministic. This fact is illustrated in Figure 6. The figure shows two different trajectories generated by the same 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.

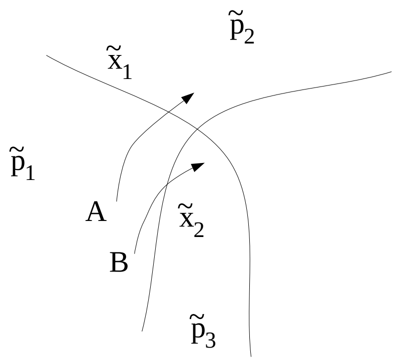

Figure 6: Nondeterminism of the DES plant model.

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 controller synthesis techniques based on discrete event systems. This work builds upon the framework of supervisory control theory initiated by Ramadge and Wonham. The main differences between the Ramadge-Wonham framework and the DES plant models of the hybrid control framework are the nondeterminism of the plant and the inability to disable plant events individually.

**Example - Thermostat/Furnace System**

The hybrid system in this example consists of a typical thermostat and furnace. Assuming the thermostat is set at 70 degrees Fahrenheit, the system behaves as follows. If the room temperature falls below 70 degrees the furnace starts and remains on until the room temperature reaches 72 degrees. At 72 degrees the furnace shuts off. For simplicity, we will assume that when the furnace is on it produces a constant amount of heat per unit time.

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

$$\dot{x} = .0042(T_0 - x) + .1r \tag{3}$$

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.

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

$$h_1(x) = x - 75 \tag{4}$$

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

The first hypersurface detects when the state exceeds 72, and the second detects when the state

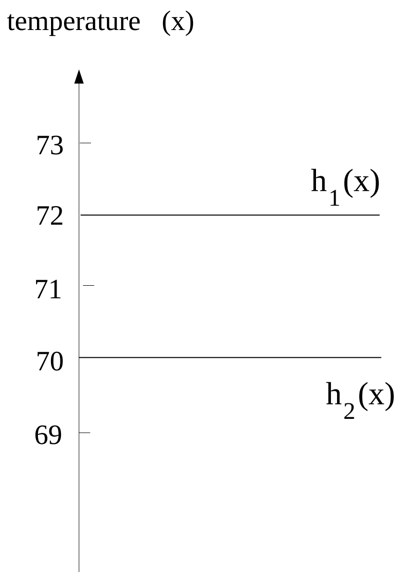

Figure 7: Partition for the thermostat/furnace system.

falls below 70. The associated functions, $\alpha_1$ and $\alpha_2$, 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 Figure 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

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

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

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

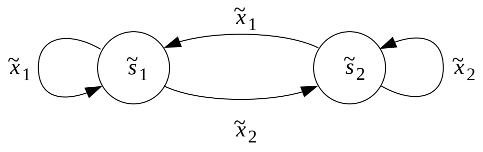
$$\begin{tikzcd}
\tilde{s}_1 \arrow[loop left, "\tilde{x}_1"] \arrow[r, bend right, "\tilde{x}_2"'] & \tilde{s}_2 \arrow[l, bend right, "\tilde{x}_1"'] \arrow[loop right, "\tilde{x}_2"]
\end{tikzcd}$$

Figure 8: Controller 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 "/" and then the plant symbols which can be generated by the transition. Notice that two of the transitions are labeled with null symbols, $\epsilon$. 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.

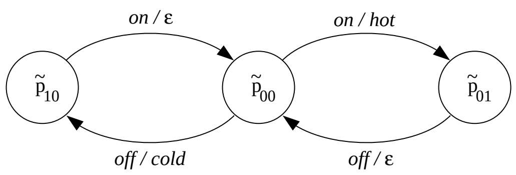

Figure 9: DES plant for the thermostat/furnace system.

## 3.2 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. Characteristics of the approaches are the fact that the models describe continuous dynamics that depend on continuous or discrete time, and that under clearly stated assumptions, properties such as reachability and stability are guaranteed. Such guarantees are of course important but they become absolutely essential in safety-critical systems such as chemical and nuclear processes, aircraft traffic control etc.

Hybrid dynamical systems 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 arise also when switching controllers are used to achieve stability and improve performance as shown in Figure 10. Typical examples of such systems are computer disk drives, constrained mechanical systems, switching power converters, and automotive powertrain applications.

Mathematically, such hybrid systems can be modeled by the equations

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

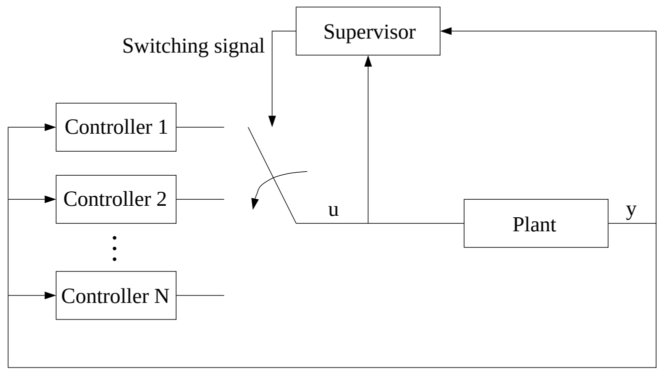

Figure 10: Switching controller feedback architecture.

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

where $x(t) \in \mathbb{R}^n$ 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 $\delta$ is the switching law that describes the logical and/or discrete-event system 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 $\alpha$. The simplified dynamics of the car are described by

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

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

where $G_i,\ i = 1, 2, 3, 4$ are the transmission gear ratios normalized by the wheel radius $R$, $k$ is an appropriate constant, $\omega$ 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

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

where $\omega_{high}$ and $\omega_{low}$ are prescribed angular velocities of the engine. The discrete state transition can be described of the finite automaton of Figure 11. □

Hybrid system stability analysis relies on classical Lyapunov stability theory. For conventional control systems, demonstrating stability depends on the existence of a continuous and differentiable

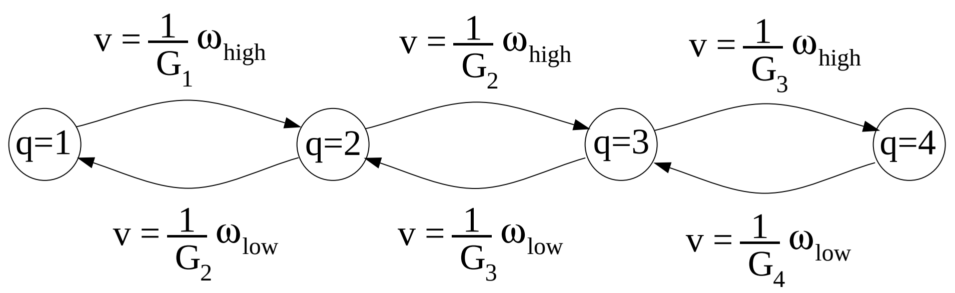
$v = \frac{1}{G_1}\omega_{high}$
$v = \frac{1}{G_2}\omega_{high}$
$v = \frac{1}{G_3}\omega_{high}$
q=1
q=2
q=3
q=4
$v = \frac{1}{G_2}\omega_{low}$
$v = \frac{1}{G_3}\omega_{low}$
$v = \frac{1}{G_4}\omega_{low}$

Figure 11: Finite automaton illustrating the switching of the gears.

Lyapunov (energy) function. In the hybrid system case, stability analysis is carried out using multiple Lyapunov functions (MLFs) in order to compose a single piecewise continuous and piecewise differential Lyapunov function that can demonstrate stability. In order to illustrate the use of MLFs, we consider the autonomous form of the hybrid system model

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

where $q(t) \in \{1, 2, \ldots, N\}$. In addition, it is assumed that there are only a finite number of switches in a bounded time interval.

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 $\bar{x} \in \Omega_i \subset \mathbb{R}^n$ is a real-valued function $V_i(x)$ defined over the region $\Omega_i$ 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 definite derivative* (for $x \in \Omega_i$ $\dot{V}_i(x) \leq 0$).

Given the system (16), suppose that each subsystem $f_i$ has an associate Lyapunov-like function $V_i$ in the region $\Omega_i$, each with equilibrium $\bar{x} = 0$ and suppose $\bigcup_i \Omega_i = \mathbb{R}^n$. 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

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

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

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

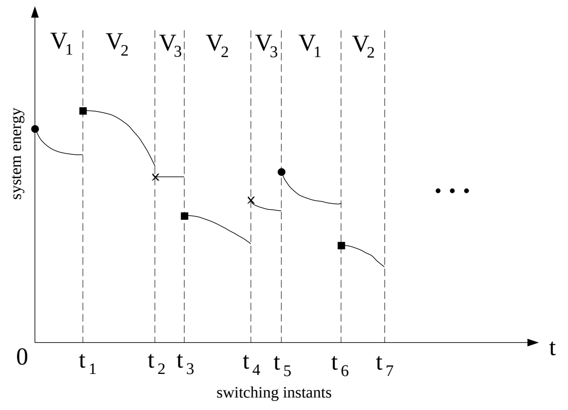

Figure 12: Stability condition.

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

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

$$q(t^+) = \delta(x(t), q(t)) \tag{19}$$

where $q(t) \in \{1, 2, \ldots, N\}$ and $A_{q(t)} \in \mathbb{R}^{n\times n}$. 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 the 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 linear switched system even when all the subsystems are unstable.

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

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

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

An important problem is to find conditions that guarantee that the switched systems $\dot{x}(t) = A_{q(t)}x(t)$ is stable for any switching signal. This situation is of great importance when a given

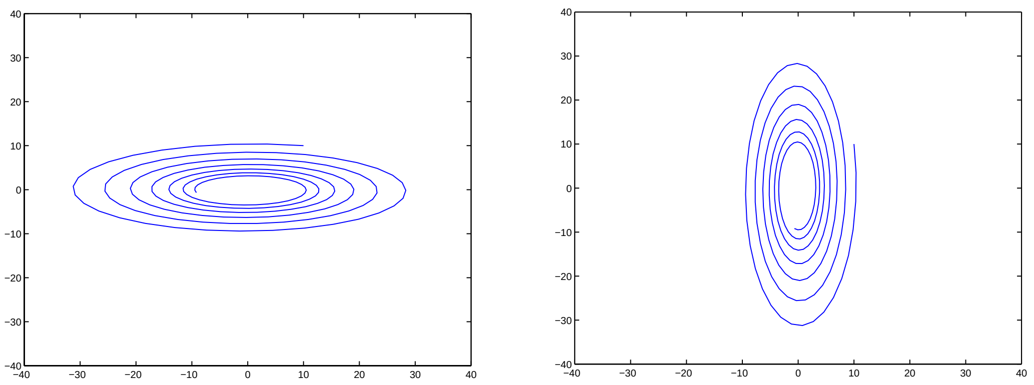
Figure 13: Unstable trajectory of switched system consisting of stable subsystems.

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 in order to allow the transient effects to dissipate. Another approach that can be used to demonstrate stability for any switching signals is to guarantee the matrices $A_i$ share a common quadratic Lyapunov function $V(x) = x^T Px$, such that $\dot{V}(x) \leq -x^T Qx, Q > 0$. These conditions on $V(x)$ are equivalent to find matrices $P$ and $Q$ that satisfy the inequalities $A_i^T P + PA_i \leq -Q$ for all $i$. Note that the existence of a common Lyapunov function, although sufficient, it 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. 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 $\Omega$-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 $\Omega_i$ denote a region where one searches for a quadratic Lyapunov function $V_i = x^T P_i x$ 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. In order to constrain the stability conditions to local regions, two steps are involved. First, the region $\Omega_i$ 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 a constrained stability condition to 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 $\Omega$-region.

In addition, the LMI formulation requires that whenever there is movement to an adjacent region $\Omega_j$ 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 as $x^T P_j x \leq x^T P_i x$. 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.

## 3.3 Hybrid Automata

Hybrid automata were introduced in the study of hybrid systems in the early 90s. Hybrid automata provide a general modeling formalism for the formal specification and algorithmic analysis of hybrid systems. They are used to model dynamical systems that consist of both discrete and analog components which arise when computer programs interact with an analog environment 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, while time elapses, through a continuous activity. The hybrid automaton in Figure 14 describes a thermostat and is used to introduce the modeling framework.

**Example** The hybrid automaton of Figure 14 models a thermostat controlling the temperature of a room by turning on and off a heater. 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 temperature falls at 70 degrees according to the condition $x = 70$. When the heater is on the temperature is rising and when it reaches 75 degrees according to the condition $x = 75$. 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 between 70 and 75 degrees. □

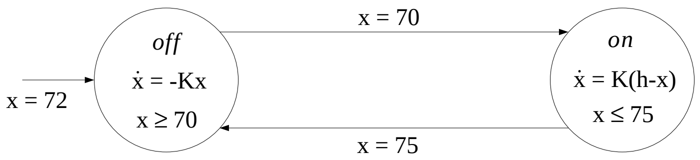

Figure 14: Hybrid automaton describing a thermostat.

A hybrid automaton consists of a finite set $X = \{x_1, \ldots, x_n\}$ of real-valued variables and a labeled directed graph $(V, E)$. The directed graph models the discrete (event) portion of the hybrid system. $V$ is a finite set of vertices and $E$ is a set of directed arcs or edges between vertices. 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 14 consists of two vertices and two edges. Note that the arc labeled by $x = 72$ 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* or *activity* 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 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* $inv(v) \in \mathbb{R}^n$ that assigns to each control mode a region of $\mathbb{R}^n$. 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 at 75 degrees.

An edge $e \in E$ is also called *control switch* or *transition* and is labeled with guarded assignment in the variables in $X$. 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 $x = m$.

A *state* $\sigma = (v, x)$ of the hybrid automaton consists of a control location $v \in V$ and a valuation $x \in \mathbb{R}^n$ of the variables in $X$. The state can change either by a discrete and instantaneous transition or by a time delay. 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

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

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 $t \in \mathbb{R} : \ 0 \leq t \leq t_i$, and (iii) $\sigma_{i+1}$ is a transition successor of $\sigma'_i = (v_i, f_i(t_i))$ and $\sigma'_i$ is a time successor of $\sigma_i$.

An important notion for the realizability of the hybrid automaton is the divergence of time. A hybrid automaton is said to be *nonzeno* if it cannot prevent time for diverging. If a hybrid automaton is nonzeno only finitely many transitions can be executed in every bounded time interval.

Another labeling function assigns to each transition an event from a finite set of events $\Sigma$. 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 15. A road is crossing 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.

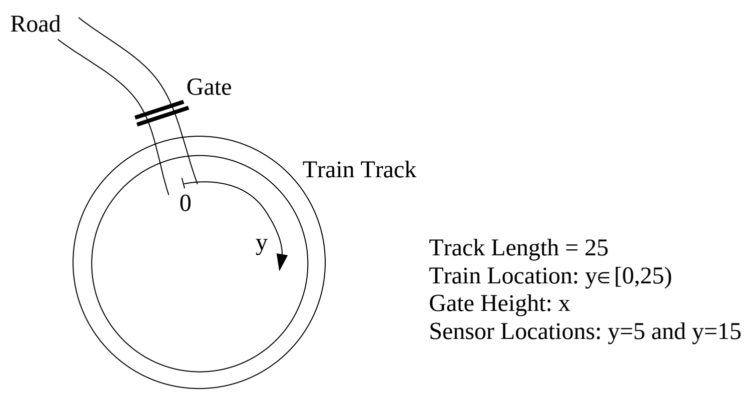

Figure 15: Train-Gate-Controller system

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 $y \in [0, 25)$. The velocity of the train is described by the differential equations $\dot{y} = f(y)$ 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 16.

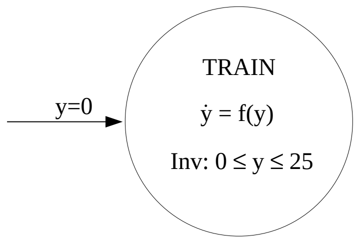

Figure 16: Hybrid automaton modeling the train

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 $\dot{x} = \frac{1-x}{2}$. When the gate is raised the height increases according to $\dot{x} = \frac{10-x}{2}$. The hybrid automaton in Figure 17 is used to model the dynamic behavior of the gate. The automaton has two control modes, *LOWER* and *RAISE*. 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 18. The controller receives information from the sensors and detects when the train reaches and 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 19. □

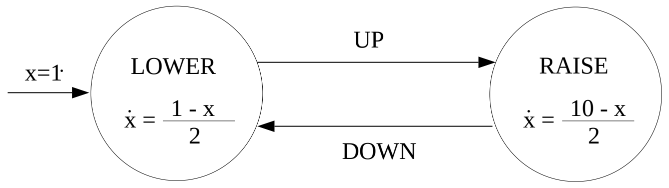

Figure 17: Hybrid automaton modeling the gate

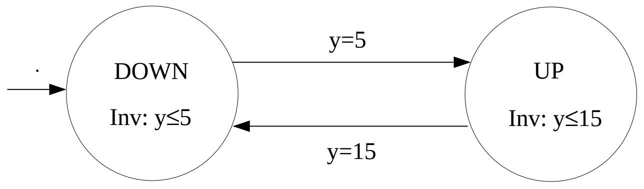

Figure 18: Hybrid automaton modeling the controller

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 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 predicate (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 as a *clock*. A discrete transition either resets the clock of 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 undecidability of verification problems of hybrid automata.

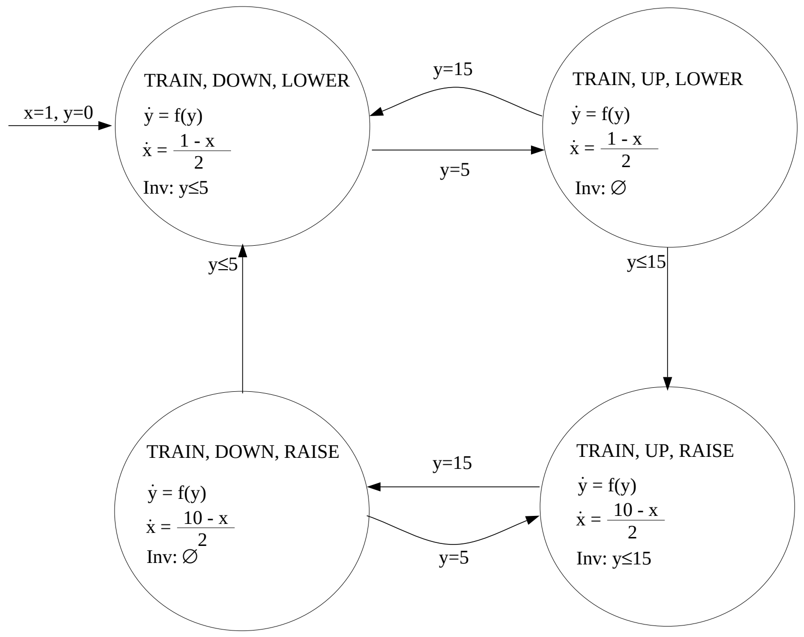

Figure 19: Hybrid automaton for the train-gate-controller system

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

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 $\mathbb{R}^n$. 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 by $pre(R)$, is the set of all states $\sigma$ such that $R$ can be reached from $\sigma$. The postcondition of $R$, denoted by $post(R)$ 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 target 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 that can be reached from the initial region $R$ 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 $R$ and 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 is must lowered ($x < 5$) whenever the train reaches the crossing. This is a safety specification that can be encoded as $y = 0 \Rightarrow x < 5$. The safety specification corresponds to a set $S$ of safe states of the hybrid automaton shown in Figure 19 which consists all four control locations and the region of $\mathbb{R}^2$ expressed by the set $\{(x, y) : \ x < 5 \wedge y = 0\}$. In order 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. $\square$

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 very important for control of batch processes. Note that even in the case the continuous dynamics of the physical system are more complicated, it is efficient to use low-level continuous controllers to impose linear ramp-like setpoints. 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 $R_v = \{x \in \mathbb{R}^n : A_v x + b_v \leq 0\}$ 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 semi-decidable algorithm is described that uses backward reachability analysis. The algorithm that 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 between a predefined number of control modes. The semi-decidability 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 been also 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 $G = \{x \in \mathbb{R}^n : l(x) < 0\}$ where $l : \mathbb{R}^n \to \mathbb{R}$ 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 from which the uncontrollable actions may be able to force the state 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 efficiently this equation.

Recently, an approach that has attracted considerable attention is based an the modeling formalism of hybrid automata and 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. The idea of using finite bisimulations for the analysis and synthesis of hybrid systems is similar to the approximation of the continuous dynamics with discrete event systems.

In summary, recent research efforts towards 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 depend on their suitability for studying specific problems.

## 4 CONCLUSIONS

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. Researchers have recognized this need and several software packages have been developed.

### Further Reading

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

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

[3] R. Alur, T. Henzinger, and E. Sontag, (eds), Hybrid Systems III-Verification and Control, Lecture Notes in Computer Science, Vol., Springer-Verlag, 1996.

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

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

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

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

[8] B. Krogh and N. Lynch, (eds), Hybrid Systems: Computation and Control, Lecture Notes in

Computer Science, Vol. 1790, Springer-Verlag, 2000.

[9] O. Maler, (ed.), Hybrid and Real-Time Systems, Lecture Notes in Computer Science, Vol. 1201, Springer-Verlag, 1997.

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

[11] Special Issue on Hybrid Control Systems, P.J. Antsaklis and A. Nerode Guest Editors, IEEE Transactions on Automatic Control,. Vol. 43, No. 4, April 1998.

[12] Special Issue on Hybrid Systems, P.J. Antsaklis and M.D. Lemmon. Guest Editors, Journal of Discrete Event Dynamic Systems: Theory and Applications, Vol. 8, No.2, June 1998.

[13] Special Issue on Hybrid Systems, J. Sifakis and A. Pnueli, Guest Editors, Journal of Theoretical Computer Science, vol 138, 1995.

[14] Special Issue on Hybrid Systems, A. Morse, C. Pantelides, S. Sastry and J. Schumacher, Guest Editors, Automatica, Vol.35, 1999.

[15] Special Issue on Hybrid Control Systems, R. Evans and A.V. Savkin, Guest Editors, Systems and Control Letters, Vol.38,1999.

[16] Special Issue on Hybrid Systems, P. Antsaklis, Guest Editor, Proceedings of the IEEE, July 2000.