# 6.43.28 Hybrid Control Systems[1]

**Karl Henrik Johansson,** Dept. of Signals, Sensors & Systems, Royal Institute of Technology, 100 44 Stockholm, Sweden, `kallej@s3.kth.se`.

**Keywords:** Hybrid system; Real-time system; Dynamical system; Control theory; Embedded software; Hierarchical control; Computer-controlled system; Discrete-event systems; Hybrid automata; Discontinuous control; Switched system; Supervisory control; Automaton; Verification; Finite state machine; Stability.

## Contents

[1]UNESCO Encyclopedia of Life Support Systems

## Glossary

**Abstraction:** The process of ignoring irrelevant details in a complicated model, while preserving properties of interest.
**Automaton:** A mathematical model for a discrete-state system.
**Composition:** The process of interconnecting hybrid systems.
**Controller:** Generates (control) signals that make a system behave in a desired way.
**Continuous-time system:** Dynamic system defined by differential equations.
**Decidability:** Guarantee of the existence of a terminating algorithm that solves a problem.
**Discrete-event system:** A discrete-state system whose state evolution depends on asynchronous events.
**Discrete-time system:** Dynamic system whose state evolution is synchronized by the system clock, often represented by a difference equation.
**Execution:** Trajectory (or solution) of a hybrid automaton.
**Finite state machine:** See *Automaton*.
**Hierarchical system:** A multi-level system that decomposes a complex system into several levels.
**Hybrid controller:** A controller that may generate both continuous-valued and symbolic control signals based on continuous-time and discrete-event dynamics.
**Hybrid system:** Dynamical system with interacting continuous and discrete components.
**Hybrid automata:** Mathematical model of a hybrid system.
**Quantizer:** Component that discretizes an analogue signal.
**Reachability:** The process of calculating the set that can be reached by the state through the evolution of the hybrid system.
**Run:** See *Execution*.
**Timed automata:** Hybrid automata with continuous dynamics given by clocks (integrators).
**Transition:** A change from one state to another.
**Verification:** Algorithmic process of analyzing whether a system satisfies a given specification.
**Zeno execution:** An execution that takes infinitely many discrete transitions in a finite amount of time.

## Summary

Most practical control systems involve both analogue and logic components. Hybrid control system is a generic term for such systems, where the continuous and the discrete dynamics are interacting. These systems arise naturally in many applications of automatic control, for example when a physical plant is controlled by a finite set of controls. Hybrid systems is a recent and very active research area, which evolves from the foundations of control theory and computer science. The developed framework is suitable for the design of complex embedded and networked control systems, which are often hard to analyze with traditional tools. Here a brief introduction to hybrid control is presented and the EOLSS articles under hybrid control systems are put into context.

## 1. Introduction

To find abstract models of technological phenomena is essential in many areas of engineering. For control system design, a good model is one that is complex enough to capture the important system characteristics (dynamics, disturbances, measurement noise etc.), but simple enough to allow application of existing analysis and design methods. Traditional control has been devoted to continuous dynamical systems, where both plant and controller are modeled by differential equations. Despite the tremendous impact this approach has had on control design, it has several limitations mainly due to that far from all real control systems can be well modeled by differential equations. Networked and embedded control systems are examples of emerging areas that have called for new modeling paradigms. High complexity and heterogeneous dynamics are common for them and most other application areas of hybrid systems. Hybrid systems provide a framework that extracts desired properties of a system while ignoring irrelevant details. Even though research projects in hybrid systems have covered a wide range of theoretical and practical topics, of which this paper only discusses a few approaches, it is important to remember that hybrid system is a young research area with most work being done over the last ten years. For instance, there does not yet exist a unified theory of hybrid systems (as for linear control systems and classes of nonlinear control systems), but the creation on such theory is very much an ongoing activity. See the bibliography in the end of the paper for literature that discusses the evolution of hybrid systems and for major collections of papers on hybrid systems such as special issues of journals in automatic control.

### 1.1. Hybrid Systems

A *hybrid* system denotes in general a system composed of two unlike components. A *hybrid control system* is a control system with both analog and digital parts. Such a system generates a mixture of continuous and discrete signals, which take values in a continuum (such as the real numbers $\mathbb{R}$) and a finite set (such as $\{a, b, c\}$), respectively.

A continuous-valued signal typically takes values in the set of real numbers (e.g., a temperature $T$ taking values in the interval $[-20, 100]$ degrees Celsius), while a discrete-valued signal takes value in a discrete (often finite) set (e.g., a thermostat valve $V$ taking the values $\{\texttt{on}, \texttt{off}\}$). The signals can either depend on time or be event-driven. A continuous-time signal is updated continuously through a differential equation (e.g., $\dot{T}(t) = -T(t) + 1$) while a discrete-time signal is updated through a difference equation (e.g., $T(t+1) = T(t) + 1$). An event-driven signal on the other hand is updated when an internal or external event happens (e.g., the switching on of the thermostat, $V^+ := \texttt{on}$, could be driven by the event that the temperature goes below 15 degrees, $T < 15$). Note that a discrete-time signal can be interpreted as an event-driven signal being updated when the event "sampling" takes place.

Let us develop a simple hybrid control system, which illustrates how continuous and discrete signals may interact. The example describes the problem of maintaining the temperature $T$ of a room at some desired level (about 19 degrees Celsius, say). If the radiator is off, the temperature dynamics is given by

$$\dot{T} = -T + 15$$

and if it is on the temperature dynamics is given by

$$\dot{T} = -T + 25.$$

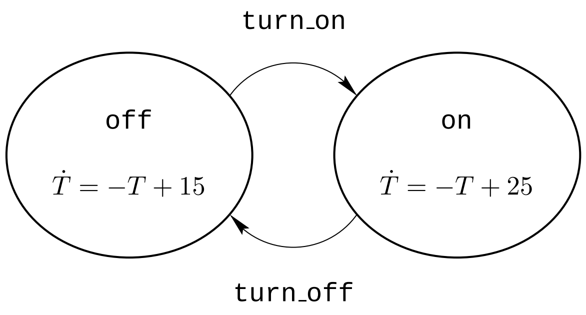
$$\begin{tikzcd}
\begin{array}{c}\texttt{off}\\ \dot{T} = -T + 15\end{array} \arrow[r, bend left, "\texttt{turn\_on}"] & \begin{array}{c}\texttt{on}\\ \dot{T} = -T + 25\end{array} \arrow[l, bend left, "\texttt{turn\_off}"]
\end{tikzcd}$$

Figure 1: Hybrid control system modeling the heating of a room. When the thermostat turns on the radiator, the hybrid system jumps from the `off` to the `on` state through the discrete transition `turn_on`. It jumps back again when the thermostat turns off the radiator. The temperature $T$, which is a continuous-time variable, is governed by one of the two differential equations depending on the current discrete state.

It is convenient to represent a hybrid system by a graph. The hybrid system describing the heating of the room can be modeled as the graph shown in Figure 1. The two vertices of the graph represent the two discrete modes of the system: the radiator is either `off` or `on`. As long as the radiator is off, the temperature $T$ will follow the dynamics specified in the left mode, i.e., $T$ will tend to 15. When the event marked `turn_on` is triggered, the discrete state of the system jumps from the `off` to the `on` mode. The transition is represented by the upper edge of the graph. In the `on` mode, the temperature follows the dynamics given by the differential equation specified in the right vertex, i.e., $T$ will tend to 25. When `turn_off` is triggered, the system returns to the `off` mode. The events `turn_on` and `turn_off` can be triggered by either external or internal signals. An external trigger can be that an operator manually turns the radiator off, that a failure causes the switch, or that a separate system (e.g., a heating system in another room) communicates the trigger. An internal trigger signal is depending on the hybrid state of the system. As an example, suppose that a thermostat is used as a heat controller. An internal trigger can then be created by identifying the event `turn_on` with the logical condition $T \leq 18$ and the event `turn_off` with the condition $T \geq 20$. For such a hybrid control system, the discrete transitions can take place only when these boolean expressions are true. The state evolution of the hybrid control system is shown in Figure 2.

For the thermostat controlled system, it is easy to see that regardless of the initial state of the system (i.e., for all initial discrete state in $\{\texttt{off}, \texttt{on}\}$ and initial continuous state $T(0) \in \mathbb{R}$), the temperature tends to an oscillate between 18 and 20 degrees. For more complicated hybrid control systems such a conclusion is not obvious, but one need theoretical and computational tools to analyze properties of hybrid control systems. These are further discussed in Section 3.

## 1.2. Applications

It is important to capture the hybrid behavior of a control system when the continuous and the discrete components of the system interact in a way that influences the overall performance. This is the case in many applications, particularly when performance measures should be improved under sustained safety constraints. Such examples are omnipresent in today's society. Hybrid system applications studied with some depth in the literature include air traffic management, chemical process control, communication networks, embedded control, engine control, and robotics. The hybridness is illustrated by the following examples:

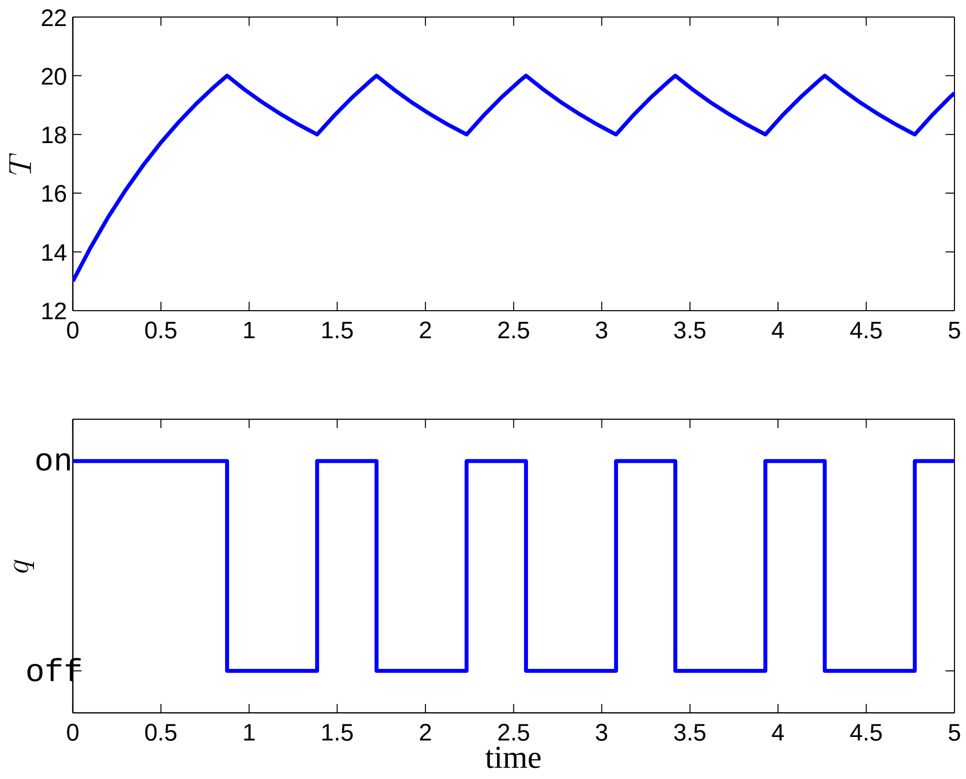

Figure 2: The evolution of the continuous and the discrete states of the hybrid control system in Figure 1. Initially the temperature is equal to $T = 15$ and the radiator is in the mode $q = \texttt{on}$. The thermostat control mechanism switches the radiator off and on as $T$ enables discrete jumps at $T = 18$ and $20$, respectively.

- Air traffic management: A finite set of manoeuvres, such as *speed change*, *short cut*, and *detour*, is used by the air traffic controller to obtain a conflict-free flight environment in which the aircrafts meet their dynamic and other constraints.
- Chemical process control: To produce a substance, an instruction sequence is designed, in which each instruction could involve one or more continuous control elements (e.g., keeping a reference temperature while a chemical reaction takes place).
- Communication networks: Large data flows are conveniently modeled as continuous variables (in order to reduce the model complexity), while traffic control mechanisms such as routing induce discrete controls.
- Embedded control: A micro-computer embedded in a physical device has an inherently discrete behavior (e.g., due to finite-precision computations and quantization of signals), but it interacts with a continuous environment through actuators and sensors.
- Engine control: A four-stroke gasoline engine is naturally modeled using four discrete modes corresponding to the position of the pistons, while combustion and power train dynamics are continuous.
- Robotics: A manipulator is accurately governed by continuous dynamics, but impacts and load shifting cause discrete and asynchronous changes.

In classical control design, it is common to separate the event-driven dynamics and the continuous-time dynamics both in the analysis and in the design. For example, in a manufacturing process, one can model the processing of individual machines by their service times. In this way, the composed discrete-event system for the whole production line can be represented by for example a Petri net and analyzed using queuing theory. The control performance and quality assessment of each machine are modeled using continuous dynamics. The continuous feedback control is constrained by the service time specifications. If they are fulfilled, the higher-level discrete-event system is a suitable model for the overall evolution of the manufacturing system. The separation of asynchronous and synchronous controls will in many cases, however, lead to a too conservative design. In the manufacturing example, this could result in too large buffers and inefficient use of the machines. If instead all the dynamics and the interactions in the manufacturing process are captured within one single hybrid model, it is possible to optimize the overall behavior and thus achieve a high-performance design. Tools in hybrid control systems address this type of problems. In the development of these tools, it has been shown that similar type of improvement can be achieved in many different areas. A thorough discussion on the application of hybrid methods to air traffic management is given in [EOLSS,6.43.28.8]. There it is shown not only that the hybrid system formalism allows to capture the dynamics of existing air traffic control systems, but also how new strategies enabling free flight can be derived.

## 2. What is a Hybrid Control System?

Hybrid control systems form a richer class of systems than ordinary control systems. The continuous flow is in general influenced not only by the continuous control, but also by the discrete mode. Similarly, the discrete dynamics are affected by both discrete control actions and, indirectly, by the continuous flow. In addition to control inputs, there might be both continuous and discrete disturbances acting on the system. Therefore, in its full generality, a hybrid control system can be a rather complicated object.

### 2.1. Continuous and Discrete Control Systems

Recall the definition of continuous and discrete control systems by the following two examples. A continuous control system [EOLSS,6.43.21] can be represented by the differential equation

$$\dot{x} = f(x, u),$$

where $x$ is the continuous state of the system, $u$ is the continuous control, and $f$ is a (typically Lipschitz continuous) function. The state space and the control space are continuous sets, e.g., $\mathbb{R}^n$ and $\mathbb{R}$, respectively. The state and control are defined over the continuous time axis and can thus be represented as functions, e.g., $x : [t_0, t_1] \mapsto \mathbb{R}^n$ and $u : [t_0, t_1] \mapsto \mathbb{R}$.

A discrete control system [EOLSS,6.43.27] on the other hand is governed by the equation

$$q^+ = g(q, v),$$

where $q$ is the discrete state of the system, $v$ is the discrete control, $g$ is a transition function, and $q^+$ denotes the new state.[2] The state space and the space of controls form finite sets, e.g., $\{q_1, \ldots, q_N\}$

[2]A discrete control system here corresponds more or less to a (finite) automaton in computer science. It should be distinguished from a *discrete-time* (or digital) control system, cf., [EOLSS 6.43.4]. Recall that a discrete-time control system is often governed by a difference equation $x_{k+1} = f(x_k, u_k)$, where the state space and the control space are continuous sets.

and $\{v_1, \ldots, v_K\}$. The discrete state and the discrete control are *not* defined over the time axis, but only form a string of symbols. One cannot hence distinguish when transitions have taken place, but only the order of them. A discrete control system is conveniently represented by an automaton (or finite state machine), where the vertices of the automaton represent the discrete states and the edges are defined by the transition function (control).

Note the similarities and differences between these two classes of control systems. Given the control signals, an evolution of the corresponding states is generated for both the continuous and the discrete systems. For the continuous system, typically the evolution is deterministic. For the discrete system, the evolution may very well be non-deterministic: different state sequences (“trajectories”) can be generated by the same sequence of controls.

Next we let the continuous and the discrete control systems interact, and thereby form a model of a *hybrid* control system. It should be emphasized here that the particular class of hybrid control system discussed next, namely, hybrid automata, is far from the only one in the literature. A hybrid automaton is however a quite general definition of a hybrid system, in the sense that many other classes of hybrid systems discussed in the literature are special instances of hybrid automata. See the bibliography and the EOLSS articles on hybrid control systems for variations.

## 2.2. Hybrid Automaton

A hybrid automaton is a dynamical system that describes the evolution in time of the values of a set of discrete and continuous state variables. Let us first define an autonomous hybrid automaton, i.e., a hybrid system without any continuous or discrete controls.

**Definition 2.1 (Hybrid Automaton)** *A hybrid automaton $H$ is a collection $H = (Q, X, f, \mathrm{Init}, D, E, G, R)$, where*

- *$Q$ is a set of discrete states;*
- *$X$ is a set of continuous states;*
- *$f : \mathbf{Q} \times \mathbf{X} \to \mathbf{X}$ is a vector field;*
- *$\mathrm{Init} \subset \mathbf{Q} \times \mathbf{X}$ is a set of initial states;*
- *$E \subset \mathbf{Q} \times \mathbf{Q}$ is a set of edges;*
- *$D : \mathbf{Q} \to 2^{\mathbf{X}}$ is a domain;*[3]
- *$G : E \to 2^{\mathbf{X}}$ is a guard condition;*
- *$R : E \times \mathbf{X} \to 2^{\mathbf{X}}$ is a reset map.*

We refer to $(q, x) \in \mathbf{Q} \times \mathbf{X}$ as the state of $H$. Roughly speaking, hybrid automata define possible evolutions for their state. Starting from an initial value $(q_0, x_0) \in \mathrm{Init}$, the continuous state $x$ flows according to the vector field $f(q_0, \cdot)$, while the discrete state $q$ remains constant. The continuous evolution can go on as long as $x$ remains in $D(q_0)$. If at some point $x$ reaches a guard $G(q_0, q_1)$, for

[3] $2^{\mathbf{X}}$ denotes the power set (set of all subsets) of $\mathbf{X}$.

some $(q_0, q_1) \in E$, the discrete state may change value to $q_1$. At the same time the continuous state gets reset to some value in $R(q_0, q_1, x)$. After this discrete transition, continuous evolution resumes and the whole process is repeated. With a slight abuse of notation, we use $q$ and $x$ to denote the evolution of the discrete and continuous parts of the state, respectively. The collection of these two maps $(q, x)$ is denoted an *execution* of the hybrid automaton.[4]

As we saw in the previous section, it is convenient to visualize hybrid automata as directed graphs $(\mathbf{Q}, E)$ with vertices $\mathbf{Q}$ and edges $E$. To each vertex $q \in \mathbf{Q}$, we associate a set of initial states $\mathrm{Init}_q = \{x \in \mathbf{X} \ : \ (q, x) \in \mathrm{Init}\}$, a vector field $f(q, \cdot)$ and a domain $D(q)$. With each edge $e \in E$, we associate a guard $G(e)$ and a reset map $R(e, \cdot)$.

Note that both purely discrete and purely continuous systems fit within the hybrid automaton model. A discrete system can be modeled as a hybrid automaton with no or trivial continuous dynamics ($\dot{x} = 0$). A continuous-time system $\dot{x} = f(x)$ can be modeled as a hybrid automaton with a single discrete state and domain equal to the whole continuous state space.

The hybrid automaton defined above is autonomous. It is, however, straightforward to introduce inputs and outputs, so that the composition of hybrid automata can be studied. Basically, one can let the vector field, the guard condition, and the reset map depend on the continuous and the discrete inputs, and let the continuous and discrete outputs depend on the hybrid state. In this way, it is for instance possible to define a feedback control system, where both the plant and the controller is given by a hybrid automaton.

In order to properly analyze a hybrid automaton, one needs to formerly introduce the meaning of an execution. In particular, it is important to be aware of properties such as non-determinism, blocking, and Zeno. These issues are discussed in [EOLSS,6.43.28.1–2].

We illustrate the notation by specifying the hybrid automaton for the heating system discussed in previous section and shown in Figure 1. The (discrete and continuous) state space of the thermostat hybrid automaton is given by

- $Q = \{q\}$ and $\mathbf{Q} = \{\texttt{on}, \texttt{off}\}$;
- $X = \{T\}$ and $\mathbf{X} = \mathbb{R}$;

which means that there is one discrete state (denoted $q$) and one continuous state (denoted $x$). The discrete state can only take the values `on` and `off`, while the continuous state can take any real value. The vector fields in the two discrete states are

- $f(\texttt{on}, T) = -T + 15$ and $f(\texttt{off}, T) = -T + 25$.

This means for example that when $q = \texttt{on}$, then the continuous dynamics is governed by the differential equation $\dot{T} = -T + 15$. The initial state is assumed to be the whole state space, i.e.,

- $\mathrm{Init} = \mathbf{Q} \times \mathbf{X}$.

[4]Alternative names of an execution include *run* and *(hybrid) trajectory*.

The possible discrete transitions are given by the edges, which are

- $E = \{(\texttt{on}, \texttt{off}), (\texttt{off}, \texttt{on})\}$.

Let us model the heating system with the thermostat controller in place, i.e., identifying the event turn_on with the condition $T \leq 18$ and the event turn_off with $T \geq 20$. For the state evolution shown in Figure 2, we implicitly assumed that discrete transitions take place as soon as the guard is enabled (i.e., as soon as the conditions $T \leq 18$ and $T \geq 20$, respectively, are fulfilled). For the hybrid automaton, the interplay between the domains, guards, and resets define the discrete dynamics. The thermostat system in Figure 1 is complete by setting the following components:

- $D(\texttt{on}) = \{T \in \mathbb{R} : \ T \geq 18\}$ and $D(\texttt{off}) = \{T \in \mathbb{R} : \ T \leq 20\}$;
- $G(\texttt{on}, \texttt{off}) = \{T \in \mathbb{R} : \ T \leq 18\}$ and $G(\texttt{off}, \texttt{on}) = \{T \in \mathbb{R} : \ T \geq 20\}$;
- $R(\texttt{off}, \texttt{on}, T) = R(\texttt{on}, \texttt{off}, T) = T$.

The continuous evolution *may* continue as long as the continuous state belongs to the domain. When a guard condition is fulfilled, a discrete transition *may* take place. In this example, the intersections $D(q) \cap G(q, q')$ consist of single points ($T = 18$ and 20, respectively), so discrete transitions are deterministic. A non-deterministic variation is discussed in next section.

The reset maps of the thermostat hybrid automaton are equal to identity maps, i.e., at a discrete transition the continuous state is kept the same: $x' := R(e, x) = x$ for all $e \in E$. A simple example of a hybrid automaton with non-identity reset maps is a bouncing ball that loses a fraction of its energy at each bounce. This can be modeled as $v := -cv$, where $v$ is the velocity of the ball and $c^2 \in (0, 1)$ the coefficient of restitution.

## 3. Analysis and Design of Hybrid Control Systems

There are many reasons for using hybrid system techniques in analysis and design of control systems. The overall motivation for hybrid methods is significant interaction between the continuous and the discrete parts of a system, as can be the case in the discrete planning of continuous processes. Another important reason for hybrid techniques is the need for hierarchical organization of controls in many of today's complex technical systems, such as in flexible manufacturing. Hybrid system theory also provides a convenient framework for modeling engineering systems with multiple time scales, where fast dynamics can be abstracted away and be treated as discrete changes affecting slower dynamics. In this section we describe some of the existing methods and tools for modeling, analysis, and design of hybrid control.

### 3.1. Modeling

The distinguishing characteristic of hybrid systems is the interaction between a continuous-time and a discrete-event component. By modeling these different components using differential equations and finite state automata, it is possible to represent a wide range of phenomena present in physical and technological systems. There are several reasons for why models of hybrid systems are needed including that (i) modeling abstractions, such as hierarchical organization of complex systems and time-scale separations of continuous components, naturally lead to a mixture of continuous and discrete behaviors, (ii) high-level control of physical plants often consists of sequential planning algorithms, executed either by computers or humans, with safety and quality constraints depending on continuous variables, and (iii) rigorous modeling of embedded control systems requires a hybrid formalism in order to capture the interaction of the logics implemented through hardware and software, the continuous dynamics of a physical plant, and their interfacing components as sampling devices, quantizers, and communication channels.

Computer scientists have extended their traditional model checking methods to deal with real-time and multi-processor systems. This led first to timed automata and then to hybrid automata, as both being generalizations of finite state machines with continuous-valued dynamical processes. Timed automata form a class of hybrid automata with particularly simple continuous dynamics, since the vector field of a timed automaton is constant and equal to $(1, \dots, 1)^T$ in each discrete state. These models have been very influential in the development of the hybrid systems theory. They evolved from research on real-time systems, where the constant vector field represents the evolution of clocks. Distributed computer systems with asynchronous clocks can be modeled in this way, and for example the functionality of communication protocols can be verified. Timed automata are particularly attractive since the verification problem for them is decidable, which basically means that it is possible to find a terminating algorithm that tells if the system satisfies a given specification. So even if a timed automaton has an infinite state space (thanks to the continuous dynamics), it appears as a finite state machine from an analysis point of view.

For a model of a hybrid system, as the hybrid automaton introduced in previous section, it is fundamental to understand if there are any executions (trajectories) that satisfy the equations of the system; i.e., whether or not the hybrid system is well-posed. In general this is a non-trivial question, because the hybrid system provides an implicit description of the system behavior, in the sense that a mathematical problem needs to be solved to find the system evolution. Examples show that the well-posedness issue is considerably more complex in hybrid systems than in continuous systems, as a result of a number of factors including the interaction of guards and domains, possible presence of sliding modes, and the occurrence of accumulations of event times (Zeno executions). Let us illustrate with a few examples.

Consider a simple hybrid system defined by $\dot{x} = -1$ if $x > 0$ and $\dot{x} = 1$ if $x < 0$, with the discrete transitions taking place when $x$ crosses zero. With $x(0) = x_0$, the continuous trajectory of the system is equal to

$$x(t) = x_0 - t \operatorname{sgn} x_0, \quad 0 \le t < x_0.$$

For $t = x_0$, we have $x(t) = 0$, so the discrete transitions are constantly enabled and will thus fire without any time progress. In this sense, the continuous evolution of the system is not defined for $t > x_0$. With a more general notion of solution, such as Filippov solutions and sliding modes as considered in variable structure systems [EOLSS,6.43.21.14], we note that our hybrid system example corresponds to the discontinuous differential equation

$$\dot{x} = -\operatorname{sgn} x.$$

For $t > x_0$, the sliding mode is governed by $\dot{x} = 0$, which hence suggests the continuation of the trajectory as $x(t) = 0$ for $t \ge x_0$. This example shows that an execution of a hybrid system may not be defined over the whole time axis, and it thus illustrates the importance of verifying the existence of executions.

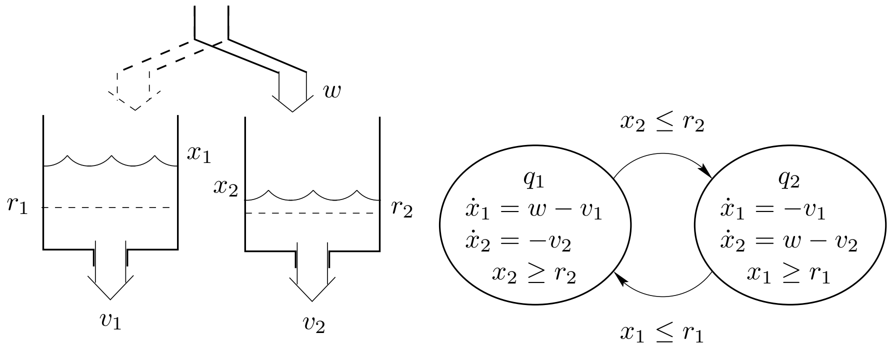

Figure 3: Water tank system and the corresponding hybrid system.

Consider again the thermostat system. Earlier we assumed that discrete transitions take place as soon as the guard is enabled. This might be a reasonable assumption for some thermostat systems, but often, however, it is convenient to model more general transitions. For example, to allow non-deterministic transition (that a transition may or may not take place). Let now the domains and guards be given by

- $D(\texttt{on}) = \{T \in \mathbb{R} : T \geq 18\}$ and $D(\texttt{off}) = \{T \in \mathbb{R} : T \leq 20\}$;
- $G(\texttt{on}, \texttt{off}) = \{T \in \mathbb{R} : T \leq 20\}$ and $G(\texttt{off}, \texttt{on}) = \{T \in \mathbb{R} : T \geq 18\}$.

Hence, $D(\texttt{on}) \cap G(\texttt{on}, \texttt{off}) = D(\texttt{off}) \cap G(\texttt{off}, \texttt{on})$ is equal to the interval $[18, 20]$. The discrete transition can take place anywhere in this interval, and thus the execution of the hybrid automaton is non-deterministic. Although the behavior of thermostat is uncertain, one can still analyze interesting question for the closed-loop system, such as "Will the temperature always stay between 15 and 25 degrees?"

Next we consider an example that exhibit the so called Zeno phenomenon. Consider the two-tank system shown in Figure 3. For $i \in \{1, 2\}$, let $x_i$ denote the volume of water in Tank $i$ and $v_i > 0$ denote the constant flow of water out of Tank $i$. Let $w$ denote the constant flow of water into the system, dedicated exclusively to either Tank 1 or Tank 2 at each time instant. The objective is to keep the water volumes above $r_1$ and $r_2$, respectively, assuming that the water volumes are above $r_1$ and $r_2$ initially. This is to be achieved by a controller that switches the inflow to Tank 1 whenever $x_1 \leq r_1$ and to Tank 2 whenever $x_2 \leq r_2$. The water tank system can be represented by the hybrid system of Figure 3. Suppose that at the initial time, it holds that $x_1 > r_1$ and $x_2 > r_2$. If $\max(v_1, v_2) < w < v_1 + v_2$, physical intuition suggests that at least one of the water tanks will eventually drain. In the hybrid model this leads to an accumulation of jump instances, which is known as the Zeno phenomenon. This is a truly hybrid phenomenon, in the sense that it requires the interaction between continuous and discrete behavior. It can not even be formulated for a purely discrete system without the notion of continuous time. A Zeno execution for the water tank system with $\max(v_1, v_2) < w < v_1 + v_2$ is shown in Figure 4. The name Zeno refers to the philosopher Zeno of Elea (ca. 500–400 B.C.), whose major work consisted of a number of paradoxes, designed

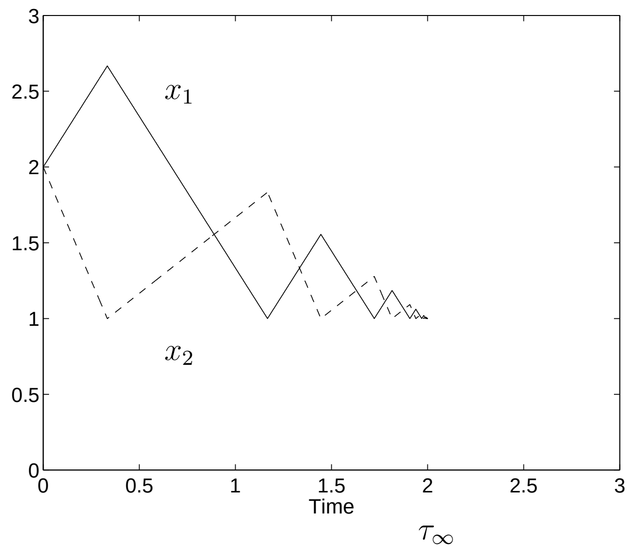

Figure 4: A Zeno execution for the water tank hybrid system.

to support his view that the concepts of plurality and motion lead to contradictions. An example is Zeno's Second Paradox of Motion, in which Achilles is racing against a tortoise.

Models of hybrid systems and their fundamental dynamical properties are discussed in [EOLSS,6.43.28.1–2]. An interesting class is the so call complementarity systems, which borrow their name from complementarity problems in mathematical programming. In particular for linear complementarity systems there exist efficient computational methods for investigating existence and uniqueness of solutions, see [EOLSS,6.43.28.2]. These models can be used to model impact mechanics and electrical networks.

Computer simulation is important in the analysis and design of hybrid control systems, particularly since the complexity of these system limit the application of analytical methods. Hybrid systems are in general difficult to simulate due to the nonsmooth characteristics of the state evolution. Therefore specific classes of numerical solvers have to be used in order to get efficient and accurate results. Software tools for hybrid systems include Dymola, gPROMS, OmSim, SHIFT, and StateFlow in Simulink. Modelica is an object-oriented modeling language for dynamical systems that is (simulator) platform independent, in a similar way as many programming languages. Modelica supports the hybrid systems formalism. Modeling and simulation of hybrid system is discussed in [EOLSS,6.43.7.3], where references for the tools listed above are given.

## 3.2. Analysis

Stability and verification are central in the analysis of hybrid control systems. While stability theory forms the foundation of classical control theory, formal verification finds its origin in theoretical computer science. We discuss both concept here and illustrate them on simple hybrid systems.

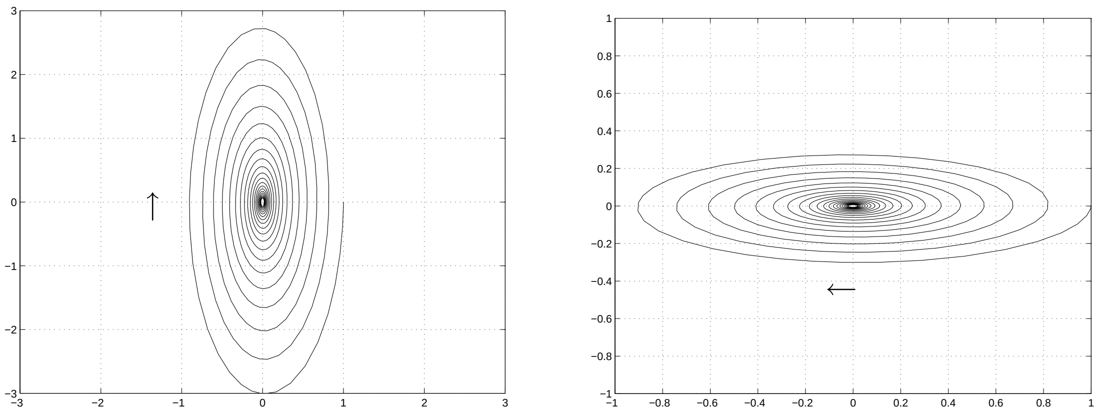

Figure 5: Phase portraits for $\dot{x} = A_1 x$ (left) and $\dot{x} = A_2 x$ (right), which both are stable foci. As described in [EOLSS,6.43.28.3], a hybrid system based on switching between these continuous systems can still be unstable.

### 3.2.1. Stability

Stability is fundamental in control theory [EOLSS,6.43.2]. It basically states that small perturbations in operating conditions lead to small changes in the system behavior. In generalizing the notion of stability to hybrid systems, one need to pay attention not only to the evolution of the continuous state but also to the discrete state. An equilibrium point of a hybrid system can either be in the interior of a domain or on the boundary between two or more domains. A simple example that illustrates stability of hybrid systems is discussed in [EOLSS,6.43.28.3]. The continuous dynamics of the example is given $\dot{x} = A_q x$, $q = 1, 2$, with

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

Both $\dot{x} = A_1 x$ and $\dot{x} = A_2 x$ are stable (eigenvalues in $-1 \pm i10\sqrt{10}$), so the trajectories of the continuous systems are spiraling towards the origin as shown in the phase portraits in Figure 5. Let us now define the discrete state of a hybrid system $H_1$ from partitioning the continuous state space: $q = 1$ when $x$ is in the first or the third quadrant, and $q = 2$ when $x$ is in the second or fourth quadrant. Then the origin is a stable equilibrium for $H_1$ since each execution spirals towards the origin. The executions consist of segments of trajectories of $\dot{x} = A_1 x$ and $\dot{x} = A_2 x$. If the hybrid system is defined slightly different, the origin might be unstable. For example, consider a hybrid system $H_2$ for which $q = 1$ when $x$ is in the second or fourth quadrant, and $q = 2$ when $x$ is in the first or third quadrant. Then each execution again consists of segments of trajectories of the continuous systems, but this time their concatenation spirals away from the origin. The example illustrates that for the stability analysis of hybrid systems it is not sufficient to study only the continuous or the discrete dynamics, but their interaction.

Lyapunov's direct and indirect methods, which are powerful tools for analyzing stability of differential and difference equations [6.43.21.5], have been extended to hybrid systems. The direct method deals with finding an energy-like *Lyapunov function* that decreases along the trajectories of the system. The idea carries straightforwardly over to hybrid systems, as long as the Lyapunov function fulfills certain conditions at the discrete transitions, see [EOLSS,6.43.28.3]. A major challenge in applying the method is in finding a suitable Lyapunov function. There exist no systematic methods for finding a Lyapunov function for a general nonlinear system, and thus not for a general hybrid system. For a linear system $\dot{x} = Ax$, a Lyapunov function is given by $V(x) = x^T P x$, where the symmetric and positive definite matrix $P$ fulfills the *Lyapunov equation* $PA + A^T P = -I$. Piecewise linear systems is a subclass of hybrid systems, for which the domains are disjoint polyhedra covering a connected subset of the continuous state space, and the continuous dynamics is given by linear systems. Computational methods for finding piecewise quadratic Lyapunov functions for classes of piecewise linear systems exist. These are based on combining Lyapunov functions for the individual linear systems, see [EOLSS,6.43.28.7]. Lyapunov's indirect method investigates stability through linearization of the dynamics about the equilibrium point. For hybrid systems, not only the vector field needs to be linearized, but also the reset map.

An important class of hybrid systems is systems with discrete transitions generated by a switching signal $\sigma : [0, \infty) \to \mathbf{Q}$, which for example can represent a control signal or an external disturbance. An example of such a system is given by $\dot{x} = A_\sigma x$. Stability analysis methods for various dynamics and switching signals are presented [EOLSS,6.43.28.7]. See also the related problem on finding a supervisor which generates a stabilizing $\sigma$ discussed in the section on control design below.

### 3.2.2. Verification

Verification is the formal process of analyzing whether a system satisfies a desired specification using a computer algorithm. A verification problem is called decidable if there exists an algorithm that solves the problem in a finite number of steps. Following early work on the verification of digital circuits, the hybrid formalism and tools have been subsequently extended to the verification of embedded software, real-time communication protocols, air traffic control, process control systems etc. Today there exist several software tools, such as HyTech, KRONOS, and UPPAAL, for the verification of various classes of hybrid systems. Verification of hybrid systems is a very active area of research, since many applications are still too complicated to be addressed by existing tools.

A common property to check with a verification algorithm is if the execution of a hybrid system enters a specified unsafe set $U \subset \mathbf{Q} \times \mathbf{X}$. For example, for a hybrid system modeling the coordination of two aircrafts, the unsafe set can correspond to the states in which the aircrafts are too close to each other. A state $(q, x)$ of a hybrid automaton $H$ is called reachable, if starting at some initial state $(q_0, x_0)$ an execution of $H$ can end up in $(q, x)$. The set of states reachable by $H$ is denoted $\mathrm{Reach}_H \subset \mathbf{Q} \times \mathbf{X}$.[5] The verification problem is then to decide if the reachable set of the hybrid system does not intersect the unsafe set, that is, to decide if $\mathrm{Reach}_H \cap U$ is empty. Algorithms for deriving reachable sets for continuous, discrete and hybrid systems are discussed in [6.43.28.6]. The reachability problem for timed automata ($f(q, x) = (1, \dots, 1)^T$) is decidable, in the sense that we can for a timed automata find a finite-step algorithm that determines if $\mathrm{Reach}_H \cap U = \emptyset$. The verification of hybrid automata, however, is in general undecidable. This is not surprising, because even for a continuous system $\dot{x} = f(x)$, it is difficult to derive the reachable set. Often this requires numerical integration of the vector field $f$, so the resulting reachable set is in practice only an approximate estimate of the reachable set. Intensive research is devoted to finding efficient algorithms for estimating the reachable set for classes of hybrid systems, e.g., hybrid systems with an affine vector field $f(q, x) = A_q x + B_q$ in each domain. Another research direction is to locate the boundary between decidable and undecidable hybrid automata. This is further discussed in [EOLSS,6.43.28.4], where extensions of timed automata are treated to multirate automata ($f(q, x) = c$, where $c$ is a constant vector not necessarily equal to $(1, \dots, 1)^T$) and to rectangular automata ($f(q, x)$ is of the form $I_1 \times \cdots \times I_n$, where each $I_i$ is an interval). In order to reduce the complexity of formally verifying whether a hybrid system satisfies a given property, it is reasonable to try to extract a simplified model by ignoring irrelevant details, while preserving the properties of interest. Such a process of abstraction is crucial in many engineering

[5] A more general defi nition of reachable set is needed for systems with disturbances and controls, see [6.43.28.6].

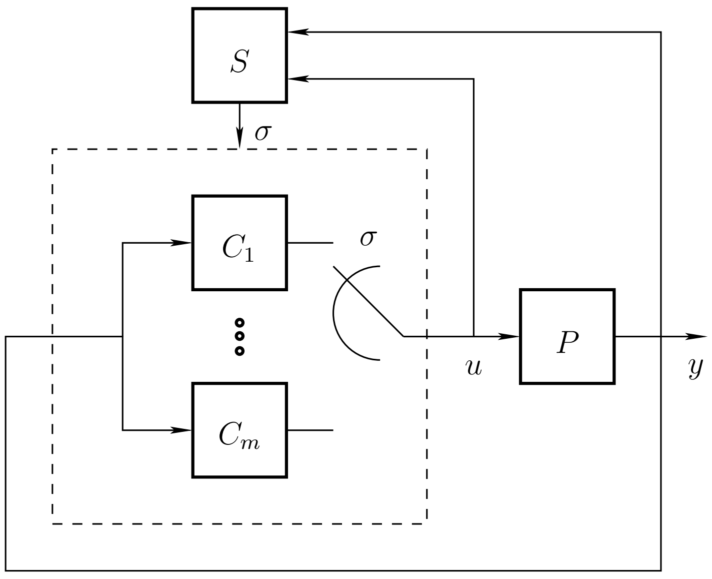

Figure 6: Supervisory control.

areas, and is surveyed for discrete, continuous, and hybrid systems in [EOLSS,6.43.28.4].

### 3.3. Control Design

There is a wide range of control design problems for hybrid systems, see the paper collections listed in the bibliography. Here we limit the discussion to supervisory control and optimal control of hybrid systems, but examples of related topics include model predictive control [EOLSS,6.43.16] of hybrid systems, quantized control, and variable structure control [EOLSS,6.43.21.14].

#### 3.3.1. Supervisory Control

A supervisory control system is shown in Figure 6 and represents a certain class of hybrid control systems. A supervisor $S$ decides which of the controllers $C_1, \ldots, C_m$ that should be active at each time instant. The switching signal $\sigma : [0, \infty) \to \{1, \ldots, m\}$ is determined by the supervisor based on the control signal $u$ and the measurement $y$. The purpose of the supervisor can be to stabilize the plant despite large uncertainties or disturbances. A supervisory control scheme can also be interpreted as so called gain scheduling, which is a common approach to handle large variations in operating conditions: controllers are designed for a few operating conditions and then during operation, the supervisor chooses the controller closest to the current condition as defined by the state of the system. Supervisory control and stabilization through hybrid control is further discussed in [EOLSS,6.43.28.7] and supervisory control of discrete event systems in [EOLSS,6.43.27.2].

#### 3.3.2. Optimal Control

The purpose of optimal control theory is to give a systematic method to synthesize control laws with properties specified by an optimization criterion or a cost function. There is a large literature on optimal control of continuous-time and discrete-time systems both in deterministic and stochastic settings, see [EOLSS,6.43.13.13–14,6.43.18.4–5]. A few problem formulations and approaches to optimal control of hybrid systems have recently been studied, but the area is still in its infancy. One formulation of an optimal control problem is to minimize the cost function

$$\int_0^T L(x(t), q(t), u(t))dt$$

with respect to the control $u$ and constrained by the dynamics of the hybrid system. An example of a control problem utilizing both continuous and discrete control is provided by a car with a gear box having four gears, as illustrated by the hybrid system in Figure 7. The longitudinal position of the car along the road is denoted by $x_1$ and its velocity by $x_2$ (lateral dynamics are ignored). The model has the control $u = (\texttt{gear}, v)$, where $\texttt{gear} \in \{1, \ldots, 4\}$ denotes the gear position and $v \in [v_{\min}, v_{\max}]$ the throttle position. Gear shift is necessary because little power can be generated by the engine at very low or very high engine speed. The function $\alpha_i$ represents the efficiency of gear $i$. A hybrid optimal control problem for this simple car model is to find a strategy to drive from $x = (a, 0)$ to $(b, 0)$ in minimum time? The problem is non-trivial if the reasonable assumption is included that each gear shift takes a certain amount of time. See [EOLSS,6.43.28.5] for a further discussion on optimal control of hybrid systems.

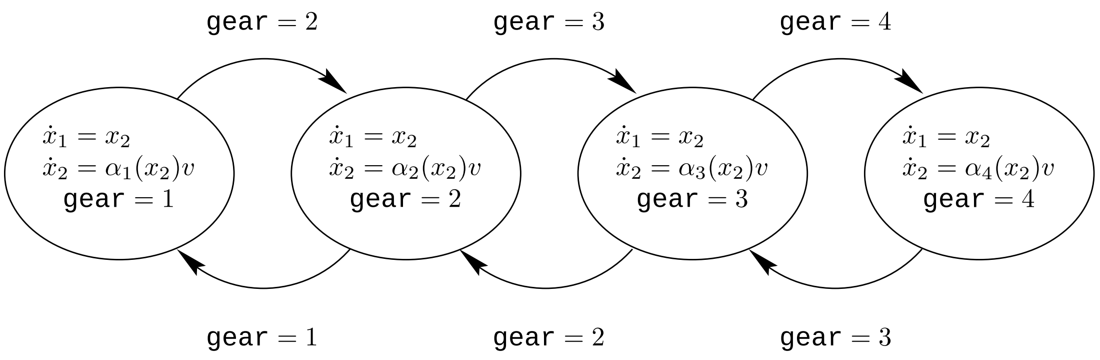

Figure 7: A hybrid system modeling a car with four gears.

## Acknowledgements

The author would like to thank Heinz Unbehauen for the opportunity to contribute to EOLSS and for his patience. The author is also grateful for comments on the manuscript by Mikael Johansson and remarks on gain scheduling by Douglas Leith. The paper is influenced by earlier collaboration with Magnus Egerstedt, John Lygeros, Shankar Sastry, Slobodan Simić, and Jun Zhang. Their contribution is gratefully acknowledged. The work was partially supported by the European Commission trough the RECSYS project and the Swedish Research Council.

## Bibliography

Alur R., Courcoubetis C., Halbwachs N., Henzinger T., Ho P.H., Nicollin X., Olivero A., Sifakis J., Yovine S. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science* **138**, 3–34. [Defines hybrid automata and the corresponding verification problem.].

Alur R., Dill D. (1994). The theory of timed automata. *Theoretical Computer Science* **126**, 193–235. [Introduces timed automata for modeling real-time systems.].

Antsaklis P.J. (2000a). A brief introduction to the theory and applications of hybrid systems. *IEEE Proceedings* **88**(7), 879–887. [Introduction to hybrid systems and outline of special issue on hybrid control systems.].

— (2000b). Special issue on hybrid systems: Theory and applications. *IEEE Proceedings* **88**(7).

Antsaklis P.J., Nerode A. (1998). Special issue on hybrid control systems. *IEEE Transactions on Automatic Control* **43**(4).

Brockett R.W. (1993). Hybrid models for motion control systems. In H. Trentelman, J. Willems, eds., *Essays in Control: Perspectives in the Theory and Its Applications*, pp. 29–53, Birkhäuser, Boston. [Introduces the use of hybrid systems for the modelling of motion control systems.].

Cassandras C.G. (1993). *Discrete Event Systems: Modeling and Performance Analysis*. Irwin Publ. [Discusses several models of interacting continuous and discrete-event dynamics, both deterministic and stochastic frameworks.].

Di Benedetto M.D. (2001). Special issue on hybrid systems in control. *International Journal of Robust and Nonlinear Control* **11**(5).

Evans R.J., Savkin A.V. (1999). Special issue on hybrid control systems. *Systems & Control Letters* **38**(3).

Hybrid Systems: Computation and Control (1998–2003). *Hybrid Systems: Computation and Control*. Lecture Notes in Computer Science, Springer-Verlag. [Proceedings of a major annual workshop on hybrid systems].

Lemmon M.D., He K.X., Markovsky I. (1999). Supervisory hybrid systems. *IEEE Control Systems Magazine* pp. 42–55. [Introductory text to hybrid control systems.].

Morse A.S., Pantelides C.C., Sastry S.S., Schumacher J.M. (1999). Special issue on hybrid systems. *Automatica* **35**(3).

van der Schaft A.J., Schumacher J.M. (1999). *An introduction to hybrid dynamical systems*. Lecture Notes in Control and Information Sciences 251, Springer-Verlag. [Introductory lecture notes in hybrid systems with an extensive discussion on complementarity systems and well-posedness.].