6.43.28.1 Modeling of Hybrid Systems[1]

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

John Lygeros, Dept. of Engineering, University of Cambridge, Cambridge CB2 1PZ, U.K., jl290@eng.cam.ac.uk.

Shankar Sastry, Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, U.S.A., sastry@eecs.berkeley.edu.

Keywords: Hybrid system; Real-time system; Dynamical system; Control theory; Embedded system; Zeno execution; Zeno time; Hierarchical control; Computer-controlled system; Discontinuous control; Switched system; Automaton; Finite state machine; Differential equation; Vehicle control; Automated highway system; PID control; Automatic control; Discrete-event system.

Contents

[1]To appear in the UNESCO Encyclopedia of Life Support Systems

Glossary

Hybrid System: Dynamical system involving interacting continuous and discrete dynamics.
Hybrid Automaton: Mathematical model of a hybrid system.
Execution: Solution (or trajectory) of a hybrid automaton.
Hybrid Time Trajectory: Set of times over which an execution is defined.
Domain: Set of states where continuous evolution is possible.
Guard: Set of states where a discrete transition is possible.
Reset: Possible states after a discrete transition.
Zeno of Elea: Greek philosopher (ca. 500–400 B.C.).
Zeno Execution: Execution that takes an infinite number of discrete transitions in a finite amount of time.

Summary

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. This paper illustrates hybrid dynamics by several simple examples. Some of these examples illustrate properties of hybrid systems not present in purely continuous or purely discrete systems, while others illustrate application domains such as vehicle control and real-time systems. A mathematical model called a hybrid automaton is then introduced, to show how hybrid dynamics can be formally analyzed.

1. Introduction

In the literature, the term "hybrid systems" is used to describe a very wide class of dynamical systems that involve the interaction of heterogeneous data types and dynamics. Of great interest is the class of hybrid systems that arises out of the interaction of continuous dynamics, that describe the evolution of a continuous state under differential or difference equations, with discrete dynamics, that describe the evolution of a finite state under automata or other models of computation. This class of hybrid systems has been the focus of intense research activity in recent years. The reason is that it provides a convenient framework for modelling a wide range of engineering systems. For example, the hybrid framework is ideal for modelling systems with multiple time scales, where the fast dynamics can be abstracted away and be treated as discrete changes affecting the slower dynamics. Examples include mechanical systems with collisions, circuits with diodes and switches, chemical processes controlled by valves or pumps, and, most importantly, embedded computation systems, where digital devices interact with an analogue environment.

Another reason for the popularity of hybrid systems is their importance in applications. Methods and tools developed for hybrid systems have already proved useful in a wide range of technological applications. 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, automotive control, bioengineering, embedded software, process control, highway systems and manufacturing. Though many of the applications are still too complicated to be addressed in their full generality by existing hybrid tools, impressive progress has been recorded in all of these application areas.

The aim of this paper is to highlight the diversity of hybrid phenomena that one encounters in physical and technological systems. In Section 2 a number of examples are presented to illustrate the types of issues that arise out of the discrete–continuous interaction and the types of applications that can be addressed using a hybrid approach. We also discuss the common themes that emerge in the study of these examples. In Section 3 we present a formal mathematical framework, which we call *hybrid automaton*, in which all of these diverse phenomena can be modeled and analysed. Then, in Section 4, we discuss how one can determine whether models developed in the hybrid automaton framework are reasonable representations of physical reality, or whether they contain fundamental flaws. Software tools for modeling hybrid systems are briefly discussed in Section 5.

2. Examples of Hybrid Systems

Hybrid control systems are a much richer class of systems than ordinary control systems. In a hybrid system there is an interaction between continuous and discrete dynamics. The continuous flow is in general influenced not only by the regular 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. In Section 3 we present a mathematical framework that allows one to model a class of hybrid phenomena. First, however, we informally introduce a number of examples, which are chosen to illustrate various characteristics of hybrid dynamics.

2.1. Water Tank System

Consider the two-tank system shown in Figure 1. 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 1.

Suppose that at the initial time $x_1 > r_1$ and $x_2 > r_2$, and that the inflow if directed to Tank 1 (i.e., the discrete state q of the system is equal to q_1). Then the continuous state flows according to the differential equation in the q_1 state in Figure 1. When the condition $x_2 \leq r_2$ specified on the edge, a discrete transition takes place. Subsequently, the state resumes flowing according to the q_2 state and so on. Such a trajectory having one continuous component, x, and one discrete component, q, is called an *execution* (sometimes a *run* or a *solution*) of the hybrid system. An execution of the hybrid system is shown in Figure 2.

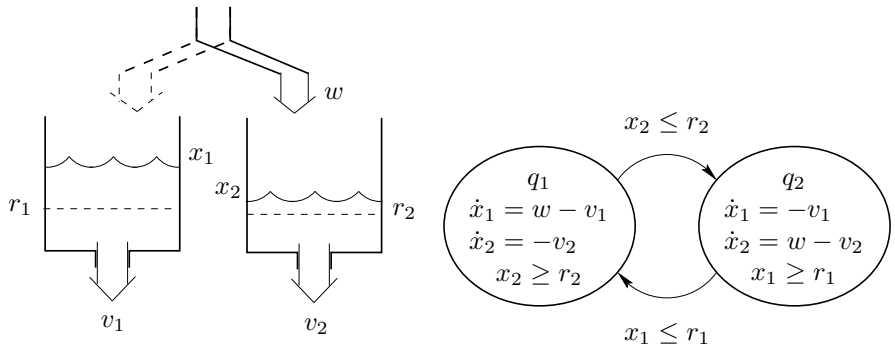

Figure 1: Water tank system and the corresponding hybrid system.

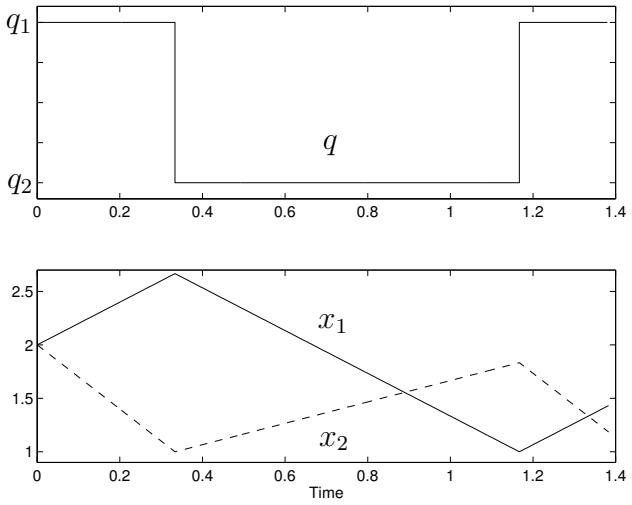

Figure 2: Example of an execution of the water tank hybrid system.

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. This behaviour is known as the Zeno phenomenon and is further discussed in Section 4.

2.2. Bouncing Ball

A model for a bouncing ball can be represented as a simple hybrid system with a continuous state of dimension two $x = (x_1, x_2)$ and a single discrete state (Figure 3). x_1 denotes the vertical position of the ball and x_2 its velocity. The continuous motion of the ball, governed by Newton's laws of motion, is represented by the differential equation in the vertex of the graph, where g denotes the gravitational acceleration. As specified, the equation is only valid as long as $x_1 \geq 0$, i.e., as long as the ball is above the ground. The ball bounces when $x_1 = 0$ and $x_2 \leq 0$, which is detailed by the left expression attached to the edge of the graph ($\wedge$ denotes the logical "and"). At each bounce, the ball loses a fraction of its energy. This is represented

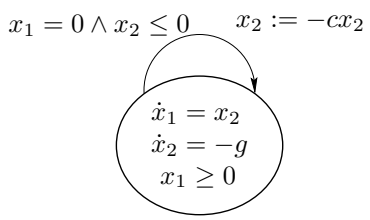

Figure 3: A hybrid system modeling a bouncing ball.

by the equation $x_2 := -cx_2$, where $c^2 \in [0, 1]$ is the coefficient of restitution. (The notation ":=" should be interpreted as if x_2 is reset to the value $-cx_2$ at the transition. The *reset map* is formally defined in Section 3.)

Starting at a point (x_1, x_2) with $x_1 > 0$, the continuous state flows according to the vector field as long as the condition $x_1 \geq 0$ is fulfilled. When $x_1 = 0$ and $x_2 \leq 0$, a discrete transition takes place and the continuous state is reset to $x_2 := -cx_2$ (x_1 remains constant). Subsequently, the state resumes flowing according to the vector field and so on.

For this example, it is easy to see that for $c \in (0, 1)$ there is an accumulation point for the times of the discrete jumps. In other words, the ball bounces infinitely many times in a finite time interval. The bouncing ball hence exhibit Zeno phenomenon, similar to the water tank system. Note however that the continuous state is constant at discrete transitions for the water tank system (the water volumes do not change during the switch of the inflow), while for the bouncing ball system the continuous state makes a jump.

2.3. Clegg Integrator

Many classical control strategies involve mode switching and other discontinuous control actions. Examples include anti-windup schemes, gain scheduling and sliding mode control. One motivation for hybrid control models is to include all these strategies within a single mathematical framework. Here we describe a classical fix in process control, where the state of the integrator in the PID controller [EOLSS,6.43.3.3] is reset whenever its input crosses zero. This so called Clegg Integrator was invented by J. C. Clegg in 1958.

Let e be the input to the Clegg Integrator and x the integrator state. The Clegg Integrator can be described by

$$\dot{x}(t) = e(t) \qquad \text{and} \qquad x(t+) = 0, \quad \text{if } e(t) = 0,$$

where the plus sign indicates that x is set to zero directly after e becomes zero. Figure 4 shows a hybrid model for the same set of equations. The hybrid system has the input e and the output x.

The advantage of using a Clegg Integrator compared to an ordinary integrator is that it gives less phase lag, and thus in many applications improved stability margin. Using the describing function method [EOLSS,6.43.6.1] it is easy to show that the Clegg Integrator gives 38 degrees phase lag, compared to 90 degrees of an ordinary integrator. A disadvantage with the Clegg Integrator is that it may induce oscillations.

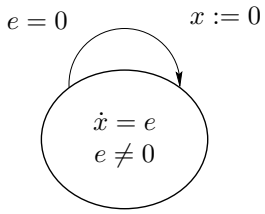

Figure 4: Hybrid system illustrating the Clegg Integrator.

2.4. Thermostat

Consider the control problem of maintaining the temperature of a room at some desired level (say 19 degrees Celsius). Assume that a thermostat is used as a controller, but that we do not have an exact model of how the thermostat functions. It is only known that the thermostat turns on the radiator when the temperature is between 16 and 18 degrees and it turns the radiator off when the temperature is between 20 and 22. This heating system can be modeled as the hybrid system in Figure 5, where x denotes the temperature and the two discrete states correspond to the radiator being off and on.

In this example, there is some uncertainty about when a transition takes place. We know that this will happen when the temperature is in the intervals $[16, 18]$ and $[20, 22]$, but not exactly when. Let us elaborate on how this ambiguity is captured by the hybrid system model. (A formal description is given in Section 3.) Note that there are three components associated with the discrete dynamics: (1) the *domains* $x \geq 16$ and $x \leq 22$, which constrain the values of the continuous state in the corresponding discrete state, (2) the *guard conditions* $x \leq 18$ and $x \geq 20$, which determine when a discrete transition is allowed to happen (is *enabled)*, and (3) the *reset map* $x \mapsto x$, which specifies the relation between the old and the new continuous state when a transition takes place (which in this example is equal to the identity map, but for the bouncing ball, for example, is $(x_1, x_2) \mapsto (x_1, -cx_2)$). The interpretation is as follows: as long as the continuous state x belongs to a domain, continuous evolution *may* continue (the temperature may continue to increase/decrease according to the differential equation). When x enters a guard, a discrete jump *may* take place (the radiator may be switched on/off). For the thermostat system this means that if, for example, the state is $(q, x) = (\texttt{no heating}, 19)$ then continuous evolution may continue. If the state is $(q, x) = (\texttt{no heating}, 17)$ either continuous evolution can continue, or a discrete jump to state $(q, x) = (\texttt{heating}, 17)$ can take place. Finally, if the state is $(q, x) = (\texttt{no heating}, 16)$, a discrete jump must must take place, because continuous evolution would lead x outside the domain.

The thermostat hybrid system is *non-deterministic*, in the sense that for a given initial condition it accepts a whole family of different executions. A formal definition of a hybrid systems and its evolution is given in Section 3, and determinism is discussed in Section 4.

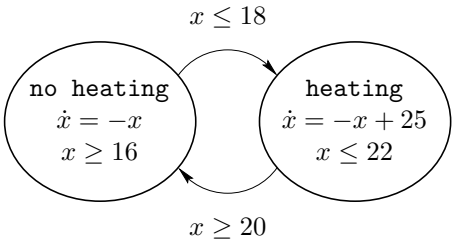

Figure 5: Hybrid system modeling a thermostat and the heating of a room.

2.5. Gear Shift Control

The gear shift example describes a control design problem where both the continuous and the discrete controls need to be determined. Figure 6 shows a model of a car with a gear box having four gears. 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 two control signals: the gear denoted $\texttt{gear} \in \{1, \ldots, 4\}$ and the throttle position denoted $u \in [u_{\min}, u_{\max}]$. These may both be considered as inputs to the system, while the position and the velocity are outputs. The gear shift is necessary because little power can be generated by the engine at very low or very high engine speed. The function α_i represents the efficiency of gear i.

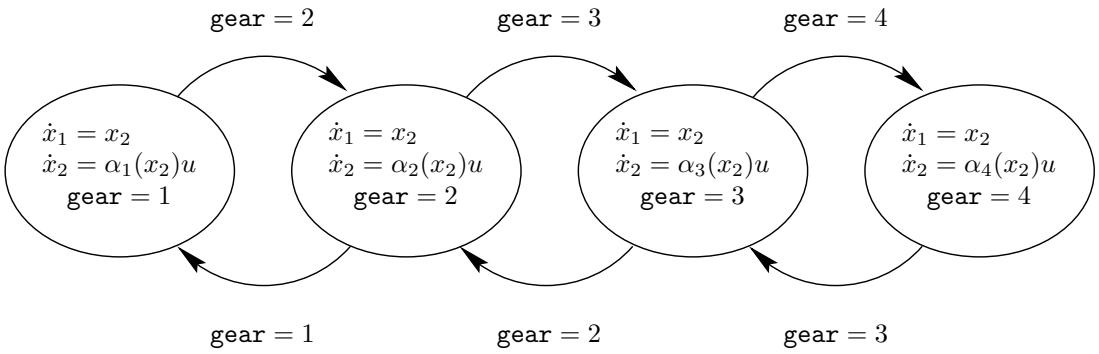

Figure 6: A hybrid system modeling a car with four gears.

Several interesting control problems can be posed for this simple car model, including the following: What is the optimal control 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. The optimal controller, which can be modeled as a hybrid system, may be derived using the theory of optimal control of hybrid systems [EOLSS,6.43.28.5].

2.6. Swing-Up of Inverted Pendulum

The inverted pendulum (Figure 7) is a popular system in teaching laboratories for illustrating control problems present in various applications such as thrust-vectored rocket control and bipedal walking. Here we discuss a hybrid control strategy for swinging up a pendulum from a downward to an upright position and then stabilizing it.

Figure 7: Pendulum on a cart.

Let x_1 be the angle between the vertical and the pendulum, x_2 the angular velocity and $u \in [-u_{\max}, u_{\max}]$ the acceleration of the pivot, which we consider as the control signal. Then, the dynamics after normalization is given by

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= g \sin x_1 - u \cos x_1, \end{aligned}$$

where $g > 0$ is the acceleration of gravity. A swing-up control strategy suggested by Åström and Furuta is based on controlling the energy of the pendulum. It has three different modes as shown in Figure 8. The modes `pos max` and `neg max` correspond to maximum acceleration in the positive and negative directions of the cart, respectively. The switching between these two modes are determined by the sign of

$$\beta(x_1, x_2) = [x_2^2/2 + g(\cos x_1 - 1)]x_2 \cos x_1.$$

To avoid chattering when the the pendulum is close to upright position (within a fixed angle $\theta > 0$), the system enters mode `loc stab`, where a locally stabilizing linear controller $u = \gamma_1 x_1 + \gamma_2 x_2$ is applied. Note that, because the acceleration of the pivot is limited to $u_{\max}$, more than one swing may be needed to swing-up the pendulum.

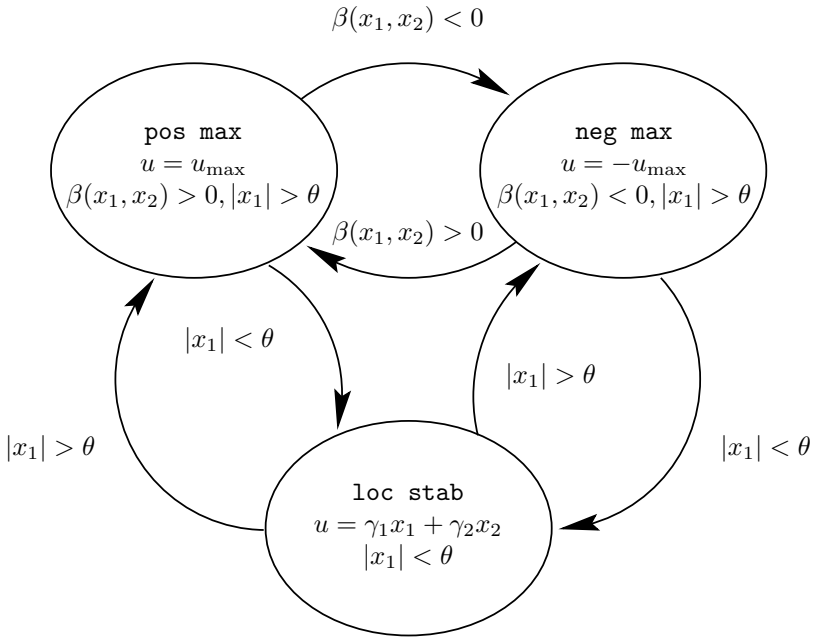

Figure 8: Hybrid control strategy to swing up and stabilize an inverted pendulum on a cart.

2.7. Computer-Controlled System

Hybrid systems are natural models for computer-controlled systems (Figure 9), since they involve a physical process (which often can be modelled as continuous-time system) and a computer (which is fundamentally a finite state machine). The classical approach to computer-controlled systems has been using sampled-data theory [EOLSS,6.43.4.1], where it is assumed that measurements and control actions are taken at a fixed sampling rate. Such a scheme is easily encoded using a hybrid model. The hybrid model also captures a more general formulation, where measurements are taken based on computer interrupts. This is sometimes closer to real-time implementations [EOLSS,6.43.4.4], for example, in embedded control systems.

2.8. Automated Highway System

Highway congestion is an increasing problem, especially in and around urban areas. One of the promising solutions considered for this problem is traffic automation, either partial or full.

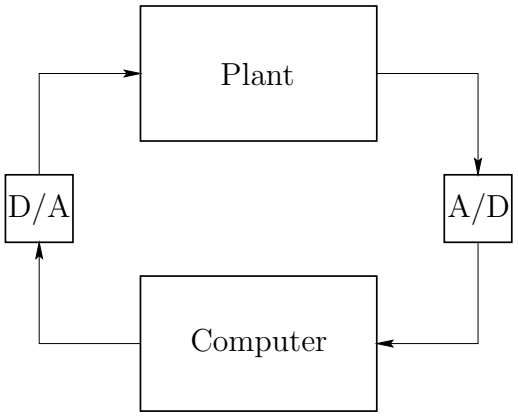

Figure 9: Computer-controlled system.

Substituting the driver by an automatic controller may reduce or eliminate human errors and hence improve safety. Moreover, as the automatic controller can react to disturbances faster than a human driver, automation may also decrease the average inter-vehicle spacing and hence increase throughput and reduce congestion and delays.

The design of an automated highway system (AHS) is an extremely challenging control problem, and a number of alternatives have been proposed for addressing it. One of the most forward-looking AHS designs involves a fully automated highway system that supports platooning of vehicles. The platooning concept of Varaiya assumes that traffic on the highway is organized in groups of tightly spaced vehicles (platoons). The first vehicle of a platoon is called the leader, while the remaining vehicles are called followers. The platooning structure achieves a balance between safety and throughput: it is assumed that the system is safe even if in emergency situations (for example, as a result of a failure) collisions do occur, as long as the relative velocity at impact is low. Of course no collisions should take place during normal operation. This gives rise to two safe spacing policies. The obvious one is that of the leaders, who are assumed to maintain a large inter-platoon spacing (of the order of 30–60 meters). The idea is that the leader has enough time to stop without colliding with the last vehicle of the platoon ahead. The spacing policy of the followers is not as intuitive, because they are assumed to maintain tight intra-platoon spacing (of the order of 1–5 meters). In case of emergency, collisions among the followers of a platoon may take place, but, because of the tight spacing, they are expected to be at low relative velocities. Recent theoretical, numerical and experimental studies have shown that an AHS that supports platooning is not only technologically feasible but, if designed properly, may lead to an improvement of both the safety and the throughput of the highway system, under normal operation.

Implementation of the platooning concept requires automatic vehicle control, since human drivers are not fast and reliable enough to produce the necessary inputs. To manage the complexity of the design process a hierarchical controller was proposed by Varaiya. The controller is organized in four layers, see Figure 10. The top two layers, called network and link, reside on the roadside and are primarily concerned with throughput maximization, while the bottom two, called coordination and regulation, reside on the vehicles and are primarily concerned with safety. The physical layer is not part of the controller. It contains the plant, i.e., the vehicles and highway, with their sensors, actuators and communication equipment.

The network layer is responsible for the flow of traffic on the entire highway system, for exam-

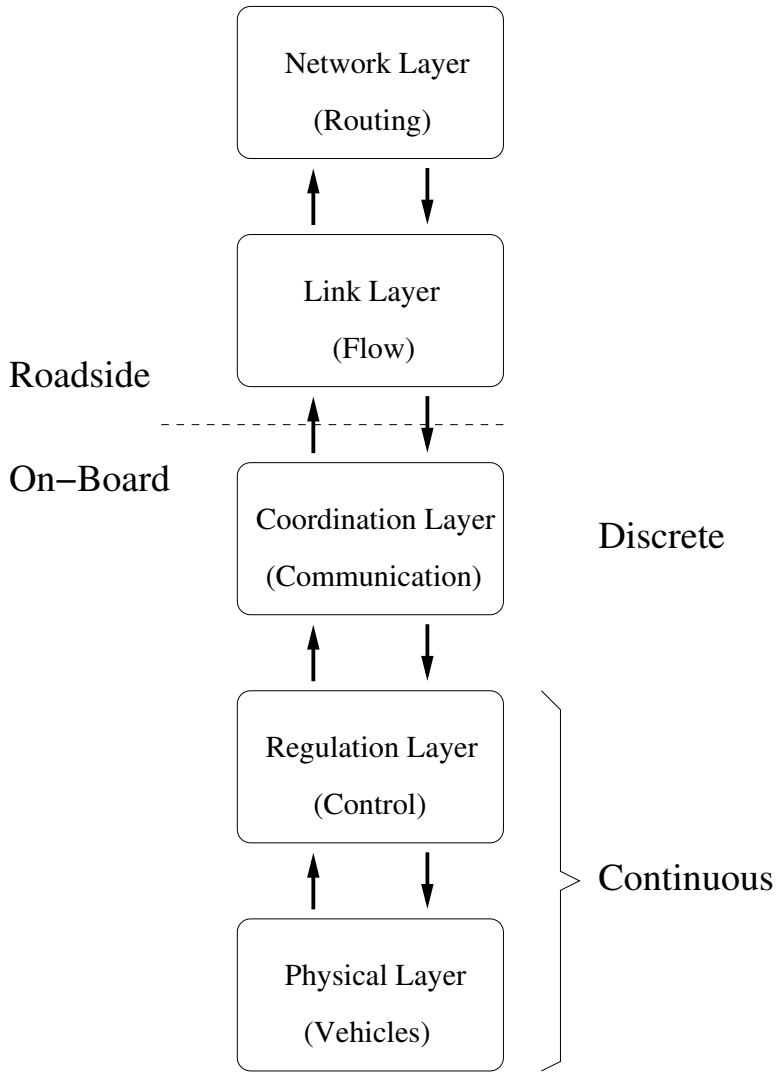

Figure 10: Hierarchical control structure of an automated highway system.

ple, several highways around an urban area. Its task is to prevent congestion and maximize throughput by dynamically routing traffic. The link layer coordinates the operation of sections (links) of the highway (for example the highway segment between two exists). Its primary concern is to maximize the throughput of the link. With these criteria in mind, it calculates an optimum platoon size and an optimum velocity and decides which lanes the vehicles should follow. It also monitors incidents and diverts traffic away from them, in an attempt to minimize their impact on traffic flow.

The coordination layer coordinates the operation of neighbouring platoons by choosing manoeuvres that the platoons need to carry out. For normal operation, these manoeuvres are *join* to join two platoons into one, *split* to break up one platoon into two, *lane change*, *entry* and *exit*. The coordination layer is primarily a discrete controller. It uses communication protocols, in the form of finite state machines, to coordinate the execution of these manoeuvres between neighbouring vehicles.

The regulation layer receives the coordination layer commands and readings from the vehicle sensors and generates throttle, steering and braking commands for the vehicle actuators. For this purpose it utilizes a number of continuous time feedback control laws that use the readings

provided by the sensors to calculate the actuator inputs required for a particular manoeuvre. In addition to the control laws needed for the manoeuvres, the regulation layer makes use of two *default* controllers, one for leader and one for follower operation.

The interaction between the coordination layer (which is primarily discrete) and the regulation layer (which is primarily continuous) gives rise to interesting hybrid dynamics. To ensure the safety of the AHS, one needs to verify that the closed loop hybrid system does not enter a bad region of its state space (e.g., does not allow any two vehicles to collide at high relative velocity). This issue can be addressed by posing the problem as a game between the control applied by one vehicle and the disturbance generated by neighbouring vehicles. It can be shown that information available through discrete coordination can be used together with the continuous controllers to ensure the safety of the closed-loop hybrid system.

3. Mathematical Models for Hybrid Systems

Hybrid systems were informally described in the previous section as systems with both continuous-time and discrete-event dynamics. In this section, a formal mathematical model is introduced.

3.1. Modelling Issues

To be able to deal effectively with all the types of hybrid behaviour demonstrated by the examples of Section 2 one needs a modelling language that is

- *descriptive*, to allow one to capture different types of continuous and discrete dynamics, be capable of modelling different ways in which discrete evolution affects and is affected by continuous evolution, allow non-deterministic models to capture uncertainty, etc.
- *composable*, to allow one to build large models by composing models of simple components.
- *abstractable*, to allow one to decompose the design problem for composite models down to design problems for individual components and, conversely, compose arguments about the performance of individual components to study the performance of the overall system.

Modelling languages that possess subsets of these properties have been proposed in the literature. Different languages place more emphasis on different aspects, depending on the applications and problems they are designed to address. Unfortunately, powerful modelling languages will inevitably allow one to produce models that are unreasonable either physically or mathematically. This issue is examined in Section 4.

In this section we introduce a specific hybrid system modelling language, that we call *hybrid automata*. Hybrid automata provide a mathematically concrete language, that is rich enough to demonstrate a number of interesting issues, but also simple enough to do so without excessive mathematical formalism and notation. We stress that hybrid automata capture a fraction of the hybrid phenomena that have been studied in the literature. Some of the generalisations that have been pursued include the following.

1. *Generalisations of the notion of time.* Executions of hybrid automata are defined over sequences of intervals of the real line. The time sets we consider exclude, among other things, the possibility of left accumulation points of discrete transitions, see [EOLSS,6.43.28.2]. More general models of time that have been considered in the literature (such as enriched time intervals) allow one to capture more phenomena.

2. *Generalisations of the continuous dynamics.* Hybrid automata assume that the evolution of continuous states in continuous time is governed by a differential equation. Generalisations include systems modeled by differential inclusions, differential algebraic systems, or, more generally, abstract sets of time functions equipped only with a few fundamental axioms (such as concatenation and the semigroup property). The last option allows one to also capture infinite dimensional continuous dynamics that include, for example, pure time delays.

3. *Inclusion of inputs and outputs.* To keep the presentation simple, only autonomous hybrid systems (with time-invariant dynamics and no inputs or outputs) are considered here. One can think of such systems as closed-loop hybrid control systems. Generalisations that include inputs and outputs and hence allow one to study composition and abstraction properties have also been extensively studied in the literature.

3.2. Hybrid Automata

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. We start by giving the *syntax* of this modelling language using both a mathematical and a graphical representation.

Definition 3.1 (Hybrid Automaton) *A hybrid automaton H is a collection $H = (Q, X, f, \mathrm{Init}, D, E, G, R)$, where*

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

We refer to $(q, x) \in Q \times X$ 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.

[2]The domain is sometimes called the invariant set, especially in the hybrid system literature in computer science. $P(X)$ denotes the power set (set of all subsets) of X.

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 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.

As we saw in the previous section, it is convenient to visualize hybrid automata as directed graphs (Q, E) with vertices Q and edges E. With each vertex $q \in Q$, we associate a set of initial states $\text{Init}_q = \{x \in X \ : \ (q, x) \in \text{Init}\}$, 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 $R(e, \cdot)$. We illustrate the notation by specifying the hybrid automaton for the water tank system in Figure 1:

- $Q = \{q\}$ and $Q = \{q_1, q_2\}$;
- $X = \{x_1, x_2\}$ and $X = \mathbb{R}^2$;
- $f(q_1, x) = (w - v_1, -v_2)$ and $f(q_2, x) = (-v_1, w - v_2)$;
- $\text{Init} = Q \times \{x \in \mathbb{R}^2 : x_1 \geq r_1 \wedge x_2 \geq r_2\}$;
- $D(q_1) = \{x \in \mathbb{R}^2 : \ x_2 \geq r_2\}$ and $D(q_2) = \{x \in \mathbb{R}^2 : \ x_1 \geq r_1\}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbb{R}^2 : x_2 \leq r_2\}$ and $G(q_2, q_1) = \{x \in \mathbb{R}^2 : x_1 \leq r_1\}$;
- $R(q_1, q_2, x) = R(q_2, q_1, x) = \{x\}$.

The graphs can be treated as formal definitions of hybrid automata, since they represent the same information as models in the style of Definition 3.1.

3.3. Executions

To formally define the *semantics* of a hybrid automaton model (i.e., the types of dynamics it encodes, or else the types of solutions that it accepts) we first need to consider the sets of times over which these solutions will be defined.

Definition 3.2 (Hybrid Time Set) *A hybrid time set is a finite or infinite sequence of intervals $\tau = \{I_i\}_{i=0}^{N}$, such that*

- *$I_i = [\tau_i, \tau_i']$ for all $i < N$;*
- *if $N < \infty$ then either $I_N = [\tau_N, \tau_N']$ or $I_N = [\tau_N, \tau_N')$; and*
- *$\tau_i \leq \tau_i' = \tau_{i+1}$ for all i.*

Note that the right endpoint of one interval coincides with the left endpoint of the following interval. The interpretation is that these are the times at which discrete transitions take place. Note also that it is possible to have $\tau_i = \tau_i'$, therefore multiple discrete transitions may take place at the same time. Hence, both purely discrete and purely continuous systems fit within

the hybrid automaton model. Since all hybrid automata discussed here are time invariant we assume that $\tau_0 = 0$ without loss of generality. Hybrid time trajectories can extend to infinity if τ is an infinite sequence or if it is a finite sequence ending with an interval of the form $[\tau_N, \infty)$.

Definition 3.3 (Execution) *An execution of a hybrid automaton H is a collection $\chi = (\tau, q, x)$, where $\tau = \{I_i\}_{i=0}^N$ is a hybrid time set, $q = \{q_i(\cdot)\}_{i=0}^N$ is a sequence of functions $q_i(\cdot) : I_i \to Q$ and $x = \{x_i\}_{i=0}^N$ is a sequence of continuously differentiable functions $x_i : I_i \to X$, such that*

- $(q_(0), x_0(0)) \in \text{Init}$*;*
- *for all i and all $t \in [\tau_i, \tau_i')$, $q_i(t) = q_i(\tau_i)$, $\dot{x}_i(t) = f\big(q_i(t), x_i(t)\big)$ and $x_i(t) \in D(q_i(t))$; and*
- *for all $i < N$, $e = (q_i(\tau_i'), q_{i+1}(\tau_{i+1})) \in E$, $x_i(\tau_i') \in G(e)$, and $x_{i+1}(\tau_{i+1}) \in R(e, x_i(\tau_i'))$.*

The definition of an execution involves conditions on the initial state, the continuous evolution and the discrete evolution. As an example, consider again the execution of the water tank automaton shown in Figure 2. The hybrid time set τ consists in this case of three intervals. The discrete state evolution, $q = \{q_1(\cdot), q_2(\cdot), q_3(\cdot)\}$, is shown in the upper plot in Figure 2, and the continuous state evolution $x = \{x_1(\cdot), x_2(\cdot), x_3(\cdot)\}$ is shown in the lower plot.

We say that a hybrid automaton H *accepts* an execution χ if χ fulfills the conditions of Definition 3.3. For an execution $\chi = (\tau, q, x)$, we use $(q_0, x_0) = \big(q_0(\tau_0), x_0(\tau_0)\big)$ to denote the initial state. The *execution time* $\mathcal{T}(\chi)$ is defined as

$$\mathcal{T}(\chi) = \sum_{i=0}^{N} (\tau_i' - \tau_i) = \lim_{i \to N} \tau_i' - \tau_0.$$

An execution is called infinite if τ is either an infinite sequence or if $\mathcal{T}(\chi) = \infty$.

The concept of a *reachable state* is fundamental in the study of hybrid systems. A state, (q, x), of a hybrid automaton, H, is called reachable if starting at some initial state, $(q_0, x_0) \in \text{Init}$, H can end up in (q, x). The set of states reachable by H is denoted Reach_H.

The existence and uniqueness conditions developed in the next section also involve the set of states from which continuous evolution is impossible. Let $\psi(t, q, x)$ denote the flow of the vector field $f(q, \cdot)$. Then the set of states from which continuous evolution is impossible is given by

$$\text{Out}_H = \{(q, x) \in Q \times X : \forall \epsilon > 0,\ \exists t \in [0, \epsilon),\ \psi(t, q, x) \notin D(q)\}.$$

These are hence the states for which the continuous dynamics instantaneously force the state outside the domain.

4. Properties of Hybrid Systems

When analyzing hybrid automata from a control theoretic perspective it is natural to ask fundamental questions, such as "does the system accept an infinite execution for every initial state?", "do the executions depend continuously on the initial state?", "is the hybrid system

stable?" (in some appropriate sense) etc. Here we only touch upon these questions by discussing existence and uniqueness issues. Stability issues are discussed in [EOLSS,6.43.28.3] and [EOLSS,6.43.28.7]. Analysis questions one may be interested in from a computer science point of view are discussed in [EOLSS,6.43.28.4].

4.1. Overview of Issues

A common danger in hybrid modelling is lack of existence of solutions. In most of the hybrid languages one can easily construct models that admit no solutions for certain initial states (*blocking* hybrid systems). This is an undesirable property when modelling physical systems, since it suggests that the mathematical model provides an incomplete picture of the physical reality; the evolution of the physical system is likely to continue despite the fact that the evolution of the mathematical model is undefined.

Even if a hybrid system accepts executions for all initial states, it does not necessarily accept executions with infinite execution times. Hybrid systems can take an infinite number of discrete transitions in finite time (*Zeno* phenomenon). One can argue that physical systems do not exhibit Zeno behaviour. However, modelling abstraction can lead to Zeno hybrid models of physical systems that look realistic. Since abstraction is crucial for handling complex systems, understanding when it may lead to Zeno behaviour is important.

Another issue that arises in hybrid modelling is lack of uniqueness of solutions (*non-determinism*). For hybrid systems it is often desirable to retain some level of non-determinism, since it allows one to capture modelling uncertainty. This, however, requires additional care when designing controllers for such systems, or when developing arguments about their performance. One needs to adopt a style of reasoning similar to the one used in robust control to ensure that a controller is robust with respect to model uncertainty. Instead of developing arguments about *the solution* of the system, one needs to ensure that the argument holds for *all solutions* of the system.

Hybrid systems are especially challenging from the point of view of computer simulation. The problems faced by the developers of simulation algorithms are intimately related to the modelling problems discussed so far. The overall goal of a simulation is to present an accurate approximation of the behaviour of the real system. Due to interaction between different components and computational models in a hybrid system, computer simulations may however produce results that are difficult to interpret. It is therefore important that the user is aware of the following possible obstacles.

- *Stiffness*: Lack of continuity of the solution with respect to initial conditions, an inherent characteristic of hybrid systems, can lead to problems, both theoretical and practical. The most common problem is event detection, i.e., to determine when guard crossing takes place.
- *Existence and uniqueness*: Simulation algorithms may run into trouble if the simulated model does not have a solution defined in a suitable sense. This may be the case, for example, if the hybrid system exhibits chattering or Zeno solutions. It is fairly easy to construct hybrid systems that, when simulated "naively", produce the wrong results either mathematically or based on physical intuition. In certain cases the problem may be

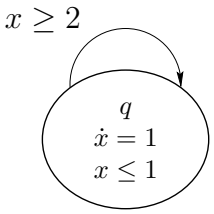

Figure 11: Blocking hybrid system.

alleviated by introducing additional discrete states and generalised solutions such as Filippov solutions. Non-determinism introduces further complications. Here the simulation algorithm may be called upon to decide between different alternatives. When a choice between continuous evolution and discrete transition is possible, a common approach is to take transitions the moment they are enabled (*as-soon-as* semantics).

- *Composability*: When simulating large scale, multi-agent systems (like the AHS), one would like to be able to build up the simulation from a number of components. It may also be desirable to be able to dynamically add components (e.g., to model vehicles joining the AHS), eliminate components (e.g., to model vehicles leaving the AHS), or redefine the interactions of components (e.g., to model vehicles changing lanes). Object-oriented modelling languages have been developed to address these needs.

Simulation packages that have been developed to address these issues include SHIFT, Dymola (based on the Modelica standard) and the Simulink tool StateFlow. These and other software tools are further discussed in Section 5.

4.2. Existence of Executions

It is easy to construct hybrid automata that do not accept infinite executions. Consider, for example, the hybrid automaton of Figure 11. If the system finds itself at state $(q, 1)$ the continuous state is forced to leave $D(q)$, but the discrete transition is not enabled since $x = 1$ does not belong to the guard. We say that the execution blocks at $(q, 1)$.

A hybrid automaton H is called *non-blocking* if it accepts infinite executions for all $(q_0, x_0) \in$ Init. It is reasonable to expect that models of real systems will be non-blocking. Therefore it is useful to determine when there exist infinite executions for a hybrid automaton. The above example suggests that a hybrid automaton H is non-blocking if for all reachable states for which continuous evolution is impossible a discrete transition is possible. More formally, one can show that H is non-blocking if for all $(q, x) \in \text{Reach}_H \cap \text{Out}_H$, there exists $(q, q') \in E$ such that $x \in G(q, q')$.

4.3. Uniqueness of Executions

A hybrid automaton may accept multiple infinite executions for a single initial state. To see this, take the blocking example above but let $D(q) = \{x \in \mathbb{R} : \ x \leq 3\}$. Then, at $(q, x) = (q, 2)$ the guard is enabled, so a discrete transition may take place. However, $x = 2$ is in the interior of $D(q)$, so a continuous evolution is also possible. Actually, there is an infinite number of

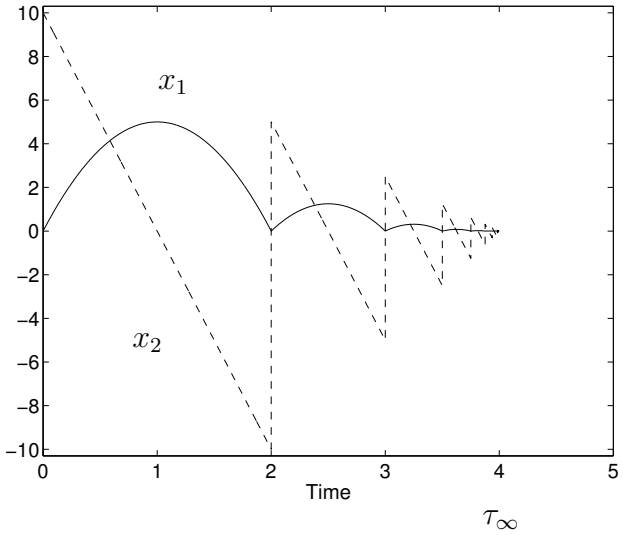

Figure 12: An example of a Zeno execution for the bouncing ball example.

maximal executions starting in $(q, 0)$; all of them have one discrete transition taking place somewhere between $2 \leq \tau_0' \leq 3$. The thermostat system presented in Section 2.4 is another example of a non-deterministic hybrid system. The hybrid model captures the uncertainty about the temperature at which the radiator is switched on and off.

A hybrid automaton H is called *deterministic* if it accepts at most one execution (which is not a strict prefix of any other execution) for all $(q_0, x_0) \in \text{Init}$. Intuitively, a hybrid automaton may be non-deterministic if either there is a choice between continuous evolution and a discrete transition, or if a discrete transition can lead to multiple destinations. More formally, it can be shown that H is deterministic if and only if for all $(q, x) \in \text{Reach}_H$,

(1) if $x \in G(q, q')$ for some $(q, q') \in E$ then $(q, x) \in \text{Out}_H$;

(2) if $(q, q') \in E$ and $(q, q'') \in E$ with $q' \neq q''$ then $G(q, q') \cap G(q, q'') = \emptyset$; and

(3) if $(q, q') \in E$ and $x \in G(q, q')$ then $|R(q, q', x)| \leq 1$.

Here, $|\cdot|$ denotes the cardinality (number of elements) of a set.

4.4. Zeno Executions

An execution of a hybrid system may exhibit infinitely many discrete jumps in finite time. This is a truly hybrid phenomenon, in the sense that it requires the interaction between continuous and discrete behaviour. It can not even be formulated for a purely discrete system without the notion of continuous time.

Formally, an execution $\chi = (\tau, q, x)$ is called *Zeno* if it is infinite but its execution time is finite, i.e., if τ is an infinite sequence and $\mathcal{T}(\chi) < \infty$. The execution time of a Zeno execution is called the *Zeno time*. An example of a Zeno execution for the bouncing ball system with $c \in (0, 1)$ is shown in Figure 12. 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 to support his view that the

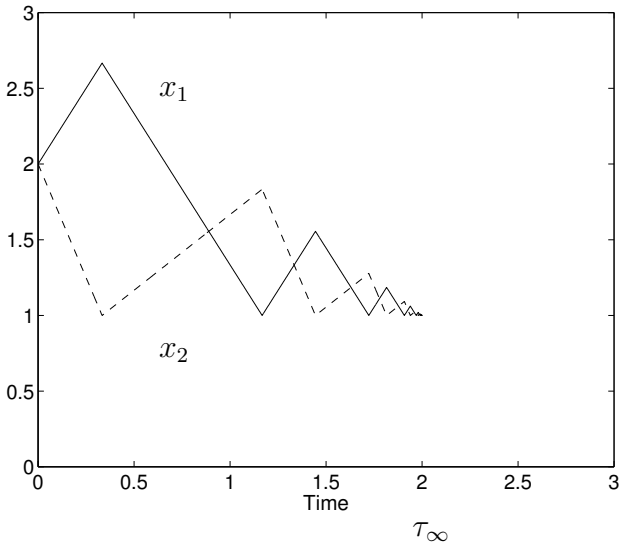

Figure 13: A Zeno execution for the water tank hybrid system.

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.

Zeno phenomena are classical in control, but often under other names. For example, consider the problem of minimizing the performance index $\int_0^\infty x^p(t)\,dt$, $p > 1$, with respect to the dynamics $\ddot{x} = u$, $x(0) \neq (0,0)$, and the control constraint $|u(t)| \leq 1$. The solutions to this problem exhibit infinitely many switchings in finite time as proved by Zelikin and Borisov, and thus corresponds to a Zeno execution. In optimal control, this is referred to as Fuller's phenomenon.

To better understand the dynamics of Zeno hybrid systems, we need to extend some notions from continuous dynamical systems to the hybrid domain. An ω limit point of a Zeno execution is called a Zeno point (or a Zeno state). A Zeno execution for the water tank system with $\max(v_1, v_2) < w < v_1 + v_2$ is shown in Figure 13. The continuous part of its Zeno point is equal to (r_1, r_2). For the bouncing ball execution in Figure 12 the continuous part of the Zeno point is the origin. Note that in both Zeno examples, the continuous part of the Zeno point is a single point. This is no coincidence. It is possible to show in general that if the reset map R is either contracting on G (as for the bouncing ball) or the identity on G (as for the water tank), the continuous part of the Zeno point is a singleton. Moreover, if the guard does not intersect the corresponding interior of the domain, then the Zeno point must be on the boundary of the domains. This is the case for both the bouncing ball and the water tank. The proof of these results and further discussion on Zeno hybrid systems are given in the references.

5. Software Tools

The development of computer-aided design tools for hybrid systems is important. Extensive research efforts have been investigated in several software packages over the last ten–fifteen years, in order to apply hybrid methods to real complex systems. It is possible to classify many of them into simulation tools and verification tools, where simulation here means numerical integration of the hybrid system dynamics and verification is to show that the hybrid system

fulfills a specific safety constraint. The descriptiveness of the underlying modeling language is crucial in both simulation and verification. Today it is possible to simulate fairly general hybrid systems, while verification is limited to certain classes of hybrid systems (such as timed automata).

Numerical computer simulation is important in the analysis and design of hybrid control systems, particularly since the complexity of many systems limit the application of analytical methods. Hybrid systems are in general difficult to simulate due to the nonsmooth characteristics of the state evolution, where the continuous evolution of the system trajectory is affected by discrete jumps. Specific classes of numerical solvers are used to get efficient and accurate results. Recent software tools for hybrid systems include ABACUSS, 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.

Verification is the formal process of analyzing whether a system satisfies a desired specification using a computer algorithm. Following early work on the verification of digital circuits, the hybrid formalism and tools have been subsequently extended to the verification of embedded control software and other applications. Today there exist several software tools, such as d/dt, HyTech, KRONOS and UPPAAL for the verification of various classes of hybrid systems. This is however still a very active area of research, since many real applications are still too complicated to be addressed in their full generality by existing tools. To enlarge the applicability of verification, new techniques are being developed that numerically approximate the set of reachable states for the hybrid system. Verification is further discussed in [EOLSS,6.43.28.4] and [EOLSS,6.43.28.6].

Acknowledgements

Part of this paper is based on work done by by the authors together with Magnus Egerstedt, Slobodan Simić and Jun Zhang. Their contribution is gratefully acknowledged. The work by the first author was partially supported by the European Commission under the RECSYS project.

Bibliography

Alur R., Courcoubetis C., Halbwachs N., Henzinger T.A., Ho P.H., Nicollin X., Olivero A., Sifakis J., Yovine S. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science* **138**(1), 3–34. [Presents a framework for formal specification and algorithmic analysis of hybrid systems. Particular focus on systems for which the continuous evolution follows piecewise-linear trajectories.].

Alur R., Henzinger T.A. (1997). Modularity for timed and hybrid systems. In A. Mazurkiewics, J. Winkowski, eds., *CONCUR 97: Concurrency Theory*, vol. 1243 of *Lecture Notes in Computer Science*, pp. 74–88, Springer-Verlag. [Discusses modular and Zeno hybrid systems from a computer science perspective. Uses the tank example in Section 2.4.].

Antsaklis, P.J. (Editor) (2000). Special issue on hybrid systems: Theory and applications. *IEEE Proceedings* **88**(7). [Collection of some recent work on hybrid systems.].

Åström K.J., Furuta K. (2000). Swinging up a pendulum by energy control. *Automatica* **36**, 287–295. [Hybrid control strategy for an unstable laboratory process, see Section 2.6.].

Aubin J.P., Lygeros J., Quincampoix M., Sastry S., Seube N. (2002). Impulse differential inclusions: A viability approach to hybrid systems. *IEEE Transactions on Automatic Control* **47**(1), 2–20.

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

Bemporad A., Morari M. (1999). Control of systems integrating logic dynamics and constraints. *Automatica* **35**(3), 407–427.

Branicky M.S., Borkar V.S., Mitter S.K. (1998). A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control* **43**(1), 31–45.

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. Poses the problem discussed in Section 2.5.].

Chutinam A., Krogh B. (1999). Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In F.W. Vaandrager, J.H. van Schuppen, eds., *Hybrid Systems: Computation and Control*, no. 1569 in LNCS, pp. 76–90, Springer Verlag.

Clegg J.C. (1958). A nonlinear integrator for servomechanisms. *Transactions of AIEE, Part II* **77**, 41–42. [Presents a modification of the integrator in PID control, which is known as the Clegg Integrator, see Section 2.2.].

Deshpande A., Gollu A., Semenzato L. (1998). The SHIFT programming language for dynamic networks of hybrid automata. *IEEE Transactions on Automatic Control* **43**(4), 584–587.

Engell S., Kowalewski S., Schulz C., Stursberg O. (2000). Continuous-discrete interactions in chemical processing plants. *IEEE Proceedings* **88**(7), 1050–1068. [Hybrid systems aspects of chemical process control, including modeling, validation and scheduling.].

Filippov A.F. (1988). *Differential Equations with Discontinuous Righthand Sides*. Kluwer Academic Publishers. [Classical text on non-smooth dynamical systems.].

Fuller A.T. (1960). Relay control systems optimized for various performance criteria. In *First World Congress IFAC*, Moscow. [Discusses an optimal control problem that leads to infinitely many controller switchings in finite time.].

Hedlund S., Rantzer A. (1999). Optimal control of hybrid systems. In *Proc. 38th IEEE Conference on Decision and Control*, Phoenix, AZ. [Presents a computational solution to the problem discussed in Section 2.5.].

Heemels M. (1999). *Linear Complementarity Systems*. Ph.D. thesis, University of Eindhoven. [Discusses a class of hybrid systems with linear continuous dynamics in each discrete state.].

Johansson K.H., Egerstedt M., Lygeros J., Sastry S. (1999). On the regularization of Zeno hybrid automata. *System & Control Letters* **38**, 141–150. [Discusses Zeno hybrid systems from a control perspective. The examples in Sections 2.1 and 2.4 are analysed.].

Lemmon M. (2000). On the existence of solutions to controlled hybrid automata. In B. Krogh, N. Lynch, eds., *Hybrid Systems: Computation and Control*, vol. 1790 of *Lecture Notes in Computer Science*, Springer-Verlag. [Studies existence of executions of hybrid systems having linear continuous dynamics in each discrete state.].

Lygeros J., Godbole D.N., Sastry S. (1998). A verified hybrid controller for automated vehicles. *IEEE Transactions on Automatic Control* **43**(4), 522–539. [Application of hybrid systems to automated highway systems.].

Lygeros J., Johansson K.H., Simić S.N., Zhang J., Sastry S. (2003). Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control* **48**(1), 2–17.

Lynch N., Segala R., Vaandrager F., Weinberg H. (1996). Hybrid I/O automata. In *Hybrid Systems III*, no. 1066 in LNCS, pp. 496–510, Springer Verlag.

Mattsson S.E., Andersson M., Åström K.J. (1993). Object-oriented modelling and simulation. In D.A. Linkens, ed., *CAD for Control Systems*, chap. 2, pp. 31–69, New York: Marcel Dekker Inc. [Discusses an object-oriented modelling paradigm for multi-domain systems.].

Mattsson S.E., Otter M., Elmqvist H. (1999). Modelica hybrid modeling and efficient simulation. In *IEEE Conference on Decision and Control*, Phoenix, AZ. [Focus on challenges in simulating hybrid systems. Illustrations are done in Modelica, an object-oriented language for modelling of large heterogeneous physical systems.].

Nerode A., Kohn W. (1993). Models for hybrid systems: Automata, topologies, stability. In *Hybrid Systems*, no. 736 in LNCS, pp. 317–356, Springer Verlag.

Pepyne D.L., Cassandras C.G. (2000). Optimal control of hybrid systems in manufacturing. *IEEE Proceedings* **88**(7), 1108–1123. [Application of hybrid systems to manufacturing processes.].

Tavernini L. (1987). Differential automata and their discrete simulators. *Nonlinear Analysis, Theory, Methods & Applications* **11**(6), 665–683. [Discusses well-posedness of a class of hybrid systems called differential automata.].

Tomlin C., Pappas G., Sastry S. (1998). Conflict resolution for air traffic management: a case study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control* **43**(4), 509–521. [Air traffic management posed as a hybrid control problem.].

Utkin V.I. (1992). *Sliding Modes in Control Optimization*. Springer-Verlag, Berlin. [Presents control design method based on discontinuous control.].

van der Schaft A.J., Schumacher J.M. (1998). Complementarity modeling of hybrid systems. *IEEE Transactions on Automatic Control* **43**(4), 483–490. [A class of hybrid systems with certain type of discrete transitions, which is, for example, suitable for modeling electrical circuits with ideal diodes.].

— (2001). Compositionality issues in discrete, continuous, and hybrid systems. *International Journal of Robust and Nonlinear Control* **11**(5), 417–434. [Discusses the importance of composition in the analysis of complex systems. Examples are taken from continuous-time, discrete-event and hybrid systems.].

Varaiya P. (1993). Smart cars on smart roads: Problems of control. *IEEE Transactions on Automatic Control* **38**(2), 195–207. [Application of hybrid systems to automated highway systems. Discusses relation between hybrid systems and hierarchical control structures.].

Zelikin M.I., Borisov V.F. (1994). *Theory of Chattering Control*. Springer-Verlag. [Analyses optimal control problem leading to Zeno behaviour.].

Zhang J., Johansson K.H., Lygeros J., Sastry S. (2001). Zeno hybrid systems. *International Journal of Robust and Nonlinear Control* **11**(5), 435–451. [Presents properties of Zeno hybrid systems, see Section 4.3.].