Hybrid Systems: Modeling, Analysis and Control

John Lygeros, Claire Tomlin, and Shankar Sastry

December 28, 2008

Contents

List of Figures

Chapter 0

Notation

The following notation is standard and is used throughout the text. Other non–standard notation is defined when introduced in the text and is referenced in the index. A word about the numbering scheme: Not all equations are numbered, but those that are frequently referenced are. Theorems, Claims, Propositions, Corollaries, Lemmas, Definitions, Examples are numbered consecutively in the order in which they appear and they are *all numbered*. Their text is presented in an *emphasized font*. If the theorems, claims, propositions, etc. are specifically noteworthy they are named in **bold font** before the statement. Exercises are at the end of each chapter and are all numbered consecutively, and if especially noteworthy are named like the theorems, claims, propositions, etc. Proofs in the text end with the symbol □ to demarcate the proof from the following text.

Sets

$\ni$	such that
$\forall$	for all
$\exists$	there exists
$a \in A$	a is an element of the set A
$A \subset B$	set A is contained in set B
$A \cup B$	union of set A with set B
$A \cap B$	Intersection of set A with set B
$P(A)$	*power set* of A, i.e. the set of all subsets of A
$p \Rightarrow q$	p implies q
$p \Leftarrow q$	q implies p
$p \Leftrightarrow q$	p is equivalent to q
$\wedge$	logical "and"
$\vee$	logical "or"
$\neg$	logical "not"
M°	interior of a set M
$\overline{M}$	closure of M
$]a, b[$	open subset of the real line
$[a, b]$	closed subset of the real line
$[a, b[$	subset of the real line closed at a, and open at b
$a \to b$	a tends to b
$a \downarrow b$	a *decreases* towards b
$a \uparrow b$	a *increases* towards b
$\oplus$	direct sum of *subspaces*
$Q \times X$	*set product* that is $\{(q, x) \mid q \in Q \text{ and } x \in X\}$

Algebra

$\mathbb{N}$	set of non-negative integers, namely, $(0, 1, 2, \ldots)$
$\mathbb{R}$	field of real numbers
$\mathbb{Z}$	ring of integers, namely, $(\ldots, -1, 0, 1, \ldots)$
j	square root of -1
$\mathbb{C}$	field of complex numbers
$\mathbb{R}_+(\mathbb{R}_-)$	set of non-negative (non-positive) reals
$\mathbb{C}_+(\mathbb{C}_-)$	set of complex numbers in the right (left) half plane, including the imaginary axis
$j\omega$ axis	set of purely imaginary complex numbers
$\mathbb{C}_-^\circ$	$\{s \in \mathbb{C} : \text{Re } s < 0\}$ = interior of $\mathbb{C}_-$
$\mathbb{C}_+^\circ$	$\{s \in \mathbb{C} : \text{Re } s > 0\}$ = interior of $\mathbb{C}_+$
A^n	set of n-tuples of elements belonging to the set A (e.g., $\mathbb{R}^n, \mathbb{R}[s]^n$)
$A^{m\times n}$	set of $m \times n$ arrays with entries in A.
$\sigma(A)$	set of eigenvalues (spectrum) of a square matrix A
$(x_k)_{k\in K}$	family of elements with K, an index set.
$F[x]$	ring of polynomials in one variable x with coefficients in a field F
$F(x)$	field of rational functions in one variable x with coefficients in a field F

Analysis

$f : A \mapsto B$	f maps the domain A into the codomain B
$f(A)$	range of $f := \{y \in B : y = f(x) \text{ for some } x \in A\}$
A°	interior of A
$\bar{A}$	closure of A
∂A	boundary of A
$\|x\|$	*Euclidean norm* $\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$ in $\mathbb{R}^n$
$C([t_0, t_1], \mathbb{R})$	vector space of continuous functions $[t_0, t_1] \mapsto \mathbb{R}$
$C([t_0, t_1])$	vector space of continuous functions $[t_0, t_1] \mapsto \mathbb{R}$
$C^k([t_0, t_1], \mathbb{R})$	vector space of continuous functions $[t_0, t_1] \mapsto \mathbb{R}$ with k continuous derivatives
$C^k([t_0, t_1], \mathbb{R}^n)$	vector space of continuous functions $[t_0, t_1] \mapsto \mathbb{R}^n$ whose first k derivatives are continuous
$\|x\|$	norm of an element x in a vector space
$\langle x, y \rangle$	inner-product of two vectors x, y in a Hilbert space
$\hat{f}, \hat{G}$	Laplace transform of scalar (or vector) function f or matrix function G both defined on $\mathbb{R}_+$
$\dot{f}, \dot{G}$	time derivative of scalar (or vector) function f or matrix function G both defined on $\mathbb{R}_+$
$Df(x)$	derivative of a function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ a matrix $\in \mathbb{R}^{m \times n}$
$D_i f(x_1, x_2, \ldots, x_p)$	Derivative of $f : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_p}$ with respect to the i-th argument
$D^2 f(x)$	second derivative of $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ with respect to its argument, a bi-linear map from $\mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^m$
$D^k_{i_1,\ldots,i_k} f(x_1, x_2, \ldots, x_p)$	k-th partial derivative of $f(x_1, \ldots, x_p)$ with respect to $x_{i_1}, \ldots, x_{i_k}$ a k-linear map from $\mathbb{R}^{n_{i_1}} \times \cdots \times \mathbb{R}^{n_{i_k}} \mapsto \mathbb{R}^m$
$L_p[t_0, t_1]$	vector space of $\mathbb{R}$ valued functions with p-th power integrable over $[t_0, t_1]$
$L^k_p[t_0, t_1]$	vector space of $\mathbb{R}^k$ valued functions with p-th power integrable over $[t_0, t_1]$
$o(x)$	little "o" of x, that is a function $g(x)$, such that $\lim_{\|x\| \to 0} \|g(x)\|/\|x\| = 0$
$O(x)$	capital "O" of x, that is a function $h(x)$, such that $\lim_{\|x\| \to 0} \|h(x)\|$ is well-defined and $\neq 0$

Chapter 1

Introduction

1.1 Embedded Control and Hybrid Systems

While rapid progress in embedded hardware and software makes plausible ever more ambitious multi-layer, multi-objective, adaptive, nonlinear control systems, adequate design methodologies and design support lag far behind. Consequently, today most of the cost in control system development is spent on ad-hoc, prohibitively expensive systems integration, and validation techniques that rely almost exclusively on exhaustively testing more or less complete versions of complex nonlinear control systems. The newest research direction in control addresses this bottleneck by focusing on *predictive* and *systematic hierarchical design* methodologies for building an analytical foundation based on *hybrid systems* and a practical set of *software design tools* which support the construction, integration, safety and performance analysis, on-line adaptation and off-line functional evolution of multi-agent hierarchical control systems. *Hybrid* systems refer to the distinguishing fundamental characteristics of software-based control systems, namely, the tight coupling and interaction of discrete with continuous phenomena. Hybridness is characteristic of all embedded control systems because it arises from several sources. First, the high-level, abstract protocol layers of *hierarchical* control designs are discrete as to make it easier to manage system complexity and to accommodate linguistic and qualitative information; the low-level, concrete control laws are naturally continuous. Second, while individual feedback control scenarios are naturally modeled as interconnections of modules characterized by their continuous input/output behavior, *multi-modal* control naturally suggests a state-based view, with states representing discrete control modes; software-based control systems typically encompass an integrated mixture of both types. Third, every digital *hardware/software implementation* of a control design is ultimately a discrete approximation that interacts through sensors and actuators with a continuous physical environment. The mathematical treatment of hybrid systems is interesting in that it builds on the preceding framework of nonlinear control, but its mathematics is qualitatively distinct from the mathematics of purely discrete or purely continuous phenomena. Over the past several years, we have begun to build basic formal models (*hybrid automata*) for hybrid systems and to develop methods for hybrid control law design, simulation, and verification. Hybrid automata, in particular, integrate diverse models such as differential equations and state machines in a single formalism with a uniform mathematical semantics and novel algorithms for multi-modal control synthesis and for safety and real-time performance analysis.

1.2 Multi-Agent Systems and Hybrid Systems

To a large extent control theory has thus far investigated the paradigm of "Centralized Control". In this paradigm, sensory information is collected from sensors observing a material process that may be distributed over space. This information is transmitted over a communication network to one center, where the commands that guide the process are calculated and transmitted back to the process actuators that implement those commands. In engineering practice, of course, as soon as the process becomes even moderately large, the central control paradigm breaks down. What we find instead is distributed control: A set of control stations, each of whom receives some data and calculates some of the actions. Important examples of distributed control are the Air Traffic Management System, the control system of an interconnected power grid, the telephone network, a chemical process control system, and automated highway transportation systems. Although a centralized control paradigm no longer applies here, control engineers have with great success used its theories and its design and analysis tools to build and operate these distributed control systems. There are two reasons why the paradigm succeeded in practice, even when it failed in principle. First, in each case the complexity and scale of the material process grew incrementally and relatively slowly. Each new increment to the process was controlled using the paradigm, and adjustments were slowly made after extensive (but by no means exhaustive) testing to ensure that the new controller worked in relative harmony with the existing controllers. Second, the processes were operated with a considerable degree of "slack." That is, the process was operated well within its performance limits to permit errors in the extrapolation of test results to untested situations and to tolerate a small degree of disharmony among the controllers. However, in each system mentioned above, there were occasions when the material process was stressed to its limits and the disharmony became intolerable, leading to a spectacular loss of efficiency. For example, most air travelers have experienced delays as congestion in one part of the country is transmitted by the control system to other parts. The distributed control system of the interconnected power grid has sometimes failed to respond correctly and caused a small fault in one part of a grid to escalate into a system-wide blackout.

We are now attempting to build control systems for processes that are vastly more complex or that are to be operated much closer to their performance limits in order to achieve much greater efficiency of resource use. The attempt to use the central control paradigm cannot meet this challenge: the material process is already given and it is not practicable to approach its complexity in an incremental fashion as before. Moreover, the communication and computation costs in the central control paradigm would be prohibitive, especially if we insist that the control algorithms be fault-tolerant. What is needed to meet the challenge of control design for a complex, high performance material process, is a new paradigm for distributed control. It must distribute the control functions in a way that avoids the high communication and computation costs of central control, at the same time that it limits complexity. The distributed control must, nevertheless, permit centralized authority over those aspects of the material process that are necessary to achieve the high performance goals. Such a challenge can be met by organizing the distributed control functions in a hierarchical architecture that makes those functions relatively autonomous (which permits using all the tools of central control), while introducing enough coordination and supervision to ensure the harmony of the distributed controllers necessary for high performance. Consistent with this hierarchical organization are sensing hierarchies with

fan-in of information from lower to higher levels of the hierarchy and a fan-out of control commands from the higher to the lower levels. Commands and information at the higher levels are usually represented symbolically, calling for discrete event control, while at the lower levels both information and commands are continuous, calling for continuous control laws. Interactions between these levels involves hybrid control. In addition protocols for coordination between individual agents are frequently symbolic, again making for hybrid control laws. The hybrid control systems approach has been successful in the control of some extremely important multi-agent systems such as automated highway systems [82, 54, **?**], air traffic control [78], groups of unmanned aerial vehicles, underwater autonomous vehicles [**?**], mobile offshore platforms, to give a few examples.

1.3 Advanced Air Transportation Automation

The introduction of advanced automation into manually operated systems has been extremely successful in increasing the performance and flexibility of such systems, as well as significantly reducing the workload of the human operator. Examples include the automation of mechanical assembly plants, of the telephone system, of the interconnected power grid, as well as transportation system automation such as controllers in high speed trains, automatic braking systems in automobiles, and avionics on board commercial jets. Accompanying this increase in automation is the necessity of ensuring that the automated system always performs as expected. This is especially crucial for *safety critical* systems: if a telephone switch crashes or a power grid node goes down, lives are usually not lost, yet if an error occurs in the automated avionics on board a commercial jet, the results could be disastrous.

Many of today's safety critical systems are growing at such a rate that will make manual operation of them extremely difficult if not impossible in the near future. The Air Traffic Control (ATC) system is an example of such a safety critical system. Air traffic in the United States alone is expected to grow by 5% annually for the next 15 years [**?**], and rates across the Pacific Rim are expected to increase by more than 15% a year. Even with today's traffic, ground holds and airborne delays in flights due to congestion in the skies have become so common that airlines automatically pad their flight times with built-in delay times. Aging air traffic control equipment certainly contributes to these delays: the plan view displays used by controllers to look at radar tracks and flight information are the very same that were installed in the early 1970's, and they fail regularly. The computer systems which calculate radar tracks and store flight plans were designed in the 1980's, using software code that was written in 1972. The introduction of new computers, display units, and communication technologies for air traffic controllers will help alleviate the problems caused by failing equipment, yet the Federal Aviation Administration (FAA) admits that any significant improvement will require that many of the basic practices of ATC be automated [**?**]. For example, today's airspace has a rigid route structure based on altitude and on ground-based navigational "fixes": current practice of air traffic controllers is to route aircraft along predefined paths connecting fixes, to manage the complexity of route planning for several aircraft at once. The rigid structure puts strict constraints on aircraft trajectories, which could otherwise follow wind-optimal or user preferred routes. Also, while a data link between aircraft and ground is being investigated as a replacement for the current voice communication over radio channels between pilot and controller, there

is a limit to the amount of information processing that a controller can perform with this data. Studies in [**?**] indicate that, if there is no change to the structure of ATC, then by the year 2015 there could be a major accident every 7 to 10 days.

The result is a perceived need in the air traffic, airline, and avionics communities for a new *architecture*, which integrates new technologies for data storage, processing, communications, and display, into a safe and efficient air traffic management system. The airlines are proponents of a decentralized architecture featuring *free flight*, meaning that each aircraft plans and tracks its own dynamic trajectory with minimal interference from ATC [**?**]. Many people (air traffic controllers in particular) view this as a radical solution, but a recent study funded by NASA [**?**] suggests that distributing some of the control authority to each aircraft would help improve the efficiency of the system as a whole. In [**?**] we propose an architecture for a new air traffic management system along these lines, in which the aircraft's flight management system uses local sensory information from Global Positioning Systems, Inertial Navigation Systems, and broadcast communication with other aircraft to resolve local conflicts without requesting clearances from ATC. While the degree of decentralization and level of automation in a new air traffic management system are still under debate (since it is very difficult to estimate the increase in efficiency from distributing the control authority), the integrity of any automated functionality in a new air traffic management system depends on a *provably-safe* design, and a high confidence that the control actions won't fail.

In the past, high confidence has been achieved by operating the system well within its performance limits. Extensive testing has been used to validate operations, and any errors occurring from untested situations would be compensated for by this degree of "slack" in the system performance. We would like to maintain high confidence but operate the system much closer to its performance limits. In order to do this, we require accurate models of the system, procedures for verifying that the design is safe to within the accuracy of these models, and procedures for synthesizing control actions for the system, so that safety is maintained.

For about the past six years, researchers in the traditionally distinct fields of control theory and computer science verification have proposed models, and verification and controller synthesis techniques for complex, safety critical systems. The area of *hybrid systems* is loosely defined as the study of systems which involve the interaction of discrete event and continuous time dynamics, with the purpose of proving properties such as reachability and stability. The discrete event models naturally accommodate linguistic and qualitative information, and are used to model modes of operation of the system, such as the mode of flight of an aircraft, or the interaction and coordination between several aircraft. The continuous dynamics model the physical processes themselves, such as the continuous response of an aircraft to the forces of aileron and throttle.

1.4 Literature on Hybrid Systems

One class of approaches to modeling and analysis of hybrid systems has been to extend techniques for finite state automata to include systems with simple continuous dynamics. These approaches generally use one of two analysis techniques: model checking, which verifies a system specification symbolically on all system trajectories, and deductive theorem

proving, which proves a specification by induction on all system trajectories. Emphasis is placed on *computability* and *decidability*, or proving that the problem: *Does the system satisfy the specification?* can be solved in a finite number of steps. Models and decidability results have been obtained for timed automata [4], linear hybrid automata [3], and hybrid input/output automata [59]. Linear hybrid automata model or abstract the continuous dynamics by differential inclusions of the form $A\dot{x} \leq b$ and verify properties of the resulting abstracted system [**?**, **?**]. While reachability and eventuality properties for timed automata have been shown to be decidable, the decidability results for linear hybrid automata are fairly narrow. For all but the simplest continuous linear dynamics (two-dimensional rectangular differential inclusions), reachability properties are semi-decidable at best, and in most cases undecidable. Methods for designing discrete controllers for timed and hybrid systems have been developed using this framework [60, 83], and computational tools have been developed for both model checking [36, 25], and theorem proving [61].

A second class of models and analysis techniques for hybrid systems has developed out of research in continuous state space and continuous time dynamical systems and control. The emphasis here has been on extending the standard modeling, reachability and stability analyses, and controller design techniques to capture the interaction between the continuous and discrete dynamics [20, 17, **?**, **?**, 53, 67]. Analysis and design techniques extend existing control techniques, such as stability theory [17], optimal control [17, 53, 67], and control of discrete event systems [49, 38], to hybrid systems. One area in which results have been hard to come by is the efficient *computation* of reachable sets for hybrid systems whose dynamics are nonlinear or are of order greater than one. Only recently, some attempts to directly approach this problem have been reported in the literature [24, 32].

Our approach to hybrid systems modeling incorporates accurate, nonlinear models of the continuous dynamics with models for discrete event dynamics. We include continuous and discrete input variables to model both parameters that the designer may control as well as disturbance parameters that the designer must control against. Using analysis based on traditional discrete and continuous optimal control techniques, and on two-person zero-sum game theory for automata and continuous dynamical systems, we derive the Hamilton-Jacobi partial differential equations whose solutions describe *exactly* the boundaries of reachable sets. Only then do we approximate: we use a clever numerical technique to solve this equation. These equations are the heart of our general controller synthesis technique for hybrid systems, in which we calculate feedback control laws for the continuous and discrete variables which guarantee that the hybrid system remains in the "safe subset" of the reachable set. While about 10 years ago such a method would have been prohibitively computationally expensive, advances in computational power and new fast methods for integrating PDEs have made such solutions feasible, even for real-time applications. The result is an analytic and numerical method for computing reachable sets and control laws for hybrid systems, which doesn't require a preprocessing step to approximate the dynamics. We have been successful in computing solutions to finite-time examples, but in our method thus far, we have not addressed considerations of decidability and computational complexity.

Chapter 2

Overview of Dynamical Systems

Roughly speaking, a **dynamical system** describes the **evolution** of a **state** over **time**. To make this notion more precise we need to specify what we mean by the terms "evolution", "state" and "time".

Certain dynamical systems can also be influenced by external inputs, which may represent either uncontrollable disturbances (e.g. wind affecting the motion of an aircraft) or control signals (e.g. the commands of the pilot to the aircraft control surfaces and engines). Some dynamical systems may also have outputs, which may represent either quantities that can be measured, or quantities that need to be regulated. Dynamical systems with inputs and outputs are sometimes referred to as **control systems**.

Based on the type of their state, dynamical systems can be classified into:

1. **Continuous**, if the state takes values in Euclidean space $\mathbb{R}^n$ for some $n \geq 1$. We will use $x \in \mathbb{R}^n$ to denote the state of a continuous dynamical system.

2. **Discrete**, if the state takes values in a countable or finite set $\{q_1, q_2, \ldots\}$. We will use q to denote the state of a discrete system. For example, a light switch is a dynamical system whose state takes on two values, $q \in \{ON, OFF\}$. A computer is also a dynamical system whose state takes on a finite (albeit very large) number of values.

3. **Hybrid**, if part of the state takes values in $\mathbb{R}^n$ while another part takes values in a finite set. For example, the closed loop system we obtain when we use a computer to control an inverted pendulum is hybrid: part of the state (namely the state of the pendulum) is continuous, while another part (namely the state of the computer) is discrete.

Based on the set of times over which the state evolves, dynamical systems can be classified as:

1. **Continuous time**, if the set of times is a subset of the real line. We will use $t \in \mathbb{R}$ to denote continuous time. Typically, the evolution of the state of a continuous time system is described by an **ordinary differential equation** (ODE). Think of the linear, continuous time system in state space form

$$\dot{x} = Ax.$$

2. **Discrete time**, if the set of times is a subset of the integers. We will use $k \in \mathbb{Z}$ to denote discrete time. Typically, the evolution of the state of a discrete time system is described by a **difference equation**. Think of the linear discrete time system in state space form
$$x_{k+1} = Ax_k.$$
3. **Hybrid time**, when the evolution is over continuous time but there are also discrete "instants" where something "special" happens. More on this in Chapter **??**.

Continuous state systems can be further classified according to the equations used to describe the evolution of their state

1. **Linear**, if the evolution is governed by a linear differential equation (continuous time) or difference equation (discrete time).
2. **Nonlinear**, if the evolution is governed by a nonlinear differential equation (continuous time) or difference equation (discrete time).

Exercise 2.1 *The linear vs nonlinear classification generally does not apply to discrete state or hybrid systems. Why?*

In this class we will start by giving some examples of the the following classes of systems:

1. Nonlinear (continuous state), continuous time systems.
2. Nonlinear, discrete time systems.
3. Discrete state, discrete time systems.

We will then concentrate on hybrid state, hybrid time systems and highlight the differences from the other classes. Classes of systems that will not be treated at all include:

- Infinite dimensional continuous state systems described, for example, by partial differential equations (PDE).
- Discrete state systems with an infinite number of states, e.g. Petri nets, push down automata, Turing machines.
- Stochastic systems, i.e. systems with probabilistic dynamics.

2.1 Examples

2.1.1 Pendulum: A Nonlinear, Continuous Time System

Consider a pendulum hanging from a weight-less solid rod and moving under gravity (Figure 2.1). Let θ denote the angle the pendulum makes with the downward vertical, l the length of the pendulum, m its mass, and d the dissipation constant. The evolution of θ is governed by
$$ml\ddot{\theta} + dl\dot{\theta} + mg\sin(\theta) = 0$$
This is a nonlinear, second order, ordinary differential equation (ODE).

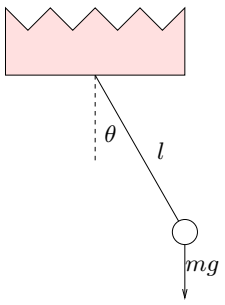


Figure 2.1: The pendulum

Exercise 2.2 *Derive this equation from Newton's laws. Why is this ODE called nonlinear?*

To determine how the pendulum is going to move, i.e. determine θ as a function of time, we would like to find a solution to this ODE . Assuming that at time $t = 0$ the pendulum starts as some initial position θ_0 and with some initial velocity $\dot{\theta}_0$, "solving the ODE" means finding a function of time

$$\theta(\cdot) : \mathbb{R} \to \mathbb{R}$$

such that

$$\begin{aligned} \theta(0) &= \theta_0 \\ \dot{\theta}(0) &= \dot{\theta}_0 \\ ml\ddot{\theta}(t) + dl\dot{\theta}(t) + mg\sin(\theta(t)) &= 0, \ \forall t \in \mathbb{R} \end{aligned}$$

Such a function is known as a **trajectory** (or **solution**) of the system. At this stage it is unclear if one, none or multiple trajectories exist for this initial condition. **Existence** and **uniqueness** of trajectories are both desirable properties for ODE that are used to model physical systems.

For nonlinear systems, even if a unique trajectory exists for the given initial condition, it is usually difficult to construct explicitly. Frequently solutions of ODE can only be approximated by **simulation**. Figure 2.2 shows a simulated trajectory of the pendulum for $l = 1$, $m = 1$, $d = 1$, $g = 9.8$, $\theta(0) = 0.75$ and $\dot{\theta}(0) = 0$.

To simplify the notation we typically write dynamical system ODE in **state space** form

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

where x is now a vector in $\mathbb{R}^n$ for some appropriate $n \geq 1$. The easiest way to do this for the pendulum is to set

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

which gives rise to the state space equations

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin(x_1) - \frac{d}{m}x_2 \end{bmatrix} = f(x)$$

The vector

$$x \in \mathbb{R}^2$$

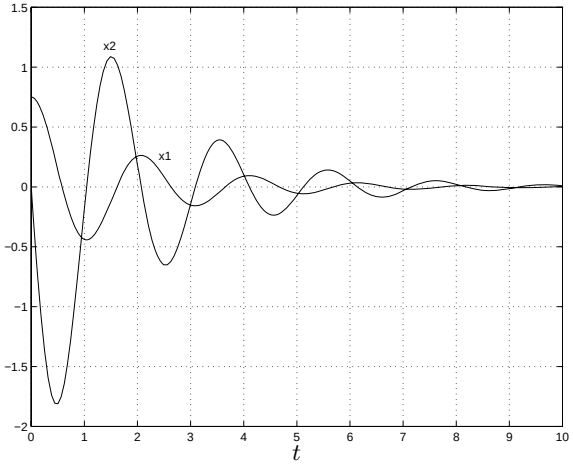


Figure 2.2: Trajectory of the pendulum.

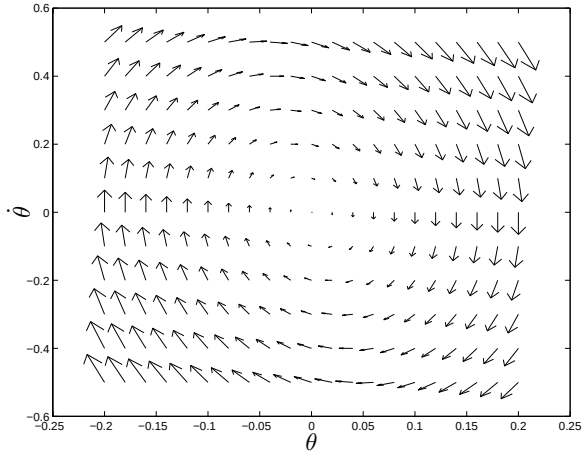


Figure 2.3: The pendulum vector field.

is called the **state** of the system. The size of the state vector (in this case $n = 2$) is called the **dimension** of the system. Notice that the dimension is the same as the order of the original ODE. The function

$$f(\cdot) : \mathbb{R}^2 \to \mathbb{R}^2$$

which describes the dynamics is called a **vector field**, because is assigns a "velocity" vector to each state vector. Figure 2.3 shows the vector field of the pendulum.

Exercise 2.3 *Other choices are possible for the state vector. For example, for the pendulum one can use* $x_1 = \theta^3 + \dot{\theta}$ *and* $x_2 = \dot{\theta}$*. What would the vector field be for this choice of state?*

Solving the ODE for θ is equivalent to finding a function

$$x(\cdot) : \mathbb{R} \to \mathbb{R}^2$$

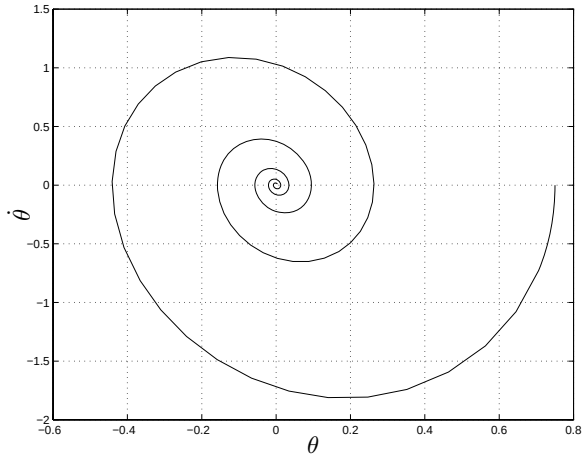


Figure 2.4: Phase plane plot of the trajectory of Figure 2.2.

such that

$$x(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \dot{\theta}_0 \end{bmatrix}$$
$$\dot{x}(t) = f(x(t)),\ \forall t \in \mathbb{R}.$$

For two dimensional systems like the pendulum it is very convenient to visualise the solutions by **phase plane** plots. These are plots of $x_1(t)$ vs $x_2(t)$ parameterised by time (Figure 2.4).

2.1.2 Logistic Map: A Nonlinear Discrete Time System

The logistic map

$$x_{k+1} = ax_k(1 - x_k) = f(x_k) \tag{2.1}$$

is a nonlinear, discrete time dynamical system that has been proposed as a model for the fluctuations in the population of fruit flies in a closed container with constant food supply [65]. We assume that the population is measured at discrete times (e.g. generations) and that it is large enough to be assumed to be a continuous variable. In the terminology of the previous section, this is a one dimensional system with state $x_k \in \mathbb{R}$, whose evolution is governed by the difference equation (2.1) given above.

The shape of the function f (Figure 2.5) reflects the fact that when the population is small it tends to increase due to abundance of food and living space, whereas when the population is large it tends to decrease, due to competition for food and the increased likelihood of epidemics. Assume that $a \leq 4$ and that the initial population is such that $0 \leq x_0 \leq 1$.

Exercise 2.4 *Show that under these assumptions* $0 \leq x_k \leq 1$ *for all* $k \in \mathbb{Z}$ *with* $k \geq 0$.

The behaviour of x_k as a function of k depends on the value of a.

1. If $0 \leq a < 1$, x_k decays to 0 for all initial conditions $x_0 \in [0, 1]$. This corresponds to a situation where there is inadequate food supply to support the population.

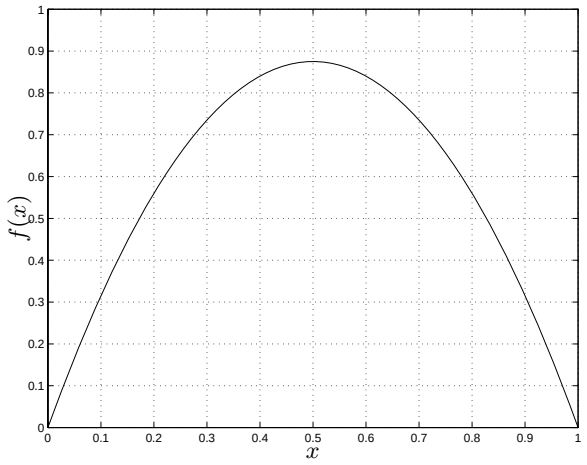


Figure 2.5: The logistic map.

2. If $1 \leq a \leq 3$, x_k tends to a steady state value. In this case the population eventually stabilises.

3. If $3 < a \leq 1 + \sqrt{6} = 3.449$, x_k tends to a 2-periodic state. This corresponds to the population alternating between two values from one generation to the next.

As a increases further more and more complicated patterns are obtained: 4-periodic points, 3-periodic points, and even chaotic situations, where the trajectory of x_k is a-periodic (i.e. never meets itself).

2.1.3 Manufacturing Machine: A Discrete System

Consider a machine in a manufacturing plant that processes parts of type p one at a time. The machine can be in one of three states: Idle (I), Working (W) or Down (D). The machine can transition between the states depending on certain events. For example, if the machine is idle and a part p arrives it will start working. While the machine is working it may break down. While the machine is down it may be repaired, etc.

Abstractly, such a machine can be modelled as a dynamical system with a **discrete state**, q, taking three values

$$q \in Q = \{I, W, D\}$$

The state "jumps" from one value to another whenever one of the **events**, σ occurs, where

$$\sigma \in \Sigma = \{p, c, f, r\}$$

(p for "part arrives", c for "complete processing", f for "failure" and r for "repair"). The state after the event occurs is given by a **transition relation**

$$\delta : Q \times \Sigma \to Q$$

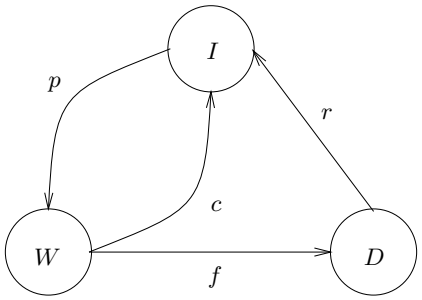


Figure 2.6: The directed graph of the manufacturing machine automaton.

Since both Q and Σ are finite sets, one can specify δ by enumeration.

$$\begin{aligned}\delta(I,p) &= W\\ \delta(W,c) &= I\\ \delta(W,f) &= D\\ \delta(D,r) &= I\end{aligned}$$

δ is undefined for the rest of the combinations of q and σ. This reflects the fact that certain events may be impossible in certain states. For example, it is impossible for the machine to start processing a part while it is down, hence $\delta(D,p)$ is undefined.

Exercise 2.5 *If the discrete state can take n values and there are m possible events, what is the maximum number of lines one may have to write down to specify δ?*

Such a dynamical system is called an **automaton**, or a **finite state machine**. Automata are special cases of **discrete event systems**. Discrete event systems are dynamical systems whose state also jumps depending on a finite set of events but can take on an infinite number of values. The dynamics of a finite state machine can be represented compactly by a **directed graph** (Figure 2.6). This is a graph whose **nodes** represent the possible values of the state (in this case I, W, D). The **arcs** of the graph represent possible transitions between the state values and are labelled by the events.

Exercise 2.6 *What is the relation between the number of arcs of the graph and the number of lines one needs to write down in order to specify δ?*

Assume that the machine starts in the idle state $q_0 = I$. What are the sequences of events the machine can experience? Clearly some sequences are possible while others are not. For example, the sequence *pcp* is possible: the machine successfully processes one part and subsequently starts processing a second one. The sequence *ppc*, on the other hand is not possible: the machine can not start processing a second part before the previous one is complete. More generally, any sequence that consists of an arbitrary number of *pc*'s (possibly followed by a single p) is an acceptable sequence. In the discrete event literature this set of sequences is compactly denoted as

$$(pc)^*(1+p)$$

where * denotes an arbitrary number (possibly zero) of pc's, 1 denotes the empty sequence (no event takes place), and + denotes "or".

Likewise, pfr is a possible sequence of events (the machine starts processing a part, breaks down and then gets repaired) while pfp is not (the machine can not start processing a part while it is down). More generally, any sequence that consists of an arbitrary number of pfr's (possibly followed by a p or a pf) is an acceptable sequence.

Exercise 2.7 *Write this set of sequences in the discrete event notation given above.*

The set of all sequences that the automaton can experience is called the **language** of the automaton. The above discussion suggests that the language of the machine automaton is

$$(pc + pfr)^*(1 + p + pf)$$

It is important to understand the properties of these languages, for example to determine how to schedule the work in a manufacturing plant.

2.1.4 Thermostat: A Hybrid System

Consider a room being heated by a radiator controlled by a thermostat. Assume that when the radiator is off the temperature, $x \in \mathbb{R}$, of the room decreases exponentially towards 0 degrees according to the differential equation

$$\dot{x} = -ax \tag{2.2}$$

for some $a > 0$.

Exercise 2.8 *Verify that the trajectories of* (2.2) *decrease to* 0 *exponentially.*

When the thermostat turns the heater on the temperature increases exponentially towards 30 degrees, according to the differential equation

$$\dot{x} = -a(x - 30). \tag{2.3}$$

Exercise 2.9 *Verify that the trajectories of* (2.3) *increase towards* 30 *exponentially.*

Assume that the thermostat is trying to keep the temperature at around 20 degrees. To avoid "chattering" (i.e. switching the radiator on an off all the time) the thermostat does not attempt to turn the heater on until the temperature falls below 19 degrees. Due to some uncertainty in the radiator dynamics, the temperature may fall further, to 18 degrees, before the room starts getting heated. Likewise, the thermostat does not attempt to turn the heater on until the temperature rises above 21 degrees. Due to some uncertainty in the radiator dynamics the temperature may rise further, to 22 degrees, before the room starts to cool down. A trajectory of the thermostat system is shown in Figure 2.7. Notice that in this case multiple trajectories may be obtained for the same initial conditions, as for certain values of the temperature there is a choice between switching the radiator on/off or not. Systems for which such a choice exists are known as **non-deterministic**.

Notice that this system has both a continuous and a discrete state. The continuous state is the temperature in the room $x \in \mathbb{R}$. The discrete state, $q \in \{ON, OFF\}$ reflects whether

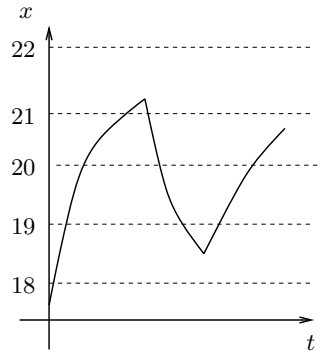


Figure 2.7: A trajectory of the thermostat system.

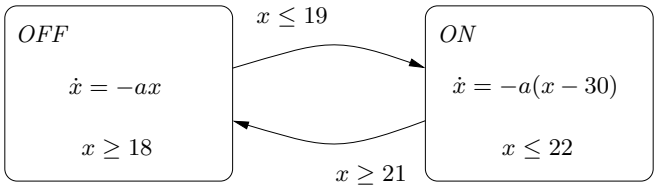


Figure 2.8: Directed graph notation for the thermostat system.

the radiator is on or off. The evolution of x is governed by a differential equation (as was the case with the pendulum), while the evolution of q is through jumps (as was the case with the manufacturing machine). The evolution of the two types of state is coupled. When $q = ON$, x rises according to differential equation (2.3), while when $q = OFF$, x decays according to differential equation (2.2). Likewise, q *can not jump* from ON to OFF unless $x \geq 21$. q *must jump* from ON to OFF if $x \geq 22$. Etc.

It is very convenient to compactly describe such **hybrid systems** by mixing the differential equation with the directed graph notation (Figure 2.8).

2.2 Review of Continuous State Space Systems

All continuous nonlinear systems considered in this class can be reduced to the standard **state space** form. It is usual to denote

- the **states** of the system by $x_i \in \mathbb{R}$, $i = 1, \ldots, n$,
- the **inputs** by $u_j \in \mathbb{R}$, $j = 1, \ldots, m$, and
- the **outputs** by $y_k \in \mathbb{R}$, $k = 1, \ldots, p$.

The number of states, n, is called the **dimension** (or **order**) of the system. The evolution of the states, inputs and outputs is governed by a set of functions

$$f_i : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}, \text{ for } i = 1, \ldots, n$$
$$h_j : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}, \text{ for } j = 1, \ldots, p$$

Roughly speaking, at a given time $t \in \mathbb{R}$ and for given values of all the states and inputs these functions determine in what direction the state will move, and what the output is going to be.

$$\begin{aligned}
\dot{x}_1 &= f_1(x_1, \ldots, x_n, u_1, \ldots, u_m, t) \\
&\vdots \\
\dot{x}_n &= f_n(x_1, \ldots, x_n, u_1, \ldots, u_m, t) \\
y_1 &= h_1(x_1, \ldots, x_n, u_1, \ldots, u_m, t) \\
&\vdots \\
y_p &= h_p(x_1, \ldots, x_n, u_1, \ldots, u_m, t)
\end{aligned}$$

Exercise 2.10 *What is the dimension of the pendulum example? What are the functions f_i?*

It is usually convenient to simplify the equations somewhat by introducing vector notation. Let

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n, \; u = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix} \in \mathbb{R}^m, \; y = \begin{bmatrix} y_1 \\ \vdots \\ y_p \end{bmatrix} \in \mathbb{R}^p,$$

and define

$$\begin{aligned}
f &: \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n \\
h &: \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^p
\end{aligned}$$

by

$$f(x,u,t) = \begin{bmatrix} f_1(x_1, \ldots, x_n, u_1, \ldots, u_m, t) \\ \vdots \\ f_n(x_1, \ldots, x_n, u_1, \ldots, u_m, t) \end{bmatrix}, \; h(x,u,t) = \begin{bmatrix} h_1(x_1, \ldots, x_n, u_1, \ldots, u_m, t) \\ \vdots \\ h_p(x_1, \ldots, x_n, u_1, \ldots, u_m, t) \end{bmatrix}.$$

Then the system equations simplify to

$$\left.\begin{aligned} \dot{x} &= f(x,u,t) \\ y &= h(x,u,t) \end{aligned}\right\} \tag{2.4}$$

Equations (3.1) are known as the **state space form** of the system. The vector space $\mathbb{R}^n$ in which the state of the system takes values is known as the **state space** of the system. If the system is of dimension 2, the state space is also referred to as the **phase plane**. The function f that determines the direction in which the state will move is known as the **vector field**.

Notice that the differential equation for x is first order, i.e. involves $\dot{x}$ but no higher derivatives of x. Sometimes the system dynamics are given to us in the form of higher order differential equations, i.e. equations involving a variable $\theta \in \mathbb{R}$ and its derivatives with respect to time up to $\frac{d^r\theta}{dt^r}$ for some integer $r \geq 1$. Such systems can be easily transformed to state space form by setting $x_1 = \theta$, $x_2 = \dot{\theta}$, $\ldots$, $x_{r-1} = \frac{d^{r-1}\theta}{dt^{r-1}}$.

Exercise 2.11 *Consider the system*

$$\frac{d^r\theta}{dt^r} + g(\theta, \frac{d\theta}{dt}, \ldots, \frac{d^{r-1}\theta}{dt^{r-1}}) = 0$$

Write this system in state space form.

It may of course happen in certain examples that there are no inputs or outputs, or that there is no explicit dependence of the dynamics on time. Systems of the form

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

(i.e. without inputs or outputs and with no explicit dependence on time) are called **autonomous** systems.

Exercise 2.12 *Is the pendulum an autonomous system?*

Exercise 2.13 *Consider a non-autonomous system of the form $\dot{x} = f(x,t)$, of dimension n. Show that it can be transformed to an autonomous system of dimension $n+1$. (Hint: append t to the state).*

2.2.1 Existence and Uniqueness of Solutions

Consider an autonomous dynamical system in state space form

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

and assume that at time $t = 0$ the state is equal to x_0, i.e.

$$x(0) = x_0$$

We would like to "solve" the dynamics of the system to determine how the state will evolve in the future (i.e. for $t \geq 0$). More precisely, given some $T > 0$ we would like to determine a function

$$x(\cdot) : [0, T] \to \mathbb{R}^n$$

such that

$$\begin{aligned} x(0) &= x_0 \\ \dot{x}(t) &= f(x(t)), \forall t \in [0, T]. \end{aligned}$$

Such a function $x(\cdot)$ is called a **trajectory** (or **solution**) of the system. Notice that given a candidate trajectory $x(\cdot) : [0, T] \to \mathbb{R}^n$ one needs to verify both the differential condition and the initial condition to ensure that $x(\cdot)$ is indeed a solution of the differential equation.

Exercise 2.14 *Assume instead of $x(0) = x_0$ it is required that $x(t_0) = x_0$ for some $t_0 \neq 0$. Show how one can construct solutions to the system*

$$x(t_0) = x_0,\ \dot{x} = f(x)$$

from solutions to

$$x(0) = x_0,\ \dot{x} = f(x)$$

by appropriately redefining t. Could you do this with a non-autonomous system?

Without any additional information, it is unclear whether one can find a function $x(\cdot)$ solving the differential equation. A number of things can go wrong.

Example (No solutions) Consider the one dimensional system

$$\dot{x} = -sign(x),\ x(0) = 0$$

A solution to this differential equation does not exist for any $T \geq 0$.

Exercise 2.15 *Assume that $x(0) = 1$. Show that solutions to the system exist for all $T \leq 1$ but not for $T > 1$.*

Incidentally, something similar would happen with the radiator system if the thermostat insisted on switching the radiator on and off exactly at 20 degrees. ■

Example (Multiple Solutions) Consider the one dimensional system

$$\dot{x} = 3x^{2/3},\ x(0) = 0$$

All functions of the form

$$x(t) = \begin{cases} (t-a)^3 & t \geq a \\ 0 & t \leq a \end{cases}$$

for any $a \geq 0$ are solutions of this differential equation.

Exercise 2.16 *Verify this.*

Notice that in this case the solution is not unique. In fact there are infinitely many solutions, one for each $a \geq 0$. ■

Example (Finite Escape Time) Consider the one dimensional system

$$\dot{x} = 1 + x^2,\ x(0) = 0$$

The function

$$x(t) = tan(t)$$

is a solution of this differential equation.

Exercise 2.17 *Verify this. What happens at $t = \pi/2$?*

Notice that the solution is defined for $T < \pi/2$ but not for $T \geq \pi/2$. ■

To eliminate such pathological cases we need to impose some assumptions on f.

Definition 2.1 *A function $f : \mathbb{R}^n \to \mathbb{R}^n$ is called* **Lipschitz continuous** *if there exists $\lambda > 0$ such that for all $x, \hat{x} \in \mathbb{R}^n$*

$$\|f(x) - f(\hat{x})\| < \lambda \|x - \hat{x}\|$$

λ is known as the Lipschitz constant. A Lipschitz continuous function is continuous, but not necessarily differentiable. All differentiable functions with bounded derivatives are Lipschitz continuous.

Exercise 2.18 *Show that for $x \in \mathbb{R}$ the function $f(x) = |x|$ that returns the absolute value of x is Lipschitz continuous. What is the Lipschitz constant? Is f continuous? Is it differentiable?*

Theorem 2.1 (Existence & Uniqueness of Solutions) *If f is Lipschitz continuous, then the differential equation*

$$\dot{x} = f(x)$$
$$x(0) = x_0$$

has a unique solution $x(\cdot) : [0, T] \to \mathbb{R}^n$ for all $T \geq 0$ and all $x_0 \in \mathbb{R}^n$.

Exercise 2.19 *Three examples of dynamical systems that do not have unique solutions were given above. Why do these systems fail to meet the conditions of the theorem? (The details are not easy to get right.)*

This theorem allows us to check whether the differential equation models we develop make sense. It also allows us to spot potential problems with proposed solutions. For example, uniqueness implies that solutions can not cross.

Exercise 2.20 *Why does uniqueness imply that trajectories can not cross? (Hint: what would happen at the crossing point?).*

2.2.2 Continuity with Respect to Initial Condition and Simulation

Theorem 2.2 (Continuity with Initial State) *Assume f is Lipschitz continuous with Lipschitz constant λ. Let $x(\cdot) : [0, T] \to \mathbb{R}^n$ and $\hat{x}(\cdot) : [0, T] \to \mathbb{R}^n$ be solutions to $\dot{x} = f(x)$ with $x(0) = x_0$ and $\hat{x}(0) = \hat{x}_0$ respectively. Then for all $t \in [0, T]$*

$$\|x(t) - \hat{x}(t)\| \leq \|x_0 - \hat{x}_0\| e^{\lambda t}$$

In other words, solutions that start close to one another remain close to one another.

This theorem provides another indication that dynamical systems with Lipschitz continuous vector fields are well behaved. For example, it provides theoretical justification for simulation algorithms. Most nonlinear differential equations are impossible to solve by hand. One can however approximate the solution on a computer, using numerical algorithms for computing integrals (Euler, Runge-Kutta, etc.). This is a process known as **simulation**.

Powerful computer packages, such as Matlab, make the simulation of most systems relatively straight forward. For example, the code used to generate the pendulum trajectories is based on a Matlab function

```
function [xprime] = pendulum(t,x)
xprime=[0; 0];
```

```
l = 1;
m=1;
d=1;
g=9.8;
xprime(1) = x(2);
xprime(2) = -sin(x(1))*g/l-x(2)*d/m;
```

The simulation code is then simply

```
>> x=[0.75 0];
>> [T,X]=ode45('pendulum', [0 10], x');
>> plot(T,X);
>> grid;
```

The continuity property ensures that the numerical approximation to the solution computed by the simulation algorithms and the actual solution remain close.

Chapter 3

Models for Hybrid Systems

3.1 Examples of Hybrid Systems

Roughly speaking, hybrid systems are dynamical systems that involve the interaction of different types of dynamics. In this class we are interested in hybrid dynamics that arise out of the interaction of continuous state dynamics and discrete state dynamics. Recall that a state variable is called discrete if it takes on a finite (or countable) number of values and continuous if it takes values in Euclidean space $\mathbb{R}^n$ for some $n \geq 1$. By their nature, discrete states can change value only through a discrete "jump" (c.f. the machining example in Chapter **??**). Continuous states can change values either through a jump (c.f. the logistic map example in Chapter **??**), or by "flowing" in continuous time according to a differential equation (c.f. the pendulum example in Chapter **??**). Hybrid systems involve both these types of dynamics: discrete jumps and continuous flows. The analysis and design of hybrid systems is in general more difficult than that of purely discrete or purely continuous systems, because the discrete dynamics may affect the continuous evolution and vice versa.

Hybrid dynamics provide a convenient framework for modelling systems in a wide range of engineering applications:

- In **mechanical systems** continuous motion may be interrupted by collisions.
- In **electrical circuits** continuous phenomena such as the charging of capacitors, etc. are interrupted by switches opening and closing, or diodes going on or off.
- In **chemical process control** the continuous evolution of chemical reactions is controlled by valves and pumps.
- In **embedded computation** systems a digital computer interacts with a mostly analogue environment.

In all these systems it is convenient (and usually fairly accurate) to model the "discrete" components (switches, valves, computers, etc.) as introducing instantaneous changes in the "continuous" components (charging of capacitors, chemical reactions, etc.).

We start our study of hybrid systems by providing a number of examples of hybrid behaviour.

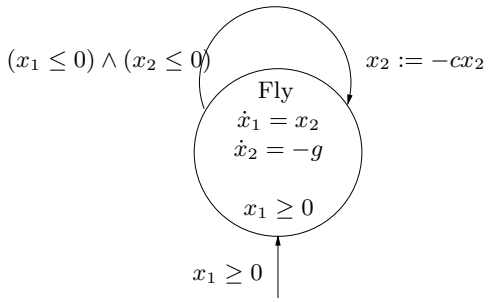


Figure 3.1: Bouncing ball

3.1.1 The Bouncing Ball

A model for a bouncing ball can be represented as a simple hybrid system (Figure 3.1) with single discrete state and a continuous state of dimension two

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

where x_1 denotes the vertical position of the ball and x_2 its vertical velocity.

The continuous motion of the ball is governed by Newton's laws of motion. This is indicated by the differential equation that appears in the *vertex* (box), where g denotes the gravitational acceleration. This differential equation is only valid as long as $x_1 \geq 0$, i.e., as long as the ball is above the ground. This is indicated by the logical expression $x_1 \geq 0$ that appears in the vertex below the differential equation.

The ball bounces when $x_1 = 0$ and $x_2 \leq 0$. This is indicated by the logical expression that appears near the beginning of the edge (arrow). At each bounce, the ball loses a fraction of its energy. This is indicated by the equation $x_2 := -cx_2$ (with $c \in [0, 1]$) that appears near the end of the edge. This is an assignment statement, which means that after the bounce the speed of the ball will be c times the speed of the ball before the bounce, and in the opposite direction.

Exercise 3.1 *Show that energy is preserved during continuous evolution. What fraction of the energy of the ball is lost at each bounce? What is the time interval that elapses between two bounces as a function of the energy of the ball?*

Starting at an initial state with $x_1 \geq 0$ (as indicated by the logical condition next to the arrow pointing to the vertex), the continuous state flows according to the differential equation 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. Such a trajectory is called an *execution* (and sometimes a *run* or a *solution*) of the hybrid system.

3.1.2 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 3.2 shows a model of a car with a gear

box having four gears.

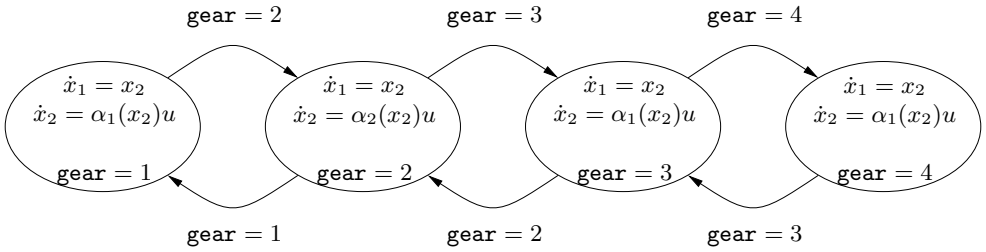


Figure 3.2: A hybrid system modelling a car with four gears.

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 $u \in [u_{\min}, u_{\max}]$. Gear shifting 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. Typical shapes of the functions α_i are shown in Figure 3.3.

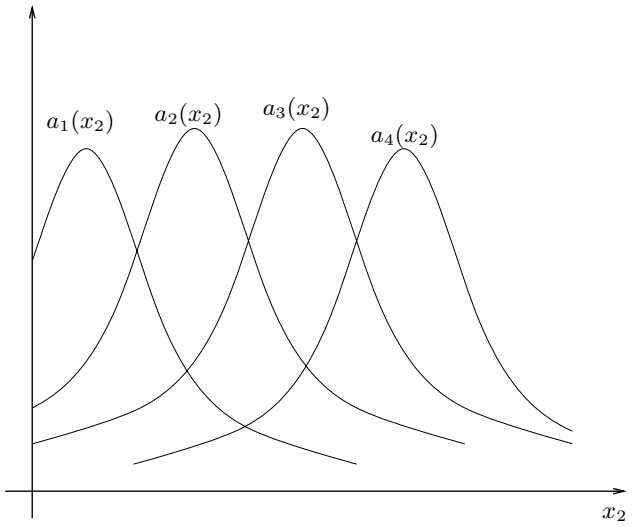


Figure 3.3: The efficiency functions of the different gears.

Exercise 3.2 *How many real valued continuous states does this model have? How many discrete states?*

Several interesting control problems can be posed for this simple car model. For example, what is the optimal control strategy to drive from $(a, 0)$ to $(b, 0)$ in minimum time? The problem is not trivial if we include the reasonable assumption that each gear shift takes a certain amount of time. The optimal controller, which can be modelled as a hybrid system, may be derived using the theory of optimal control of hybrid systems.

3.1.3 Computer-Controlled System

Hybrid systems are natural models for computer-controlled systems (Figure 3.4), 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, 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, for example, in embedded control systems.

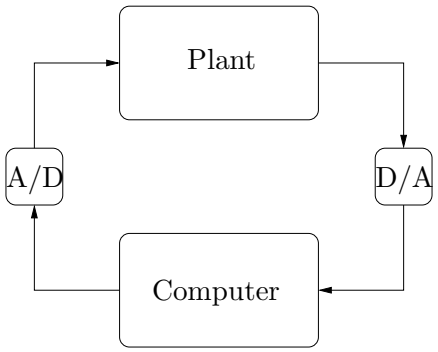


Figure 3.4: Computer-controlled system.

3.1.4 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. The use of an automated system that performs some or all of the tasks of the driver 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 [82] assumes that traffic on the highway is organised 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

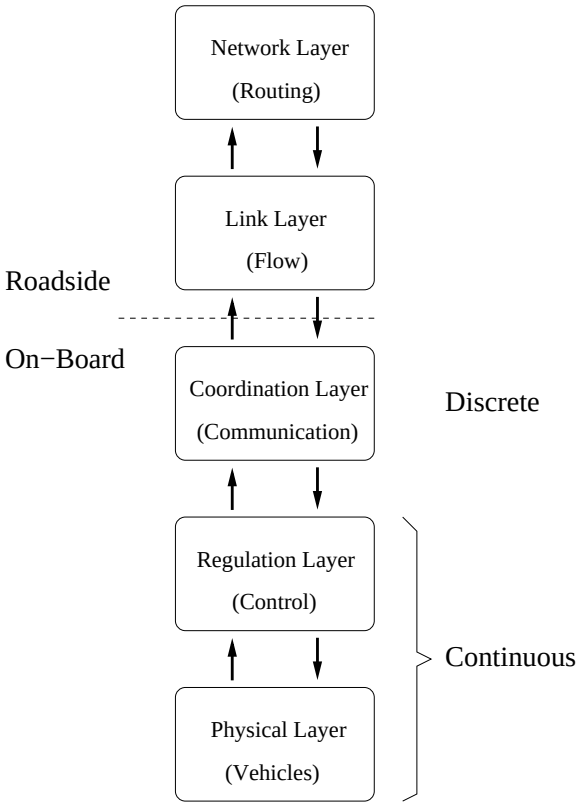


Figure 3.5: The AHS control hierarchy.

with the last vehicle of the platoon ahead. The more unintuitive spacing policy is that of the followers, who 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, computational 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 is used. The controller is organised in four layers (Figure 3.5). The top two layers, called network and link, reside on the roadside and are primarily concerned with throughput maximisation, 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 example, several highways around an urban area. Its task is to prevent congestion and maximise throughput by dynamically routing traffic. The link layer coordinates the operation of sections (links) of the highway (for example the highway segment between two exits). Its

primary concern is to maximise 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 minimise 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 utilises 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 appropriate continuous controllers to ensure the safety of the closed loop hybrid system.

3.2 Hybrid Automata

To model all these diverse phenomena 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 (e.g. the thermostat) to capture uncertainty, etc.
- *composable*, to allow one to build large models by composing models of simple components (e.g. for the AHS application).
- *abstractable*, to allow one to refine design problems for composite models down to design problems for individual components and, conversely, compose results about the performance of individual components to study the performance for the overall system.

Modelling languages that possess at least some subset of these properties have been developed in the hybrid systems literature. Different languages place more emphasis on different aspects, depending on the applications and problems they are designed to address. In this

class we will concentrate on one such language, called *hybrid automata*. The hybrid automata we will study are fairly rich (in terms of descriptive power), but are autonomous, i.e. have no inputs and outputs. They are therefore unsuitable for studying composition and abstraction properties.

A Hybrid Automation is a dynamical system that describes the evolution in time of the values of a set of discrete and continuous variables.

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

- Q *is a set of discrete variables and* Q *is countable;*
- X *is a set of continuous variables;*
- $\mathrm{I}nit \subseteq \mathbf{Q} \times \mathbf{X}$ *is a set of initial states;*
- $f : \mathbf{Q} \times \mathbf{X} \to T\mathbf{X}$ *is a vector field;*
- $\mathrm{I}nv : \mathbf{Q} \to P(X) := 2^{\mathbf{X}}$ *assigns to each* $q \in \mathbf{Q}$ *an invariant set;*
- $E \subset \mathbf{Q} \times \mathbf{Q}$ *is a collection of discrete transitions;*
- $G : E \to P(X)$ *assigns to each* $e = (q, q') \in E$ *a guard; and*
- $R : E \times \mathbf{X} \to P(X)$ *assigns to each* $e = (q, q') \in E$ *and* $x \in \mathbf{X}$ *a reset relation.*

Recall that $P(X)$ denotes the power set (set of all subsets) of X. The notation of Definition 4.9 suggests, for example, that the function Dom assigns a set of continuous states D$om(q) \subseteq \mathbb{R}^n$ to to each discrete state $q \in Q$. We refer to $(q, x) \in Q \times X$ as the *state* of H.

Hybrid automata define possible evolutions for their state. Roughly speaking, starting from an initial value $(q_0, x_0) \in \mathrm{I}nit$, the continuous state x flows according to the differential equation

$$\begin{aligned} \dot{x} &= f(q_0, x), \\ x(0) &= x_0, \end{aligned}$$

while the discrete state q remains constant

$$q(t) = q_0.$$

Continuous evolution can go on as long as x remains in D$om(q_0)$. If at some point the continuous state x reaches the guard $G(q_0, q_1) \subseteq \mathbb{R}^n$ of some edge $(q_0, q_1) \in E$, the discrete state may change value to q_1. At the same time the continuous state gets reset to some value in $R(q_0, q_1, x) \subseteq \mathbb{R}^n$. After this discrete transition, continuous evolution resumes and the whole process is repeated.

To simplify the discussion, we assume from now on that the number of discrete states is finite, and that for all $q \in Q$, the vector field $f(q, \cdot)$ is Lipschitz continuous. Recall that this ensures that the solutions of the differential equation $\dot{x} = f(q, x)$ are well defined (Chapter **??**). Finally, we assume that for all $e \in E$, $G(e) \neq \emptyset$, and for all $x \in G(e)$,

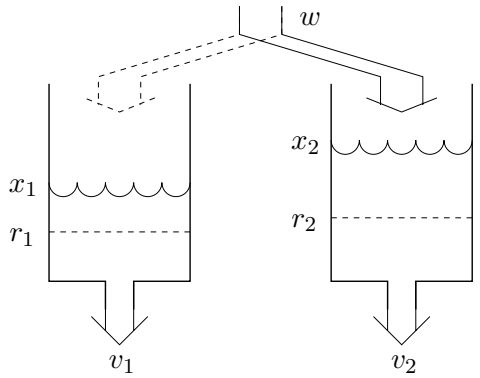


Figure 3.6: The water tank system.

$R(e, x) \neq \emptyset$. This assumption eliminates some pathological cases and in fact be imposed without loss of generality.

As we saw in Chapter **??** and in the examples discussed above, it is often convenient to visualise 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 $\{x \in \mathbf{X} \mid (q, x) \in \mathrm{I}nit\}$, a vector field $f(q, \cdot) : \mathbb{R}^n \to \mathbb{R}^n$ and a domain $\mathrm{D}om(q) \subseteq \mathbb{R}^n$. An edge $(q, q') \in E$ starts at $q \in Q$ and ends at $q' \in Q$. With each edge $(q, q') \in E$, we associate a guard $G(q, q') \subseteq \mathbb{R}^n$ and a reset function $R(q, q', \cdot) : \mathbb{R}^n \to P(\mathbb{R}^n)$.

Example (Water Tank System) The two tank system, shown in Figure 3.6, consists of two tanks containing water. Both tanks are leaking at a constant rate. Water is added to the system at a constant rate through a hose, which at any point in time is dedicated to either one tank or the other. It is assumed that the hose can switch between the tanks instantaneously.

For $i = 1, 2$, 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. 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 $x_2 \leq r_2$.

It is straight forward to define a hybrid automaton, to describe this process:

- $Q = \{q_1, q_2\}$ (two discrete states, inflow going left and inflow going right);
- $X = \mathbb{R}^2$ (two continuous states, the level of water in the two tanks);
- (when the inflow is going to the tank on the right, the water level in the left tank goes down while the water level in right tank goes up, and vice verse)

$$f(q_1, x) = \begin{bmatrix} w - v_1 \\ -v_2 \end{bmatrix}, \text{ and } f(q_2, x) = \begin{bmatrix} -v_1 \\ w - v_2 \end{bmatrix};$$

- $\mathrm{I}nit = \{q_1, q_2\} \times \{x \in \mathbb{R}^2 \mid x_1 \geq r_1 \wedge x_2 \geq r_2\}$ (start with both water levels above the low level marks r_1 and r_2);

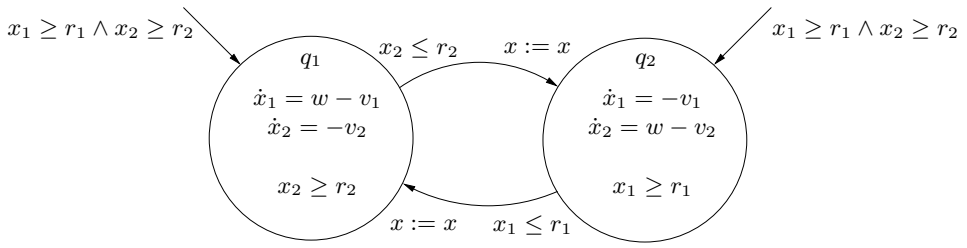


Figure 3.7: Graphical representation of the water tank hybrid automaton.

- $\mathrm{D}om(q_1) = \{x \in \mathbb{R}^2 \mid x_2 \geq r_2\}$ and $\mathrm{D}om(q_2) = \{x \in \mathbb{R}^2 \mid x_1 \geq r_1\}$ (put water in the current tank as long as the level in the other tank is above the low level mark);
- $E = \{(q_1, q_2), (q_2, q_1)\}$ (possible to switch inflow from left to right and vice versa);
- $G(q_1, q_2) = \{x \in \mathbb{R}^2 \mid x_2 \leq r_2\}$ and $G(q_2, q_1) = \{x \in \mathbb{R}^2 \mid x_1 \leq r_1\}$ (switch the inflow to the other tanks as soon as the water there reaches the low level mark);
- $R(q_1, q_2, x) = R(q_2, q_1, x) = \{x\}$ (the continuous state does not change as a result of switching the inflow).

The directed graph corresponding to this hybrid automaton is shown in Figure 3.7. ■

The directed graphs contain exactly the same information as Definition 4.9. They can therefore be treated as informal definitions of hybrid automata. It is common to remove the assignment $x := x$ from an edge of the graph when the continuous state does not change as a result of the discrete transition corresponding to that edge.

3.2.1 Hybrid Time Sets & Executions

Hybrid automata involve both continuous "flow" determined by differential equations and discrete "jumps" determined by a directed graph (like an automaton). To characterise the evolution of the state of a hybrid automaton one has to think of a set of times that contains both continuous intervals (over which continuous evolution takes place) and distinguished discrete points in time, when discrete transitions happen. Such a set of times is called a *hybrid time set*.

Definition 3.2 (Hybrid Time Set) *A hybrid time set is a sequence of intervals* $\tau = \{I_0, I_1, \ldots, I_N\} = \{I_i\}_{i=0}^{N}$*, finite or infinite (i.e.* $N = \infty$ *is allowed) 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*.*

An example of a hybrid time set is given in Figure 3.8. Notice that the right endpoint, τ_i', of the interval I_i coincides with the left endpoint, τ_{i+1} of the interval I_{i+1} (c.f. the time

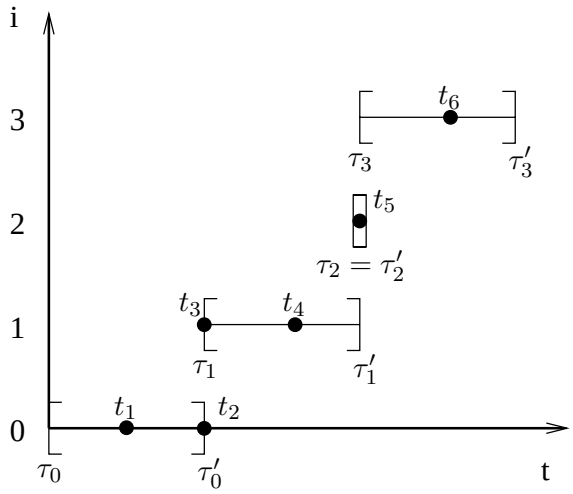


Figure 3.8: A hybrid time set $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{3}$.

instants labelled t_2 and t_3 in Figure 3.8). The interpretation is that these are the times at which discrete transitions of the hybrid system take place. τ_i' corresponds to the time instant just before a discrete transition, whereas τ_{i+1} corresponds to the time instant just after the discrete transition. Discrete transitions are assumed to be instantaneous, therefore $\tau_i' = \tau_{i+1}$. The advantage of this convention is that it allows one to model situations where multiple discrete transitions take place one after the other at the same time instant, in which case $\tau_{i-1}' = \tau_i = \tau_i' = \tau_{i+1}$ (c.f. the interval $I_2 = [\tau_2, \tau_2]'$ in Figure 3.8).

Despite its somewhat complicated nature, a hybrid time set, τ, is a rather well behaved mathematical object. For example, there is a natural way in which the elements of the hybrid time set can be ordered. For $t_1 \in [\tau_i, \tau_i'] \in \tau$ and $t_2 \in [\tau_j, \tau_j'] \in \tau$ we say that t_1 *precedes* t_2 (denoted by $t_1 \prec t_2$) if $t_1 < t_2$ (i.e. if the real number t_1 is less that the real number t_2) or if $i < j$ (i.e. if t_1 belongs to an earlier interval than t_2). In Figure 3.8, we have $t_1 \prec t_2 \prec t_3 \prec t_4 \prec t_5 \prec t_6$. In general, given any two distinct time instants, t_1 and t_2, belonging to some τ we have that either $t_1 \prec t_2$ or $t_2 \prec t_2$ (c.f. given any two distinct real numbers x and y, either $x < y$ or $y < x$). Using mathematical terminology, one would say that each hybrid time set τ is linearly ordered by the relation $\prec$.

Given two hybrid time sets τ and $\hat{\tau}$ there is also a natural way to define if one is "shorter" than the other (τ is called a *prefix* of $\hat{\tau}$ if it is "shorter"). More formally, we say that $\tau = \{I_i\}_{i=0}^{N}$ is a prefix of $\hat{\tau} = \{\hat{I}_i\}_{i=0}^{M}$ (and write $\tau \sqsubseteq \hat{\tau}$) if either they are identical, or τ is a finite sequence, $N \leq M$ (notice that M can be infinite), $I_i = \hat{I}_i$ for all $i = 0, \ldots, N-1$, and $I_N \subseteq \hat{I}_N$. We say that τ is a *strict prefix* of $\hat{\tau}$ (and write $\tau \sqsubset \hat{\tau}$) if $\tau \sqsubseteq \hat{\tau}$ and $\tau \neq \hat{\tau}$. In Figure 3.9, τ is a strict prefix of both $\hat{\tau}$ and $\tilde{\tau}$, but $\hat{\tau}$ is not a prefix of $\tilde{\tau}$ and $\tilde{\tau}$ is not a prefix of $\hat{\tau}$. Notice that given τ and $\hat{\tau}$ we may have neither $\hat{\tau} \sqsubseteq \tau$ nor $\tau \sqsubseteq \hat{\tau}$ (c.f. given two sets of real numbers $A \subseteq \mathbb{R}$ and $B \subseteq \mathbb{R}$ it is possible to have neither $A \subseteq B$ nor $B \subseteq A$). Using mathematical terminology, one would say that the set of all hybrid time sets is partially ordered by the relation $\sqsubseteq$.

Hybrid time sets will be used to define the time horizon over which the states of hybrid systems evolve. What does it mean for the state to "evolve" over a hybrid time set? For continuous systems with state $x \in \mathbb{R}^n$ such an evolution was a function, $x(\cdot) : [0, T] \to \mathbb{R}^n$, mapping a time interval $[0, T]$ to the set $\mathbb{R}^n$ where the state lives (see Chapter **??**). For

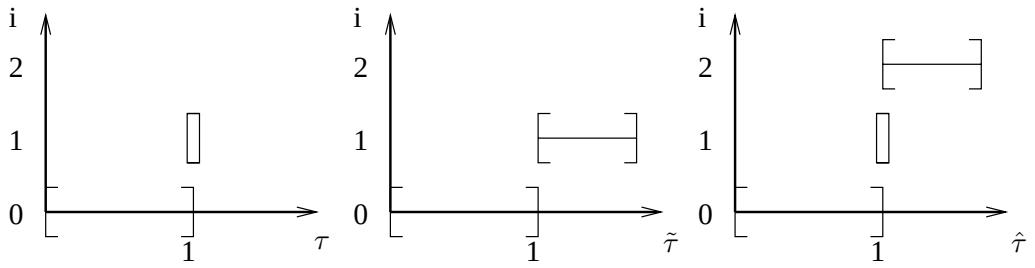


Figure 3.9: $\tau \sqsubset \hat{\tau}$ and $\tau \sqsubset \tilde{\tau}$.

discrete systems (like the manufacturing machine example of Chapter **??**) whose state takes values in a finite set $q \in \{q_1, \ldots, q_n\}$ such an evolution was a sequence of states. For hybrid systems, where the state has both a continuous component $x \in \mathbb{R}^n$ and a discrete component $q \in \{q_1, \ldots, q_n\}$ we need to come up with a mixture of these two notions.

Definition 3.3 (Hybrid Trajectory) *A hybrid trajectory is a triple (τ, q, x) consisting of a hybrid time set $\tau = \{I_i\}_0^N$ and two sequences of functions $q = \{q_i(\cdot)\}_0^N$ and $x = \{x_i(\cdot)\}_0^N$ with $q_i(\cdot) : I_i \to Q$ and $x(\cdot) : I_i \to \mathbb{R}^n$.*

An execution of an autonomous hybrid automaton is a hybrid trajectory, (τ, q, x) of its state variables. The elements listed in Definition 4.9 impose restrictions on the types of hybrid trajectories that the hybrid automaton finds "acceptable".

Definition 3.4 (Execution) *An execution of a hybrid automaton H is a hybrid trajectory, (τ, q, x), which satisfies the following conditions:*

- Initial condition*: $(q_0(0), x_0(0)) \in \mathrm{I}nit$.*
- Discrete evolution*: for all i, $(q_i(\tau_i'), q_{i+1}(\tau_{i+1})) \in E$, $x_i(\tau_i') \in G(q_i(\tau_i'), q_{i+1}(\tau_{i+1}))$, and $x_{i+1}(\tau_{i+1}) \in R(q_i(\tau_i'), q_{i+1}(\tau_{i+1}), x_i(\tau_i'))$.*
- Continuous evolution*: for all i,*

  1. *$q_i(\cdot) : I_i \to Q$ is constant over $t \in I_i$, i.e. $q_i(t) = q_i(\tau_i)$ for all $t \in I_i$;*
  2. *$x_i(\cdot) : I_i \to X$ is the solution to the differential equation*

$$\frac{dx_i}{dt} = f(q_i(t), x_i(t))$$

  over I_i starting at $x_i(\tau_i)$; and,

  3. *for all $t \in [\tau_i, \tau_i')$, $x_i(t) \in \mathrm{D}om(q_i(t))$.*

Definition 3.4 specifies which of the hybrid trajectories are executions of H and which are not by imposing a number of restrictions. The first restriction dictates that the executions should start at an acceptable initial state in Init. For simplicity, we will use $(q_0, x_0) = (q_0(\tau_0), x_0(\tau_0)) \in \mathrm{I}nit$ to denote the initial state of an execution (τ, q, x). As for continuous systems, we can assume that $\tau_0 = 0$ without loss of generality. The second restriction determines when discrete transitions can take place and what the state after discrete transitions can be. The requirements relate the state before the discrete transition

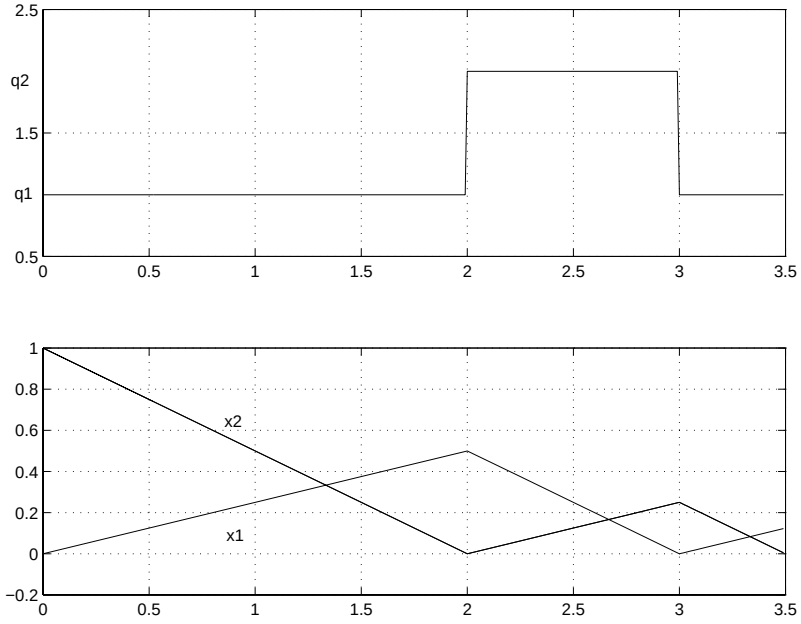


Figure 3.10: Example of an execution of the water tank hybrid automaton.

$(q_i(\tau_i'), x_i(\tau_i'))$ to the state after the discrete transition $(q_{i+1}(\tau_{i+1}), x_{i+1}(\tau_{i+1}))$: they should be such that $(q_i(\tau_i'), q_{i+1}(\tau_{i+1}))$ is an edge of the graph, $x_i(\tau_i')$ belongs to the guard of this edge and $x_{i+1}(\tau_{i+1})$ belongs the the reset map of this edge. In this context, it is convenient to think of the guard $G(e)$ as *enabling* a discrete transition $e \in E$: the execution *may* take a discrete transition $e \in E$ from a state x as long as $x \in G(e)$. The third restriction determines what happens along continuous evolution, and when continuous evolution must give way to a discrete transition. The first part dictates that along continuous evolution the discrete state remains constant. The second part requires that along continuous evolution the continuous state flows according to the differential equation $\dot{x} = f(q, x)$. Notice that the differential equation depends on the discrete state we are currently in (which is constant along continuous evolution). The third part requires that along continuous evolution the state must remain in the domain, D$om(q)$, of the discrete state. In this context, it is convenient to think of D$om(q)$ as *forcing* discrete transitions: the execution *must* take a transition if the state is about to leave the domain.

Example (Water Tank (cont.)) Figure 3.10 shows an execution of the water tank automaton. The hybrid time set τ of the execution consists of three intervals, $\tau = \{[0, 2], [2, 3], [3, 3.5]\}$. The evolution of the discrete state is shown in the upper plot, and the evolution of the continuous state is shown in the lower plot. The values chosen for the constants are $r_1 = r_2 = 0$, $v_1 = v_2 = 1/2$ and $w = 3/4$. The initial state is $q = q_1$, $x_1 = 0$, $x_2 = 1$. ■

A convenient interpretation is that the hybrid automaton *accepts* (as opposed to generates) executions. This perspective allows one to consider, for example, hybrid automata that accept multiple executions for some initial states, a property that can prove very useful when modelling uncertain system (as illustrated by the thermostat example of Chapter **??**).

Definition 3.5 (Classification of executions) *An execution (τ, q, x) is called:*

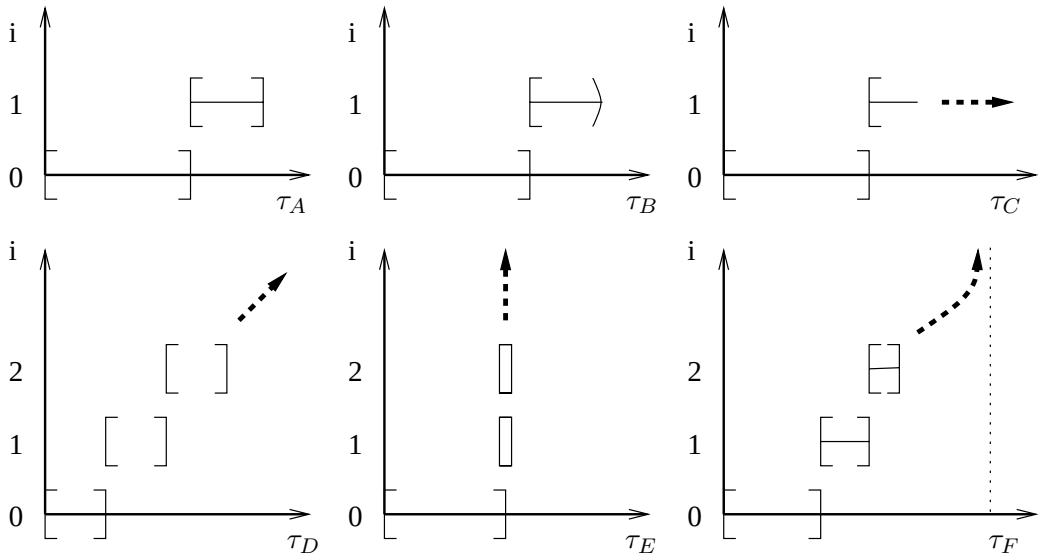


Figure 3.11: τ_A finite, τ_C and τ_D infinite, τ_E and τ_F Zeno.

- **Finite**, *if τ is a finite sequence and the last interval in τ is closed.*
- **Infinite**, *if τ is an infinite sequence, or if the sum of the time intervals in τ is infinite, i.e.*
$$\sum_{i=0}^{N}(\tau_i' - \tau_i) = \infty.$$
- **Zeno**, *if it is infinite but $\sum_{i=0}^{\infty}(\tau_i' - \tau_i) < \infty$.*
- **Maximal** *if it is not a strict prefix of any other execution of H.*

Figure 3.11 shows examples of hybrid time sets of finite, infinite and Zeno executions.

Exercise 3.3 *Show that an execution is Zeno if and only if it takes an infinite number of discrete transitions in a finite amount of time. Does an execution definer over the hybrid time set τ_B of Figure 3.11 belong to any of the classes of Definition 3.5?*

3.3 Bibliography and Further Reading

Hybrid systems arise naturally in a number of engineering applications. In addition to the applications mentioned above, the hybrid paradigm has also been used successfully to address problems in air traffic control [78], automotive control [14], bioengineering [**?**], chemical process control [50, 28], highway systems [82, 39] and manufacturing [69].

The formal definition of hybrid automata is based on a fairly standard class of autonomous hybrid systems. The notation used here comes from [56, 42]. This class of systems has been studied extensively in the literature in a number of variations, for a number of purposes, and by a number of authors. Special cases of the class of systems considered here include switched systems [66], complementarity systems [81], mixed logic dynamic systems [35], and

piecewise linear systems [44] (the autonomous versions of these, to be more precise). The hybrid automata considered here are a special case of the hybrid automata of [9] and the impulse differential inclusions of [12] (discussed in Chapter **??** of these notes), both of which allow differential inclusions to model the continuous dynamics. They are a special case of the General Hybrid Dynamical Systems of [19], which allow the continuous state to take values in manifolds (different ones for each discrete state). They are also a special case of hybrid input/output automata of [59], which, among other things, allow infinite-dimensional continuous state.

3.4 Existence Conditions

We prove some facts about the existence and uniqueness of executions for a restricted class of hybrid systems where the vector field "crosses" the boundary of the invariant set. Some more examples will be given as homework. To eliminate any problems that may arise strictly as a result of continuous evolution we introduce the following assumption: Assume $f(q,x)$ is globally Lipschitz continuous in its second argument.

Definition 3.6 (Reachable State) *A state $(\hat{q},\hat{x}) \in \mathbf{Q}\times\mathbf{X}$ is called reachable by H if there exists a finite execution $\chi = (\tau,q,x)$ with $\tau = \{[\tau_i,\tau_i']\}_{i=0}^{N}$ and $(q(\tau_N'),x(\tau_N')) = (\hat{q},\hat{x})$.*

We use R*each*$(H) \subseteq \mathbf{Q}\times\mathbf{X}$ to denote the set of all states reachable by H.

Definition 3.7 (Non-Blocking and Deterministic Automaton) *A hybrid automaton, H, is called non-blocking if $\mathcal{H}^{\infty}_{(q_0,x_0)}$ is non-empty for all $(q_0,x_0) \in$ I*nit*. A hybrid automaton is called deterministic if $\mathcal{H}^{M}_{(q_0,x_0)}$ contains at most one element for all $(q_0,x_0) \in$ I*nit*.*

In other words:

$$H \text{ is } \textit{non-blocking} \text{ if } \left|\mathcal{H}^{M}_{(q_0,x_0)}\right| \geq 1 \text{ for all } (q_0,x_0) \in \mathrm{I}nit$$

(i.e. there exists at least one infinite execution for every initial condition) and

$$H \text{ is } \textit{deterministic} \text{ if } \left|\mathcal{H}^{\infty}_{(q_0,x_0)}\right| \leq 1 \text{ for all } (q_0,x_0) \in \mathrm{I}nit$$

(i.e. there exists at most one maximal execution for every initial condition)

Assume f is analytic in its second argument. For a function $\sigma : \mathbf{Q}\times\mathbf{X} \to \mathbb{R}$, also analytic in its second argument, inductively define the *Lie derivatives* of σ along f, $L_f^m\sigma : \mathbf{Q}\times\mathbf{X} \to \mathbb{R}$, $m = 0,1,\ldots$ by

$$L_f^0\sigma(q,x) = \sigma(q,x) \quad \text{and} \quad L_f^m\sigma(q,x) = \left(\frac{\partial}{\partial x}L_f^{m-1}\sigma(q,x)\right) f(q,x), \quad \text{for} \quad m > 0.$$

We define the *pointwise relative degree* of σ with respect to f, as the function $n_{(\sigma,f)} : \mathbf{Q}\times\mathbf{X} \to \mathbb{N}$ given by

$$n_{(\sigma,f)}(q,x) := \min\left\{m \in \mathbb{N} :\ L_f^m\sigma(q,x) \neq 0\right\}$$

Note that $n_{(\sigma,f)}(q,x) = 0$ for all (q,x) such that $\sigma(q,x) \neq 0$.

Definition 3.8 (Transverse Invariants) *A hybrid automaton is said to have* transverse invariants *if f is analytic in its second argument and there exists a function $\sigma : \mathbf{Q} \times \mathbf{X} \to \mathbb{R}$, also analytic in its second argument, such that*

- $I(q) = \{x \in \mathbf{X} : \sigma(q, x) \geq 0\}$ *for all* $q \in \mathbf{Q}$*; and*
- *for all* $(q, x) \in \mathbf{Q} \times \mathbf{X}$ *there exists a finite* $m \in \mathbb{N}$ *such that* $L_f^m \sigma(q, x) \neq 0$*, i.e.* $n_{(\sigma,f)}(q, x) < \infty$ *for all* (q, x)*.*

The transverse invariant condition implies that the invariant sets are closed subsets of $\mathbf{X}$.

Example : (Transverse invariants of the bouncing ball)
Recall that for the bouncing ball automaton introduced in Chapter 1, $I(FLY) = \{(x_1, x_2) \in \mathbb{R} : x_1 \geq 0\}$. Define $\sigma : \mathbf{Q} \times \mathbf{X} \to \mathbb{R}$: $\sigma(FLY, (x_1, x_2)) = x_1$. Then

1. f is analytic in x (as it is a polynomial function)
2. σ is analytic in x (as it is also a polynomial function)
3. $I(FLY) = \{(x_1, x_2) \in \mathbb{R} : \sigma(FLY, (x_1, x_2)) \geq 0\}$
4. We check for finiteness of the relative degree of all points:

   If $x_1 \neq 0$, $L_f^0 \sigma(FLY, (x_1, x_2)) \neq 0$, therefore $n_{(\sigma,f)}(q, x) = 0$

   If $x_1 = 0 \wedge x_2 \neq 0$, $L_f^1 \sigma(FLY, (0, x_2)) = \dfrac{\partial \sigma}{\partial x} f = x_2 \neq 0$, therefore $n_{(\sigma,f)}(q, x) = 1$

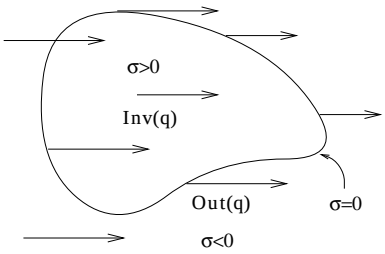
   If $x_1 = 0 \wedge x_2 = 0$, $L_f^2 \sigma(FLY, (0, 0)) = -g \neq 0$, therefore $n_{(\sigma,f)}(q, x) = 2$

Overall, the automaton has transverse invariants.

For an automaton with transverse invariants and for all $q \in \mathbf{Q}$ we also define the set

$$\mathrm{Out}(q) := \left\{ x \in \mathbf{X} : L_f^{n_{(\sigma,f)}(q,x)} \sigma(q, x) < 0 \right\}.$$

Note that $I(q)^c \subseteq \mathrm{Out}(q)$.



Lemma 3.1 *A hybrid automaton with transverse invariants is non-blocking if for all* $q \in \mathbf{Q}$ *and for all* $(q, x) \in \mathrm{R}each(H)$ *with* $x \in \mathrm{Out}(q)$*, there exists* $(q, q') \in E$ *such that*

- $x \in G(q, q')$*; and*
- $R(q, q', x) \neq \emptyset$*.*

Proof: Consider an arbitrary initial state $(q_0, x_0) \in I\mathit{nit}$ and assume, for the sake of contradiction, that there does not exist an infinite execution starting at (q_0, x_0). Let $\chi = (\tau, q, x)$ denote a maximal execution starting at (q_0, x_0), and note that τ is a finite sequence.

First consider the case $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{N-1} [\tau_N, \tau_N')$. Let $(q_N, x_N) = \lim_{t \to \tau_N'} (q(t), x(t))$. Note that, by the definition of execution and a standard existence argument for continuous dynamical systems, the limit exists and χ can be extended to $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \{[\tau_i, \tau_i']\}_{i=0}^{N}$, $\hat{q}(\tau_N') = q_N$, and $\hat{x}(\tau_N') = x_N$. This contradicts the maximality of χ.

Now consider the case $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{N}$, and let $(q_N, x_N) = (q(\tau_N'), x(\tau_N'))$. Clearly, $(q_N, x_N) \in \mathrm{R}each(H)$. If $x_N \notin \mathrm{Out}(q_N)^c = \{x \in \mathbf{X} : L_f^{n_{(\sigma,f)}(q,x)}\sigma(q,x) > 0\}$, then, by the assumption that f and σ are analytic in their second argument, there exists $\epsilon > 0$ such that χ can be extended to $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \tau[\tau_{N+1}, \tau_{N+1}')$, $\tau_{N+1} = \tau_N$, and $\tau_{N+1}' = \tau_{N+1} + \epsilon$, by continuous evolution.

If, on the other hand $x_N \in \mathrm{Out}(q_N)$, then there exists $(q', x') \in \mathbf{Q} \times \mathbf{X}$ such that $(q_N, q') \in E$, $x_N \in G(q_N, q')$ and $x' \in R(q_N, q', x_N)$. Therefore, χ can be extended to $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \{[\tau_i, \tau_i']\}_{i=0}^{N+1}$, $\tau_{N+1} = \tau_{N+1}' = \tau_N$, $q(\tau_{N+1}) = q'$, $x(\tau_{N+1}) = x'$ by a discrete transition. In both cases the maximality of χ is contradicted. ∎

Loosely speaking, the conditions of Lemma 3.3 indicate that a hybrid automaton with transverse invariants is non-blocking if transitions with non-trivial reset relations are enabled along the boundary of the invariant sets, at points where the continuous flow forces the state to exit the invariants. The conditions of Lemma 3.3 are tight, in the sense that blocking automata that violate the conditions exist, but are not necessary, in the sense that not all automata that violate the conditions are blocking.

Lemma 3.2 *A deterministic hybrid automaton with transverse invariants is non-blocking only if the conditions of Lemma 3.3 are satisfied.*

Proof: Consider a deterministic hybrid automaton H that violates the conditions of Lemma 3.3, that is there exists $(q', x') \in \mathrm{R}each(H)$ such that $x \in \mathrm{Out}(q)$, but there is no $\hat{q}' \in \mathbf{Q}$ with $(q', \hat{q}') \in E$, $x' \in G(q', \hat{q}')$ and $R((q', \hat{q}'), x') \neq \emptyset$. Since $(q', x') \in \mathrm{R}each(H)$, there exists $(q_0, x_0) \in \mathit{Init}$ and a finite execution, $\chi = (\tau, q, x) \in \mathcal{H}_{(q_0,x_0)}$ such that $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{N}$ and $(q', x') = (q(\tau_N'), x(\tau_N'))$.

We first show that χ is maximal. Assume first that there exists $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \{[\tau_i, \tau_i']\}_{i=0}^{N-1}[\tau_N, \tau_N + \epsilon$ for some $\epsilon > 0$. This requires that the execution can be extended beyond (q', x') by continuous evolution, which violates the assumption that $x \in \mathrm{Out}(q)$. Next assume that there exists $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \tau[\tau_{N+1}, \tau_{N+1}']$ with $\tau_{N+1} = \tau_N'$. This requires that the execution can be extended beyond (q', x') by a discrete transition, that is there exists $(\hat{q}', \hat{x}') \in \mathbf{Q}$ such that $(q', \hat{q}') \in E$, $x' \in G(q', \hat{q}')$ and $\hat{x}' \in R((q', \hat{q}'), x')$. This also contradicts our original assumptions. Overall, $\chi \in \mathcal{H}_{(q_0,x_0)}^{M}$.

Now assume, for the sake of contradiction that H is non-blocking. Then, there exists $\chi' \in \mathcal{H}_{(q_0,x_0)}^{\infty}$. By definition, $\chi' \in \mathcal{H}_{(q_0,x_0)}^{M}$. But $\chi \neq \chi'$ (as the former is finite and the latter infinite), therefore $\mathcal{H}_{(q_0,x_0)}^{M} \supset \{\chi, \chi'\}$. This contradicts the assumption that H is deterministic. ∎

3.5 Examples of Existence of Executions of Hybrid Systems

3.5.1 Example 1: Lie Derivatives and Relative Degree

- Consider the closed unit disc, S, in $\mathbb{R}^2$:

$$S = \{x \in \mathbb{R}^2 \ : \ x_1^2 + x_2^2 \leq 1\}$$

- Consider a vector field, f, in $\mathbb{R}^2$:

$$f(x) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- We want to classify the set of points in $\mathbb{R}^2$ such that the state can evolve for "a while" along f while staying in S; for short, this will be referred to as whether the state *can* or *can not evolve while staying in* S.

- More formally, consider the execution $x(t)$ of the continuous dynamical system:

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \tag{3.1}$$

  We want to determine the set of all x_0 for which there exists $\epsilon > 0$ such that $x(t) \in S$ for all $t \in [0, \epsilon]$.

- We begin analyzing this by considering three separate sub-sets of the state space, namely:
  - $x_0(t) \notin S$
  - $x_0(t) \in S^o$
  - x on the boundary of S

- **Notation**: S^o denotes the interior of the set S. This is the largest open set contained in S.

- Clearly, when x_0 lies outside of S, we can not evolve while staying in S, since for all $\epsilon > 0$ there exists $t \in [0, \epsilon]$ (in particular $t = 0$) such that $x(t) \notin S$.

- Clearly, when x_0 lies in the interior of S (**not** on the boundary), the state can evolve while staying in S. If $x_0 \in S^o$, then there exists $\epsilon > 0$ such that $x(t) \in S$ for all $t \in [0, \epsilon]$, since S^o is an open set and $x(t)$ is a continuous function of time.

- When x_0 lies on the boundary of S, it is unclear if the state can evolve and remain in S. Our intuition leads us to the following conclusions:
  - if f points into S at x_0, the state should be able to evolve while staying in S.
  - if f points out of S at x_0, the state should not be able to evolve while staying in S.
  - if f is tangential to S at x_0, it is unclear if the state can or can not evolve while staying in S.

- We would like to make this intuitive notion precise mathematically. For this reason we introduce the concepts of the Lie derivative and the relative degree.

- Embed the boundary of S in the level set of a function, $\sigma : \mathbb{R}^2 \to \mathbb{R}$

$$\sigma(x) = 1 - (x_1^2 + x_2^2)$$

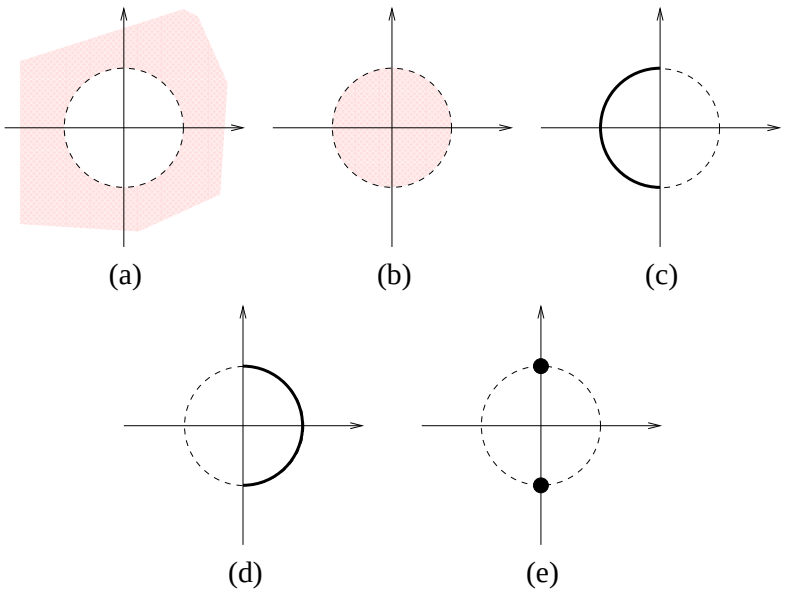


Figure 3.12: The unit disc S

- Using $\sigma(x)$, we note that:
  $\sigma(x) < 0 \iff x \notin S$
  $\sigma(x) > 0 \iff x \in S$
  $\sigma(x) = 0 \iff x$ on the boundary of S

- We can reformulate our conditions on state evolution as follows:

$$\begin{aligned} x \notin S &\iff \{x \in \mathbb{R}^2 \,:\, \sigma(x) < 0\} \rightarrow \text{ the state can not evolve while staying in } S \text{ (F} \\ x \in S^0 &\iff \{x \in \mathbb{R}^2 \,:\, \sigma(x) > 0\} \rightarrow \text{ the state can evolve while staying in } S \text{ Figur} \\ x \text{ on the boundary of } S &\iff \{x \in \mathbb{R}^2 \,:\, \sigma(x) = 0\} \rightarrow ? \end{aligned}$$

- Let us investigate the situation where x is on the boundary of S.

- Pick x_0 such that $\sigma(x_0) = 0$.

- Consider the execution $x(t)$ of (3.1), and in particular value of of $\sigma(x(t))$. Note that $\sigma(x(0)) = \sigma(x_0) = 0$.

- Consider the derivative of $\sigma(x(t))$ with respect to time:

$$\begin{aligned} \frac{d}{dt}\sigma(x(t)) &= \frac{\partial \sigma(x(t))}{\partial x}\dot{x}(t) \\ &= [-2x_1 \quad -2x_2]\begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= -2x_1 \end{aligned}$$

- Define the *Lie derivative of σ along f* by $\mathcal{L}_f\sigma(x) = -2x_1$.

- If $\sigma(x_0) = 0$ but $\mathcal{L}_f\sigma(x_0) > 0$, i.e., if $x_1^2 + x_2^2 = 1$ and $x_1 < 0$, then σ is increasing as time is progressing (i.e., we are moving inside S), therefore we can evolve while staying in S from x_0 (Figure 3.12(c)).
- If $\sigma(x_0) = 0$ but $\mathcal{L}_f\sigma(x_0) < 0$, i.e., if $x_1^2 + x_2^2 = 1$ and $x_1 > 0$, then σ is decreasing as time is progressing (i.e. we are moving outside S), therefore we can not evolve while staying in S from x_0 (Figure 3.12(d)).
- What if $\sigma(x_0) = 0$ and $\mathcal{L}_f\sigma(x_0) = 0$, i.e. if $x_1^2 + x_2^2 = 1$ and $x_1 = 0$?
- Consider the second Lie derivative, i.e.,

$$\frac{d}{dt}\mathcal{L}_f\sigma(x(t)) = \mathcal{L}_f^2\sigma(x(t)) = -2$$

- In this case, we can not evolve while staying in S since $\sigma(x_0) = 0$, $\mathcal{L}_f\sigma(x_0) = 0$, and $\mathcal{L}_f^2\sigma(x_0) < 0$; hence, the state is bound to move outside S (Figure 3.12(e)).
- Overall, the state can not evolve while staying in S from the following points:

$$\begin{aligned}\text{Out} \quad = \quad & \{x_0 \in \mathbb{R}^2 \ : \ \sigma(x_0) < 0\} \cup \\ & \{x_0 \in \mathbb{R}^2 \ : \ \sigma(x_0) = 0 \wedge \mathcal{L}_f\sigma(x_0) < 0\} \cup \\ & \{x_0 \in \mathbb{R}^2 \ : \ \sigma(x_0) = 0 \wedge \mathcal{L}_f\sigma(x_0) = 0 \wedge \mathcal{L}_f^2\sigma(x_0) < 0\} \cup \\ & \dots\end{aligned}$$

- Define the *relative degree* as the function $n : \mathbb{R}^2 \to \mathbb{N}$ by:

$$n(x_0) = \begin{cases} 0 & \text{if } \sigma(x_0) \neq 0 \\ 1 & \text{if } \sigma(x_0) = 0 \wedge \mathcal{L}_f\sigma(x_0) \neq 0 \\ 2 & \text{if } \sigma(x_0) = 0 \wedge \mathcal{L}_f\sigma(x_0) = 0 \\ \wedge \mathcal{L}_f^2\sigma(x_0) \neq 0 & \\ \dots & \end{cases}$$

- Then Out can be written more compactly as:

$$\text{Out} = \{x_0 \in \mathbb{R}^2 \ : \ \mathcal{L}_f^{n(x_0)}\sigma(x_0) < 0\}$$

3.5.2 Analytic Functions

Definition 3.9 (Analytic Function) *A function is analytic if it is infinitely differentiable and the function's Taylor Series converges.*

- Considering example 1, we can see that both $\sigma(x)$ and $f(x)$ are analytic.
- Analytic vector fields are important because they produce executions that are analytic as a function of time.

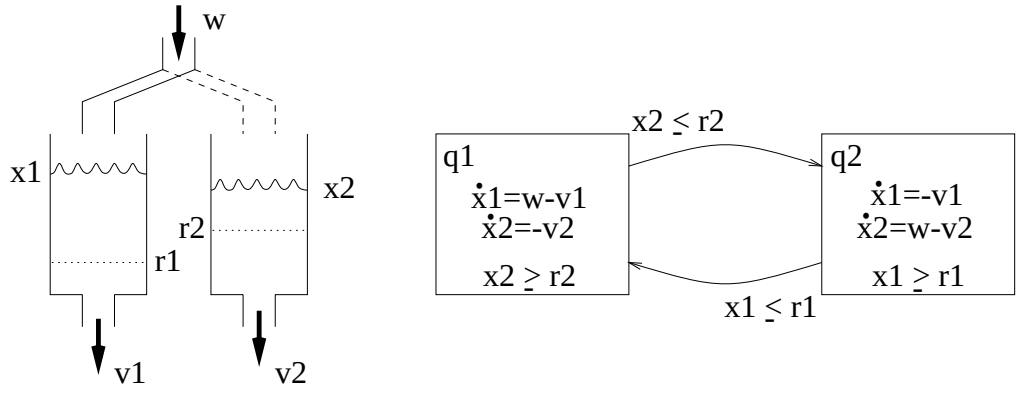


Figure 3.13: The water tank system

- Referring to example 1, $\sigma(x(t))$ is an analytic function of t, since it is the composition of the analytic function σ with the analytic exectution $x(t)$ of the analytic vector field f. The Taylor series expansion around $t = 0$:

  $$\sigma(x(t)) = \sigma(x_0) + \frac{d}{dt}\sigma(x_0)t + \frac{d^2}{dt^2}\sigma(x_0)\frac{t^2}{2!} + \ldots = \sigma(x_0) + \mathcal{L}_f\sigma(x_0)t + \mathcal{L}_f^2\sigma(x_0)\frac{t^2}{2!} + \ldots$$

  suggests why the first non-zero Lie derivative of σ dictates whether we the state can evolve while staying in S or not.

3.6 Example 2: The Water Tank System

Consider the water tank system of [6], shown in Figure 3.15. For $i = 1, 2$, 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 point in time. The control task is to keep the water volumes above r_1 and r_2, respectively (assuming that $x_1(0) > r_1$ and $x_2(0) > r_2$). This is to be achieved by a switched control strategy that switches the inflow to Tank 1 whenever $x_1 \leq r_1$ and to Tank 2 whenever $x_2 \leq r_2$. More formally:

Definition 3.10 (Water Tank Automaton) *The water tank automaton is a hybrid automaton with*

- $\mathbf{Q} = \{q_1, q_2\}$ *and* $\mathbf{X} = \mathbb{R}^2$*;*
- I$nit = \mathbf{Q} \times \{x \in \mathbf{X} : (x_1 > r_1) \wedge (x_2 > r_2)\}$*,* $r_1, r_2 > 0$*;*
- $f(q_1, x) = (w - v_1, -v_2)^T$ *and* $f(q_2, x) = (-v_1, w - v_2)^T$*,* $v_1, v_2, w > 0$*;*
- $I(q_1) = \{x \in \mathbf{X} : x_2 \geq r_2\}$ *and* $I(q_2) = \{x \in \mathbf{X} : x_1 \geq r_1\}$*;*
- $E = \{(q_1, q_2), (q_2, q_1)\}$*;*
- $G(q_1, q_2) = \{x \in \mathbf{X} : x_2 \leq r_2\}$ *and* $G(q_2, q_1) = \{x \in \mathbf{X} : x_1 \leq r_1\}$*; and*
- $R(q_1, q_2, x) = R(q_2, q_1, x) = \{x\}$.

Recall that:

Definition 3.11 (Transverse Invariants) *A hybrid system H has transverse invariants if:*

1. *f is analytic in x,*
2. *$\exists\, \sigma : \mathbf{Q} \times \mathbf{X} \to \mathbb{R}$, analytic in x, such that $Inv(q) = \{x \in \mathbf{X} : \sigma(q,x) \geq 0\}$, and*
3. *$n(q,x) < \infty\ \forall (q,x) \in \mathbf{Q} \times \mathbf{X}$.*

Proposition 3.1 *The water tank automaton has transverse invariants.*

Proof:

1. f is constant for each q; hence f is analytic in x
2. Consider $\sigma(q_1,x) = x_2 - r_2$ and $\sigma(q_2,x) = x_1 - r_1$. Then σ is polynomial (hence analytic) in x and $\sigma(q,x) \geq 0 \iff x \in Inv(q)$
3. if $\sigma(q,x) \neq 0$ then:

$$\sigma(q,x) \neq 0 \iff \begin{cases} q = q_1 \wedge x_2 \neq r_2 \\ q = q_2 \wedge x_1 \neq r_1 \end{cases} \text{ therefore } n(q,x) = 0$$

if $\sigma(q,x) = 0$ then:

$$\begin{aligned} \sigma(q,x) &= 0 \iff (q = q_1 \wedge x_2 = r_2) \vee (q = q_2 \wedge x_1 = r_1) \\ L_f\sigma(q,x) &= \begin{cases} -v_2 & \text{if } q = q_1 \\ -v_1 & \text{if } q = q_2 \end{cases} \text{ therefore } n(q,x) = 1 \end{aligned}$$

In either case, $n(q,x) \leq 1 < \infty$.

■

Recall that:

Lemma 3.3 (Non-Blocking Automata) *A hybrid automaton with transverse invariants is non-blocking if for all $(q,x) \in Reach(H)$ with $x \in Out(q)$ $\exists\ (q,q') \in E$:*

1. *$x \in G(q,q')$*
2. *$R((q,q'),x) \neq \emptyset$*

Proposition 3.2 *The water tank automaton is non-blocking.*

Proof:

- By Proposition 3.1, the water tank automaton has trnasverse invariants, therefore Lemma 3.3 can be applied.

- Out for this example can be written as follows:

$$Out(q) = \begin{cases} \{x_2 < r_2\} \cup \{x_2 = r_2 \wedge -v_2 < 0\} = \{x_2 \leq r_2\} & \text{if } q = q_1 \\ \{x_1 \leq r_1\} & \text{if } q = q_2 \end{cases}$$

- Condition 1 of Lemma 3.3 is satisfied:

$$Out(q) = \begin{cases} \{x_2 \leq r_2\} = G(q_1, q_2) & \text{if } q = q_1 \\ \{x_1 \leq r_1\} = G(q_2, q_1) & \text{if } q = q_2 \end{cases}$$

- Condition 2 of Lemma 3.3 is satisfied:

$$R((q_1, q_2), x) = R((q_2, q_1), x) = \{x\} \neq \emptyset$$

■

Lemma 3.4 *A deterministic hybrid automaton wiuth transverse invariants is non-blocking only if the conditions of Lemma 1 of Lecture 4 are satisfied.*

Proof: Consider a deterministic hybrid automaton H that violates the conditions of Lemma 1, Lecture 4, that is there exists $(q', x') \in \mathrm{R}each(H)$ such that $x \in \mathrm{Out}(q)$, but there is no $\hat{q}' \in \mathbf{Q}$ with $(q', \hat{q}') \in E$, $x' \in G(q', \hat{q}')$ and $R((q', \hat{q}'), x') \neq \emptyset$. Since $(q', x') \in \mathrm{R}each(H)$, there exists $(q_0, x_0) \in \mathrm{I}nit$ and a finite execution, $\chi = (\tau, q, x) \in \mathcal{H}_{(q_0,x_0)}$ such that $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{N}$ and $(q', x') = (q(\tau_N'), x(\tau_N'))$.

We first show that χ is maximal. Assume first that there exists $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \{[\tau_i, \tau_i']\}_{i=0}^{N-1}[\tau_N, \tau_N + \epsilon$ for some $\epsilon > 0$. This requires that the execution can be extended beyond (q', x') by continuous evolution, which violates the assumption that $x \in \mathrm{Out}(q)$. Next assume that there exists $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \tau[\tau_{N+1}, \tau_{N+1}']$ with $\tau_{N+1} = \tau_N'$. This requires that the execution can be extended beyond (q', x') by a discrete transition, that is there exists $(\hat{q}', \hat{x}') \in \mathbf{Q}$ such that $(q', \hat{q}') \in E$, $x' \in G(q', \hat{q}')$ and $\hat{x}' \in R((q', \hat{q}'), x')$. This also contradicts our original assumptions. Overall, $\chi \in \mathcal{H}_{(q_0,x_0)}^{M}$.

Now assume, for the sake of contradiction that H is non-blocking. Then, there exists $\chi' \in \mathcal{H}_{(q_0,x_0)}^{\infty}$. By definition, $\chi' \in \mathcal{H}_{(q_0,x_0)}^{M}$. But $\chi \neq \chi'$ (as the former is finite and the latter infinite), therefore $\mathcal{H}_{(q_0,x_0)}^{M} \supset \{\chi, \chi'\}$. This contradicts the assumption that H is deterministic. ■

3.7 Uniqueness Conditions

Lemma 3.5 *A hybrid automaton with transverse invariants is deterministic if and only if for all $q, q', q'' \in \mathbf{Q}$ and all $(q, x) \in \mathrm{R}each(H)$:*

- *If $x \in \bigcup_{(q,q') \in E} G(q, q')$ then $x \in \mathrm{Out}(q)$;*
- *if $(q, q') \in E$ and $(q, q'') \in E$ with $q' \neq q''$ then $x \notin G(q, q') \cap G(q, q'')$; and*
- *if $(q, q') \in E$ and $x \in G(q, q')$ then $|R(q, q', x)| \leq 1$.*

Proof: For the "if" part, assume, for the sake of contradiction, that there exists an initial state $(q_0, x_0) \in Init$ and two maximal executions $\chi = (\tau, q, x)$ and $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ starting at (q_0, x_0) with $\chi \neq \hat{\chi}$. Let $\psi = (\rho, p, y) \in \mathcal{H}_{(q_0,x_0)}$ denote the maximal common prefix of χ and $\hat{\chi}$. Such a prefix exists as the executions start at the same initial state. Moreover, ψ is not infinite, as $\chi \neq \hat{\chi}$. Therefore, as in the proof of Lemma 1 in Lecture 4, ρ can be assumed to be of the form $\rho = \{[\rho_i, \rho_i']\}_{i=0}^{N}$, as otherwise the maximality of ψ would be contradicted by an existence and uniqueness argument of the continuous solution along f. Let $(q_N, x_N) = (q(\rho_N'), x(\rho_N')) = (\hat{q}(\rho_N'), \hat{x}(\rho_N'))$. Clearly, $(q_N, x_N) \in Reach(H)$. We distinguish the following cases:

Case 1: $\rho_N' \notin \{\tau_i'\}$ and $\rho_N' \notin \{\hat{\tau}_i'\}$, i.e., ρ_N' is not a time when a discrete transition takes place in either χ or $\hat{\chi}$. Then, by the definition of execution and a standard existence and uniqueness argument for continuous dynamical systems, there exists $\epsilon > 0$ such that the prefixes of χ and $\hat{\chi}$ are defined over $\hat{\rho} = \{[\rho_i, \rho_i']\}_{i=0}^{N-1}[\rho_N, \rho_N' + \epsilon)$ and are identical. This contradicts the maximality of ψ.

Case 2: $\rho_N' \in \{\tau_i'\}$ and $\rho_N' \notin \{\hat{\tau}_i'\}$, i.e., ρ_N' is a time when a discrete transition takes place in χ but not in $\hat{\chi}$. The fact that a discrete transition takes place from (q_N, x_N) in χ indicates that there exists $q' \in \mathbf{Q}$ such that $(q_N, q') \in E$ and $x_N \in G(q_N, q')$. The fact that no discrete transition takes place from (q_N, x_N) in $\hat{\chi}$ indicates that there exists $\epsilon > 0$ such that $\hat{\chi}$ is defined over $\hat{\rho} = \{[\rho_i, \rho_i']\}_{i=0}^{N-1}[\rho_N, \rho_N' + \epsilon)$. A necessary condition for this is that $x_N \notin \mathrm{Out}(q)$. This contradicts Condition 1 of the lemma.

Case 3: $\rho_N' \notin \{\tau_i'\}$ and $\rho_N' \in \{\hat{\tau}_i'\}$, symmetric to Case 2.

Case 4: $\rho_N' \in \{\tau_i'\}$ and $\rho_N' \in \{\hat{\tau}_i'\}$, i.e., ρ_N' is a time when a discrete transition takes place in both χ and $\hat{\chi}$. The fact that a discrete transition takes place from (q_N, x_N) in both χ and $\hat{\chi}$ indicates that there exist (q', x') and $(\hat{q}', \hat{x}')$ such that $(q_N, q') \in E$, $(q_N, \hat{q}') \in E$, $x_N \in G(q_N, q')$, $x_N \in G(q_N, \hat{q}')$, $x' \in R(q_N, q', x_N)$, and $\hat{x}' \in R(q_N, \hat{q}', x_N)$. Note that by Condition 2 of the lemma, $q' = \hat{q}'$, hence, by Condition 3, $x' = \hat{x}'$. Therefore, the prefixes of χ and $\hat{\chi}$ are defined over $\hat{\rho} = \{[\rho_i, \rho_i']\}_{i=0}^{N}[\rho_{N+1}, \rho_{N+1}']$, with $\rho_{N+1} = \rho_{N+1}' = \rho_N'$, and are identical. This contradicts the maximality of ψ.

This concludes the proof of the "if" part. For the "only if" part, assume that there exists $(q', x') \in Reach(H)$ such that at least one of the conditions of the lemma is violated. Since $(q', x') \in Reach(H)$, there exists $(q_0, x_0) \in Init$ and a finite execution, $\chi = (\tau, q, x) \in \mathcal{H}_{(q_0,x_0)}$ such that $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{N}$ and $(q', x') = (q(\tau_N'), x(\tau_N'))$. If condition 1 is violated, then there exists $\hat{\chi}$ and $\tilde{\chi}$ with $\hat{\tau} = \{[\tau_i, \tau_i']\}_{i=0}^{N-1}[\tau_N, \tau_N + \epsilon)$, $\epsilon > 0$ and $\tilde{\tau} = \tau[\tau_{N+1}, \tau_{N+1}']$, $\tau_{N+1} = \tau_N'$, such that $\chi < \hat{\chi}$ and $\chi < \tilde{\chi}$. If condition 2 is violated, there exist $\hat{\chi}$ and $\tilde{\chi}$ with $\hat{\tau} = \tilde{\tau} = \tau[\tau_{N+1}, \tau_{N+1}']$, $\tau_{N+1} = \tau_N'$ and $\hat{q}(\tau_{N+1}) \neq \tilde{q}(\tau_{N+1})$, such that $\chi < \hat{\chi}$, $\chi < \tilde{\chi}$. Finally, if condition 3 is violated, then there exist $\hat{\chi}$ and $\tilde{\chi}$ with $\hat{\tau} = \tilde{\tau} = \tau[\tau_{N+1}, \tau_{N+1}']$, $\tau_{N+1} = \tau_N'$ and $\hat{x}(\tau_{N+1}) \neq \tilde{x}(\tau_{N+1})$, such that $\chi < \hat{\chi}$, $\chi < \tilde{\chi}$. In all three cases, let $\overline{\hat{\chi}} \in \mathcal{H}^M_{(q_0,x_0)}$ and $\overline{\tilde{\chi}} \in \mathcal{H}^M_{(q_0,x_0)}$ denote maximal executions of which $\hat{\chi}$ and $\tilde{\chi}$ are prefixes of respectively. Since $\hat{\chi} \neq \tilde{\chi}$, $\overline{\hat{\chi}} \neq \overline{\tilde{\chi}}$, therefore $\left|\mathcal{H}^M_{(q_0,x_0)}\right| \geq 2$, therefore H is non-deterministic. ∎

Loosely speaking, the conditions of Lemma 3.5 indicate that an automaton with transverse invariants is deterministic if and only if discrete transitions have to be forced by the the continuous flow exiting the invariant set, no two discrete transitions can be enabled simultaneously, and no point can be mapped onto two different points by the reset map. Note that the conditions are both necessary and sufficient.

3.8 Summary

Summarizing, Lemma 1 of Lecture 4 and Lemma 3.5 of Lecture 5 give the following:

Theorem 3.1 (Existence and Uniqueness of Executions) *A hybrid automaton with transverse invariants accepts a unique infinite execution for all* $(q_0, x_0) \in \mathrm{I}nit$ *if it satisfies the conditions of Lemma 1 of Lecture 4 and Lemma 3.5 of Lecture 5.*

Proof: If the hybrid automaton satisfies the conditions of Lemma 1 of Lecture 4, then $|\mathcal{H}^{\infty}_{(q_0,x_0)}| \geq 1$, for all $(q_0, x_0) \in \mathrm{I}nit$. If it satisfies the conditions of Lemma 3.5 of Lecture 5, then $|\mathcal{H}^{M}_{(q_0,x_0)}| \leq 1$, for all $(q_0, x_0) \in \mathrm{I}nit$. But every infinite execution is also maximal, therefore $\mathcal{H}^{\infty}_{(q_0,x_0)} \subseteq \mathcal{H}^{M}_{(q_0,x_0)}$. Therefore, $1 \leq |\mathcal{H}^{\infty}_{(q_0,x_0)}| \leq |\mathcal{H}^{M}_{(q_0,x_0)}| \leq 1$, or in other words $|\mathcal{H}^{\infty}_{(q_0,x_0)}| = |\mathcal{H}^{M}_{(q_0,x_0)}| = 1$. ∎

The conditions of the theorem are tight. Similar conditions for the case where the invariant sets are open are given in Problem Set 1.

Notice that the conditions of the lemmas mention the set of reachable states, but do not necessarily require one to compute it explicitly. For the purposes of Theorem 3.1 it suffices to show that the conditions of the lemmas hold in a set of states that contains $\mathrm{R}each(H)$ (for example $\mathbf{Q} \times \mathbf{X}$).

Definition 3.12 (Invariant Set) *A set of states* $S \subseteq \mathbf{Q} \times \mathbf{X}$ *is called invariant if* $\mathrm{R}each(H) \subseteq S$.

Proposition 3.3 *The class of invariant sets is closed under union and intersection.*

Trivially $\mathbf{Q} \times \mathbf{X}$ is an invariant set. More interesting sets are typically shown to be invariant by an induction argument on the length of the system executions.

3.9 Example: The Water Tank System

Consider the water tank system of [6], shown in Figure 3.15. For $i = 1, 2$, 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 point in time. The control task is to keep the water volumes above r_1 and r_2, respectively (assuming that $x_1(0) > r_1$ and $x_2(0) > r_2$). This is to be achieved by a switched control strategy that switches the inflow to Tank 1 whenever $x_1 \leq r_1$ and to Tank 2 whenever $x_2 \leq r_2$. More formally:

Definition 3.13 (Water Tank Automaton) *The water tank automaton is a hybrid automaton with*

- $\mathbf{Q} = \{q_1, q_2\}$ *and* $\mathbf{X} = \mathbb{R}^2$*;*
- $\mathrm{I}nit = \mathbf{Q} \times \{x \in \mathbf{X} : (x_1 > r_1) \wedge (x_2 > r_2)\}$*,* $r_1, r_2 > 0$*;*
- $f(q_1, x) = (w - v_1, -v_2)^T$ *and* $f(q_2, x) = (-v_1, w - v_2)^T$*,* $v_1, v_2, w > 0$*;*

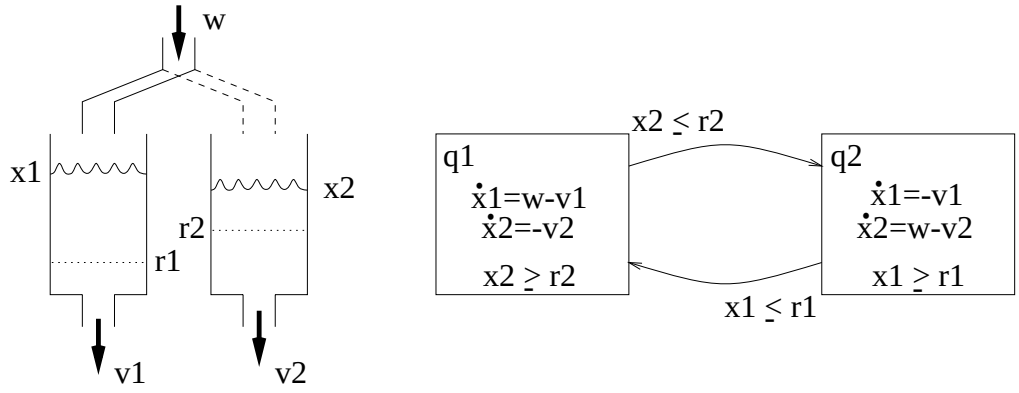


Figure 3.14: The water tank system

- $I(q_1) = \{x \in \mathbf{X} : x_2 \geq r_2\}$ *and* $I(q_2) = \{x \in \mathbf{X} : x_1 \geq r_1\}$*;*
- $E = \{(q_1, q_2), (q_2, q_1)\}$*;*
- $G(q_1, q_2) = \{x \in \mathbf{X} : x_2 \leq r_2\}$ *and* $G(q_2, q_1) = \{x \in \mathbf{X} : x_1 \leq r_1\}$*; and*
- $R(q_1, q_2, x) = R(q_2, q_1, x) = x$*.*

Proposition 3.4 (Existence and Uniqueness of Executions) *The water tank automaton accepts a unique infinite execution for each initial state.*

Proof: Let $\sigma(q_1, x) = x_2 - r_2$ and $\sigma(q_2, x) = x_1 - r_1$. Then $L_f^1\sigma(q_1, x) = -v_2 < 0$ and $L_f^1\sigma(q_2, x) = -v_1 < 0$. Since both f and σ are analytic functions of x and $I(q_i) = \{x \in \mathbf{X} : \sigma(q_i, x) \geq 0\}$, the water tank automaton has transverse invariants.

Note that $n(q_1, x) = 0$ if $x_2 \neq r_2$, and $n(q_1, x) = 1$ if $x_2 = r_2$, therefore $\text{Out}(q_1) = \{x \in \mathbf{X} : x_2 \leq r_2\} = G(q_1, q_2)$ (and similarly for q_2). This implies that Condition 1 of Lemma 1, Lecture 4 and Condition 1 of Lemma 3.5 are satisfied. Moreover, $|R(q_1, q_2, x)| = |R(q_2, q_1, x)| = 1$, therefore Condition 2 of Lemma 1, Lecture 4 and Condition 3 of Lemma 3.5 are satisfied. Condition 2 of Lemma 3.5 is also trivially satisfied. The claim follows by Theorem 3.1. ■

3.10 Zeno Executions

- What does "Zeno" mean? The name Zeno refers to the philosopher Zeno of Elea (500–400 B.C.), whose major work consisted of a number of famous paradoxes. They were designed to explain the view of his mentor, Parmenides, that the ideas of motion and evolving time lead to contradictions. An example is Zeno's Second Paradox of Motion, in which Achilles is racing against a turtle.
- An execution is called Zeno, if it contains an infinite number of transitions in a finite amount of time. An automaton is called Zeno if it accepts a Zeno execution.
- Formally, recall that:

Definition 3.14 (Zeno Hybrid Automaton) *An execution $\chi = (\tau, q, x)$ of a hybrid automaton H is called Zeno if it is infinite, but $\tau_\infty = \sum_i (\tau'_i - \tau_i) < \infty$. A hybrid automaton H is called Zeno, if there exists $(q_0, x_0) \in \mathrm{Init}$ such that $\mathcal{H}_{(q_0,x_0)}$ contains a Zeno execution.*

- τ_∞ is referred to as the *Zeno time*.
- The definition is *existential* (it is enough for one execution to be Zeno for the automaton to be called Zeno).

Example: Consider again the water tank system (Figure 3.15), and recall that it is deterministic and non-blocking. Therefore it accepts a unique infinite execution for each initial condition (q_0, x_0). We show that if

$$\max\{v_1, v_2\} < w < v_1 + v_2$$

then this execution is Zeno. Without loss of generality set $r_1 = r_2 = 0$. Note that all executions reach a state where $q = q_1$, $x_1 = 0$ in finite time (after at most two transitions). Therefore, consider an execution with:

$$\tau_0 = 0, \;\; q(\tau_0) = q_1, \;\; x_1(\tau_0) = 0, \;\; x_2(\tau_0) = h \geq 0$$

The first transition takes place at:

$$\tau'_0 = \tau_1 = \frac{h}{v_2}, \;\; q(\tau_1) = q_2, \;\; x_1(\tau_1) = (w - v_1)\frac{h}{v_2}, \;\; x_2(\tau_1) = 0$$

The second transition takes place at:

$$\tau'_1 = \tau_2 = \tau_1 + \frac{(w - v_1)h}{v_1 v_2}, \;\; q(\tau_2) = q_1, \;\; x_1(\tau_2) = 0, \;\; x_2(\tau_2) = \frac{(w - v_1)(w - v_2)h}{v_1 v_2}$$

The third transition takes place at:

$$\tau'_2 = \tau_3 = \tau_2 + \frac{(w - v_1)(w - v_2)h}{v_1 v_2^2}, \;\; q(\tau_3) = q_2, \;\; x_1(\tau_3) = \frac{(w - v_1)^2(w - v_2)h}{v_1 v_2^2}, \;\; x_2(\tau_3) = 0$$

and so on. Overall,

$$\begin{aligned}
\sum_{i=0}^{\infty}(\tau'_i - \tau_i) &= \frac{h}{v_2} + \frac{(w - v_1)h}{v_1 v_2} + \frac{(w - v_1)(w - v_2)h}{v_1 v_2^2} + \dots \\
&= \frac{h}{v_2}\left[\sum_{i=1}^{\infty}\left(\frac{w - v_1}{v_1}\right)^i\left(\frac{w - v_2}{v_2}\right)^{i-1} + \sum_{i=0}^{\infty}\left(\frac{w - v_1}{v_1}\right)^i\left(\frac{w - v_2}{v_2}\right)^i\right] \\
&= \frac{h}{v_2}\left[\frac{w - v_1}{v_1} + 1\right]\left[\sum_{i=0}^{\infty}\left(\frac{(w - v_1)(w - v_2)}{v_1 v_2}\right)^i\right]
\end{aligned}$$

Since $w < v_1 + v_2$ the infinite sum converges. Therefore:

$$\sum_{i=0}^{\infty}(\tau'_i - \tau_i) = \frac{hw}{v_1 v_2}\left(\frac{1}{1 - \frac{(w - v_1)(w - v_2)}{v_1 v_2}}\right) = \frac{h}{v_1 + v_2 - w}$$

Note that this is the time one would expect the tanks to drain.

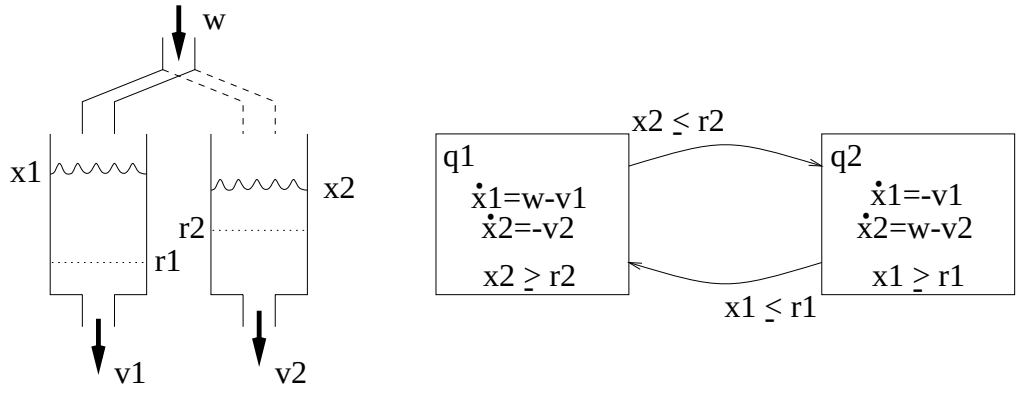


Figure 3.15: The water tank system

3.11 Types of Zeno Phenomena

- Zeno phenomena do not arise in physical systems.
- Zeno phenomena arise as a consequence of modeling over-abstraction.
- They are artifacts of our desire to make the modeling formalism powerful enough to capture a wide class of hybrid phenomena.
- They are a nuisance in more ways than one, as they lead to:
  - **Semantical problems**: How is an execution to be defined beyond the Zeno time?
  - **Analysis problems**: Induction and reachability proofs become suspect. For example, for the water tank system one can show that $x_1 \geq r_1 \wedge x_2 \geq r_2$ along all executions. Clearly, however, if $w < v_1 + v_2$ the tanks will drain in finite time.
  - **Controller Synthesis problems**:
    1. The controller can cheat by forcing time to converge. For example, in the case of the water tank system, the controller appears to succeed in maintaining the water levels above r_1 and r_2.
    2. Some classes of controls, such as relaxed controls and sliding mode controls, are naturally Zeno.
  - **Simulation problems**: Simulation stalls at the Zeno time (refer to lecture by Karl Johansson).
- Classification of the Zeno phenomena:
  1. Mathematical curiosities. Example: non-analytic invariants. Consider the hybrid automaton:
     - $\mathbf{Q} = \{q_1, q_2\}$ and $\mathbf{X} = \mathbb{R}$;
     - $Init = \mathbf{Q} \times \mathbf{X}$;
     - $f(q, x) = 1$ for all $(q, x) \in \mathbf{Q} \times \mathbf{X}$;
     - $I(q_1) = \{x \in \mathbf{X} : e^{-1/|x|} \sin(1/x) \leq 0\}$ and $I(q_2) = \{x \in \mathbf{X} : e^{-1/|x|} \sin(1/x) \geq 0\}$;

- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbf{X} : e^{-1/|x|} \sin(1/x) \geq 0\}$ and $G(q_2, q_1) = \{x \in \mathbf{X} : e^{-1/|x|} \sin(1/x) \leq 0\}$; and
- $R(q_1, q_2, x) = R(q_2, q_1, x) = x$.

The execution of H with initial state $(q_1, -1)$ exhibits an infinite number of discrete transitions by $\tau_\infty = 1$. The reason is that the (non-analytic) function $e^{-1/|x|} \sin(1/x)$ has an infinite number of zeros in the finite interval $(-1, 0)$.

2. Discontinuous vector fields. Example: sliding surfaces. Consider the hybrid automaton:
   - $\mathbf{Q} = \{q_1, q_2\}$ and $\mathbf{X} = \mathbb{R}$;
   - I$nit = \mathbf{Q} \times \mathbf{X}$;
   - $f(q_1, x) = -1$, $f(q_2, x) = 1$;
   - $I(q_1) = \{x \in \mathbf{X} : x \geq 0\}$ and $I(q_2) = \{x \in \mathbf{X} : x \leq 0\}$;
   - $E = \{(q_1, q_2), (q_2, q_1)\}$;
   - $G(q_1, q_2) = \{x \in \mathbf{X} : x \leq 0\}$ and $G(q_2, q_1) = \{x \in \mathbf{X} : x \geq 0\}$; and
   - $R(q_1, q_2, x) = R(q_2, q_1, x) = x$.

   All executions reach $x = 0$ in finite time and and take an infinite number of transitions from then on, without any time progress.
3. Non-zero time progress, continuous state. Example: the water tank system.
4. Non-zero time progress, discontinuous state. Example: the bouncing ball.
5. Non-zero time progress, discontinuous state, no limit. Example: Bouncing ball with a switch. Consider the hybrid automaton:
   - $\mathbf{Q} = \{q\}$ and $\mathbf{X} = \mathbb{R}^3$;
   - I$nit = \{q\} \times \{x \in \mathbf{X} : x_1 \geq 0 \wedge x_3 = 1\}$;
   - $f(q, x) = (x_2, -g, 0)^T$ with $g > 0$;
   - $I(q) = \{x \in \mathbf{X} : x_1 \geq 0\}$;
   - $E = \{(q, q)\}$;
   - $G(q, q) = \{x \in \mathbf{X} : [x_1 < 0] \vee [(x_1 = 0) \wedge (x_2 \leq 0)]\}$; and
   - $R(q, q, x) = (x_1, -x_2/c, -x_3)^T$ with $c > 1$.

   Like the bouncing ball system, this system takes an infinite number of jumps until a finite time τ_∞. However, unlike the bouncing ball, as $t \to \tau_\infty$ the state does not converge (in particular x_3 keeps jumping from 1 to -1 faster and faster).

p

3.12 Resolving the Zeno Phenomenon

he only known conditions to characterize the Zeno phenomenon are fairly trivial

Theorem 3.2 *If* $\mathbf{Q}$ *is finite and* $(\mathbf{Q}, E)$ *is a directed acyclic graph then* H *is not Zeno.*

Proof: If there is no loop in the graph $(\mathbf{Q}, E)$ there can only be a finite number of transitions in each execution. ∎

Theorem 3.3 *Assume there exists a finite collection of states* $\{(q_i, x_i)\}_{i=1,\ldots,N}$ *such that:*

1. $(q_1, x_1) = (q_N, x_N)$
2. *there exists* $i = 1, \ldots, N$, *such that* $(q_i, x_i) \in \mathrm{R}each(H)$
3. *for* $i = 1, \ldots, N-1$, $(q_i, q_{i+1}) \in E$, $x_i \in G(q_i, q_{i+1})$ *and* $x_{i+1} \in R(q_i, q_{i+1}, x_i)$.

Then H *is Zeno.*

Proof: Consider the execution that reaches the reachable (q_i, x_i), and then takes an infinite number of transitions around the loop without allowing time to evolve. ∎

- Some better conditions may be obtained for special cases (e.g. discontinuous vector fields).
- In some cases it may be possible to resolve/avoid the Zeno phenomenon by appropriately redefining the automata or the executions.
  - Most mathematical curiosities can be eliminated by analyticity assumptions.
  - For discontinuous vector fields introduce new states and consider Filippov solutions.
  - Filippov solutions motivated by relaxation. Conditions exist under which relaxation leads to unique Filippov solutions.
  - This is not always the case with hybrid automata. In some cases regularization seems to lead to well defined extensions for the Zeno executions (for example in the case of the bouncing ball, Figures 3.16 and 3.17) while in others the extension seems to depend on the details of the regularization (for example, in the case of the water tank system, Figures 3.18 and 3.19).

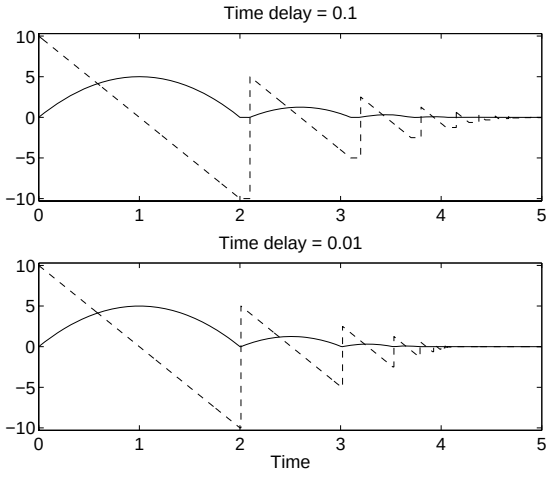


Figure 3.16: Simulation of bouncing ball with temporal regularization. The upper plot corresponds to a time delay during the bounce of $\epsilon = 0.1$ and the lower to $\epsilon = 0.01$.

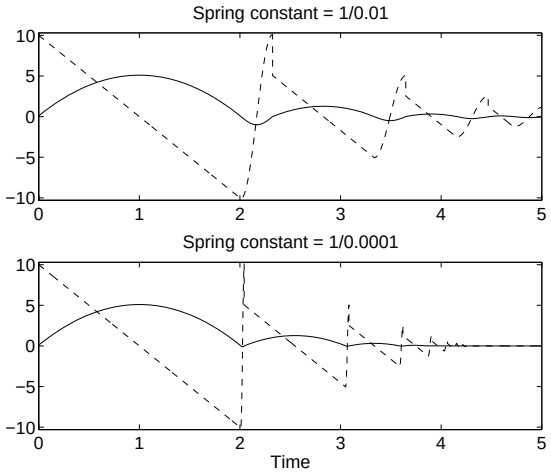


Figure 3.17: Simulation of bouncing ball with dynamical regularization. The upper plot corresponds to spring constant $1/\epsilon = 1/0.01$ and the lower to $1/\epsilon = 0.0001$.

3.13 Automata with Inputs and Outputs

- Up to now: *autonomous hybrid systems*, also known as *closed hybrid systems*, i.e. hybrid automata with no "inputs" or "outputs".
- Good for:
  - Modeling and simulation of small to medium physical systems.
  - Analysis, verifying that all executions satisfy certain desirable properties.
- Limited because:
  - No sense of "control".
  - System must be modeled as a single, monolithic block, which may be cumbersome, wasteful, or impossible. We would like to be able to build up the system description out of the "composition" of smaller pieces. For example, in automated highway model each vehicle individually.
- Introduce *input* and *output* variables, and define appropriate *composition* operation.
- Big topic, we will only scheme the surface here.
- Discussion based on [59, 8], notation changed over to the one used so far in the class.
- **Notation**: consider a collection of variables X, and a subset $X' \subseteq X$. For any valuation $x \in \mathbf{X}$ we define the *restriction* of x to X', $x|_{X'}$, to be a valuation of the variables in X' that agrees with x.
- **Example**: consider $X = \{x_1, x_2, x_3\}$ with $\mathbf{X} = \mathbb{R}^3$. Then $(x_1 = 1, x_2 = 2, x_3 = 3)|_{\{x_1,x_2\}} = (x_1 = 1, x_2 = 2)$.
- Easily extends to sets of valuations.

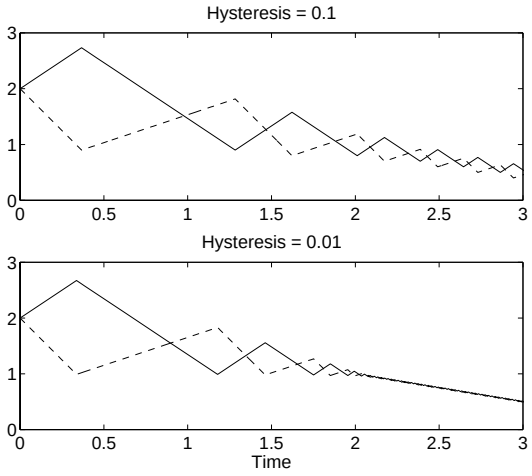


Figure 3.18: Simulation of spatial regularized water tanks. The upper plot corresponds to hysteresis $\epsilon = 0.1$ and the lower to $\epsilon = 0.01$. The solid line is x_1 and the dashed x_2.

3.13.1 Open Hybrid Automata

Definition 3.15 (Open Hybrid Automaton) *An open hybrid automaton H is a collection $H = (Q, X, V, Y, \mathrm{I}nit, f, h, \mathrm{I}nv, E, G, R)$, where*

- *Q is a finite collection of discrete state variables;*
- *X is a finite collection of continuous state variables;*
- *V is a finite collection of input variables. We assume $V = V_D \cup V_C$, where V_D contains discrete and V_C contains continuous variables.*
- *Y is a finite collection of output variables. We assume $Y = Y_D \cup Y_C$, where Y_D contains discrete and Y_C contains continuous variables.*
- *$\mathrm{I}nit \subseteq \mathbf{Q} \times \mathbf{X}$ is a set of initial states;*
- *$f : \mathbf{Q} \times \mathbf{X} \times \mathbf{V} \to \mathbb{R}^n$ is a vector field;*
- *$h : \mathbf{Q} \times \mathbf{X} \to \mathbf{Y}$ is a vector field;*
- *$\mathrm{I}nv : \mathbf{Q} \to 2^{\mathbf{X} \times \mathbf{V}}$ assigns to each $q \in \mathbf{Q}$ an invariant set;*
- *$E \subset \mathbf{Q} \times \mathbf{Q}$ is a collection of discrete transitions;*
- *$G : E \to 2^{\mathbf{X} \times \mathbf{V}}$ assigns to each $e = (q, q') \in E$ a guard; and*
- *$R : E \times \mathbf{X} \times \mathbf{V} \to 2^{\mathbf{X}}$ assigns to each $e = (q, q') \in E$, $x \in \mathbf{X}$ and $v \in \mathbf{V}$ a reset relation.*

Remarks:

1. We refer to $(q, x) \in \mathbf{Q} \times \mathbf{X}$ as the *state of* H, to $v \in \mathbf{V}$ as the input of H and to $y \in \mathbf{Y}$ as the output of H.

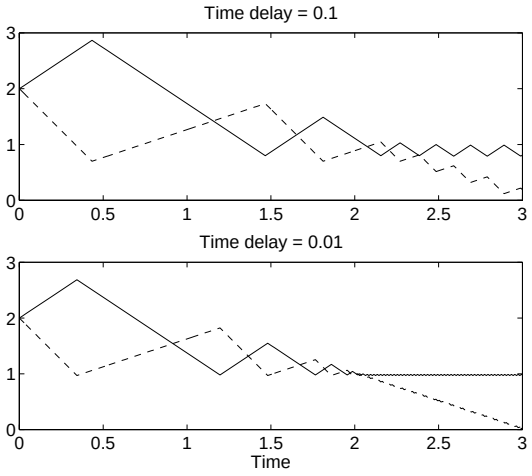


Figure 3.19: Simulation of temporally regularized water tank automaton. The upper plot corresponds to time delay during switching from one tank to the other of $\epsilon = 0.1$ and the lower to $\epsilon = 0.01$. The solid line is x_1 and the dashed x_2.

2. To avoid technicalities with continuous dynamics we impose the following assumption:

Assumption 3.1 *Assume $f(q, x, v)$ and $h(q, x)$ are globally Lipschitz continuous in x and $f(q, x, v)$ is continuous in v.*

Definition 3.16 (Execution) *An execution χ of an open hybrid automaton H is a collection $\chi = (\tau, q, x, v, y)$ with $\tau \in \mathcal{T}$, $q : \tau \to \mathbf{Q}$, $x : \tau \to \mathbf{X}$, $v : \tau \to \mathbf{V}$ and $y : \tau \to \mathbf{Y}$ satisfying:*

- *Initial condition: $(q(\tau_0), x(\tau_0)) \in \mathrm{I}nit$;*
- *Continuous evolution: for all i with $\tau_i < \tau_i'$, q, x, v and y are continuous over $[\tau_i, \tau_i']$ and for all $t \in [\tau_i, \tau_i')$, $(x(t), v(t)) \in \mathrm{I}nv\big(q(t)\big)$ and $\frac{d}{dt}x(t) = f\big(q(t), x(t), v(t)\big)$;*
- *Discrete Evolution: for all i, either $(q(\tau_i'), x(\tau_i')) = (q(\tau_{i+1}), x(\tau_{i+1}))$, or $e_i = \big(q(\tau_i'), q(\tau_{i+1})\big) \in E$, $(x(\tau_i'), v(\tau_i') \in G(e_i)$, and $x(\tau_{i+1}) \in R\big(e_i, x(\tau_i'), v(\tau_i')\big)$; and,*
- *Output Evolution: for all $t \in \tau$, $y(t) = h(q(t), x(t))$.*

Remarks:

1. For an execution $\chi = (\tau, q, x, v, y)$ we use $(q_0, x_0) = \big(q(\tau_0), x(\tau_0)\big)$ to denote the initial state of χ.
2. Two types of transition:
   - Internal transitions, where the state changes.
   - Environmental transitions, taking place "somewhere else" and affecting only the input variables of H (if anything).
3. Control can enter in a number of places. We can use v to:

- Guide the continuous evolution through f.
- Enable internal transitions through G.
- Force internal transitions through Inv.
- Determine the state after a transition through R.

4. Valuation of the input variables in unconstrained:
   - Initially, and
   - After every discrete transition.

   A continuity constraint is imposed during continuous evolution.

5. Possible extensions:
   - Input-Output dependencies: $h : \mathbf{Q} \times \mathbf{X} \times \mathbf{V} \to \mathbf{Y}$. Useful for modeling sensors, actuators, etc. See [8].
   - Set valued output maps: $h : \mathbf{Q} \times \mathbf{X} \to 2^{\mathbf{Y}}$. Useful for modeling nondeterminism.
   - After every discrete transition.
   - State dependent input constraints: $\phi : \mathbf{Q} \times \mathbf{X} \to 2^{\mathbf{V}}$. Useful for controller synthesis, enforcing state constraints.
   - Actions: associated with discrete transitions, can be thought of as discrete variables changing value. Useful for modeling discrete commands, etc. See [59].
   - Synchronous vs. I/O interaction. See [26] or notes of lecture by Tunc Simsek.

Example: (Steam Boiler System, Figure 6.1) Consider the steam boiler system of [37] (originally introduced in [2, 1]). The steam boiler consists of a tank containing water and a heating element that causes the water to boil and escape as steam. The water is replenished by two pumps which at time t pump water into the boiler at rates $u_1(t)$ and $u_2(t)$ respectively. At every time t, pump i can either be on ($u_i(t) = \overline{P}_i$) or off ($u_i(t) = 0$). There is a delay $\overline{T}_i$ between the time pump i is ordered to switch on and the time q_i switches to P_i. There is no delay when the pumps are switched off. We will use three hybrid automata to describe this system, one for the boiler, $B = (Q_B, X_B, V_B, Y_B, \mathrm{I}nit_B, f_B, h_B, \mathrm{I}nv_B, E_B, G_B, R_B)$, and one for each of the pumps, $P_i = (Q_i, X_i, V_i, Y_i, \mathrm{I}nit_i, f_i, h_i, \mathrm{I}nv_i, E_i, G_i, R_i)$.

The boiler automaton is defined by:

- $Q_B = \{q_B\}$, $\mathbf{Q}_B = \{BOILING\}$;
- $X_B = \{w, r\}$, $\mathbf{X}_B = \mathbb{R}^2$;
- $V_B = \{u_1, u_2, d\}$, $\mathbf{V}_B = [0, \overline{P}_1] \times [0, \overline{P}_2] \times [-D_1, D_2]$, where $\overline{P}_1, \overline{P}_2, D_1, D_2 > 0$;
- $Y_B = \{y_1, y_2\}$, $\mathbf{Y}_B = \mathbb{R}^2$;
- I$nit = \{BOILING\} \times [0, W] \times [0, R]$, where $W, R > 0$;
- $$f(BOILING, w, r, u_1, u_2, d) = \begin{bmatrix} \dot{w} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} u_1 + u_2 - r \\ d \end{bmatrix}$$

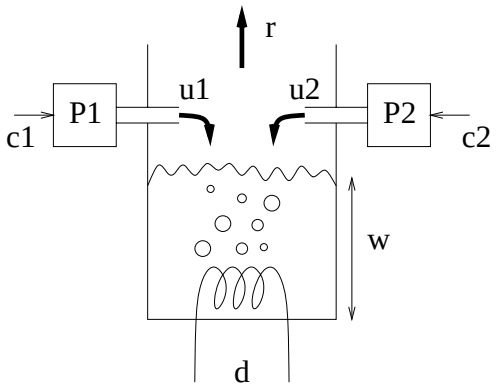


Figure 3.20: The Steam Boiler

- $$h(BOILING, w, r) = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} w \\ r \end{bmatrix}$$
- $\mathrm{I}nv(BOILING) = \mathbf{X}_B \times \mathbf{V}_B$
- $E = \emptyset$

Notice that since $E = \emptyset$, G and R are trivial and need not be explicitly defined.

The automaton for pump i (Figure 6.2) is defined by:

- $Q_i = \{q_i\}$, $\mathbf{Q}_i = \{OFF, GOING\_ON, ON\}$;
- $X_i = \{T_i\}$, $\mathbf{X}_i = \mathbb{R}$;
- $V_i = \{c_i\}$, $\mathbf{V}_i = \{0, 1\}$;
- $Y_i = \{u_i\}$, $\mathbf{Y}_i = [0, \overline{P}_i]$;
- $\mathrm{I}nit = \{OFF\} \times \{0\}$,
- $$f(q_i, T_i, c_i) = \begin{cases} 1 & \text{if } q_i = GOING\_ON \vee q_i = ON \\ 0 & \text{if } q_i = OFF \end{cases}$$
- $$h(q_i, T_i) = \begin{cases} 0 & \text{if } q_i = OFF \vee q_i = GOING\_ON \\ \overline{P}_i & \text{if } q_i = ON \end{cases}$$
- $$\mathrm{I}nv(q_i) = \begin{cases} \mathbf{X}_i \times \{c_i = 0\} & \text{if } q_i = OFF \\ \{T_i \leq \overline{T}_i\} \times \{c_i = 1\} & \text{if } q_i = GOING\_ON \\ \mathbf{X}_i \times \{c_i = 1\} & \text{if } q_i = ON \end{cases}$$
- $E = \{(OFF, GOING\_ON), (GOING\_ON, OFF), (GOING\_ON, ON), (ON, OFF)\}$

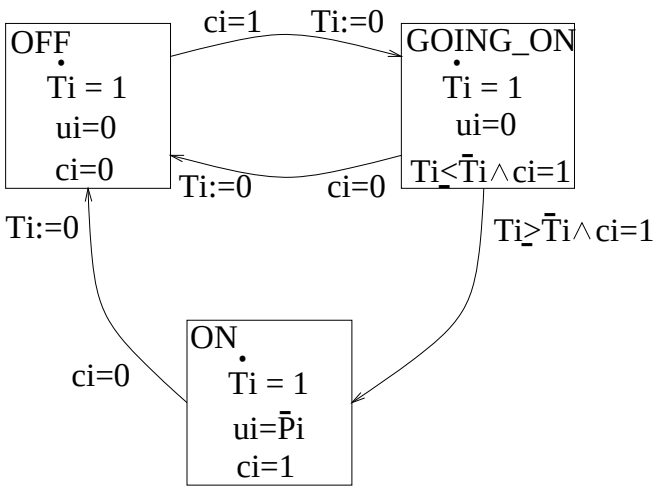


Figure 3.21: The pump hybrid automaton

- $$G(e) = \begin{cases} \mathbf{X}_i \times \{c_i = 1\} & \text{if } e = (OFF, GOING\_ON) \\ \mathbf{X}_i \times \{c_i = 0\} & \text{if } e = (GOING\_ON, OFF) \\ \{T_i \geq \overline{T}_i\} \times \{c_i = 1\} & \text{if } e = (GOING\_ON, ON) \\ \mathbf{X}_i \times \{c_i = 0\} & \text{if } e = (ON, OFF) \end{cases}$$

- $$R(e, T_i, c_i) = \begin{cases} \{0\} & \text{if } e = (OFF, GOING\_ON) \\ \{0\} & \text{if } e = (GOING\_ON, OFF) \\ \{T_i\} & \text{if } e = (GOING\_ON, ON) \\ \{0\} & \text{if } e = (ON, OFF) \end{cases}$$

3.13.2 Traces

- Consider an execution $\chi = (\tau, q, x, v, y) \in \mathcal{H}$ of an open hybrid automaton H. To the "outside world" the only visible part of this execution is (τ, v, y).
- A transition is called *inert* if $(v(\tau_i'), y(\tau_i')) = (v(\tau_{i+1}), y(\tau_{i+1}))$.
- Inert transitions can be eliminated, by concatenating the trajectories either side of the transition.
- More formally, the *concatenation* of two functions:

$$\begin{aligned} (v_1, y_1) &: [\tau_i, \tau_i'] \to \mathbf{V} \times \mathbf{Y} \\ (v_2, y_2) &: [\tau_{i+1}, \tau_{i+1}'] \to \mathbf{V} \times \mathbf{Y} \end{aligned}$$

with $\tau_i' = \tau_{i+1}$ and $(v(\tau_i'), y(\tau_i')) = (v(\tau_{i+1}), y(\tau_{i+1}))$, can be defined as the function

$$(v, y) : [\tau_i, \tau_{i+1}'] \to \mathbf{V} \times \mathbf{Y}$$

such that $(v(t), y(t)) = (v_1(t), y_1(t))$ for all $t \in [\tau_i, \tau_i']$ and $(v(t), y(t)) = (v_2(t), y_2(t))$ for all $t \in [\tau_{i+1}, \tau_{i+1}']$.

- A *trace* of H is a finite or infinite execution of H, restricted to the input and output variables, with all inert transitions removed and the corresponding trajectories concatenated.
- We use Trace(H) to denote the set of all traces of by H.
- Two open hybrid automata H_1 and H_2 are *comparable* if $V_1 = V_2$ and $Y_1 = Y_2$.
- We say an open hybrid automaton H_1 implements an open hybrid automaton H_2 if they are comparable and $\text{Trace}(H_1) \subseteq \text{Trace}(H_2)$.
- Think of H_2 as a specification and of H_1 as a proposed "implementation" of that specification (in software or in hardware).

3.13.3 Composition

Definition 3.17 (Compatible Hybrid Automata) *Two open hybrid automata H_1 and H_2 are called compatible if:*

$$\begin{aligned}(Q_1 \cup X_1) \cap (Q_2 \cup X_2 \cup V_2 \cup Y_2) &= \emptyset \\ (Q_2 \cup X_2) \cap (Q_1 \cup X_1 \cup V_1 \cup Y_1) &= \emptyset \\ Y_1 \cap Y_2 &= \emptyset\end{aligned}$$

For two compatible hybrid automata, H_1 and H_2 define:

$$\begin{aligned}V_1 = V_{11} \cup V_{12} \text{ with } V_{11} = V_1 \setminus Y_2 \text{ and } V_{12} = V_1 \cap Y_2 \\ V_2 = V_{21} \cup V_{22} \text{ with } V_{21} = V_2 \setminus Y_1 \text{ and } V_{22} = V_2 \cap Y_1\end{aligned}$$

For this partition define the maps:

$$\begin{aligned}h_{12} : \mathbf{Q}_1 \times \mathbf{X}_1 &\longrightarrow \mathbf{V}_{21} \\ (q_1, x_1) &\longmapsto h_2(q_1, x_1)|_{V_{21}} \\ &\text{and} \\ h_{21} : \mathbf{Q}_2 \times \mathbf{X}_2 &\longrightarrow \mathbf{V}_{12} \\ (q_2, x_2) &\longmapsto h_2(q_2, x_2)|_{V_{12}}\end{aligned}$$

Definition 3.18 (Composition) *The composition of two compatible hybrid automata H_1 and H_2 is a hybrid automaton $H = H_1 \| H_2 = (Q, X, V, Y, \text{Init}, f, h, \text{Inv}, E, G, R)$, with*

- $Q = Q_1 \cup Q_2$;
- $X = X_1 \cup X_2$;
- $V = (V_1 \cup V_2) \setminus (Y_1 \cup Y_2)$;
- $Y = Y_1 \cup Y_2$;
- $\text{Init} = \{(q, x) \in \mathbf{Q} \times \mathbf{X} \ : \ \left(q|_{Q_i}, x|_{X_i}\right) \in \text{Init}_i, \ \text{for } i = 1, 2\}$;

- $f : \mathbf{Q} \times \mathbf{X} \times \mathbf{V} \to \mathbb{R}^{n_1+n_2}$ *given by:*

$$f(q,x,v) = \begin{bmatrix} f_1\left(q|_{Q_1}, x|_{X_1}, v|_{V_{11}}, h_{21}\left(q|_{Q_2}, x|_{X_2}\right)\right) \\ f_2\left(q|_{Q_2}, x|_{X_2}, v|_{V_{22}}, h_{12}\left(q|_{Q_1}, x|_{X_1}\right)\right) \end{bmatrix}$$

- $h : \mathbf{Q} \times \mathbf{X} \to \mathbf{Y}$ *is a vector field given by:*

$$h(q,x) = \left(h_1\left(q|_{Q_1}, x|_{X_1}\right), h_2\left(q|_{Q_2}, x|_{X_2}\right)\right)$$

- $\mathrm{I}nv : \mathbf{Q} \to 2^{\mathbf{X}\times\mathbf{V}}$ *given by:*

$$\mathrm{I}nv(q) = \left\{(q,x) \in \mathbf{Q} \times \mathbf{X} \ : \ \left(q|_{Q_i}, x|_{X_i}, v|_{V_{ii}}, h_{ji}\left(q|_{Q_j}, x|_{X_2}\right)\right) \in \mathrm{I}nv_i, \ \textit{for } i,j = 1,2, i \neq j\right\}$$

- $E \subset \mathbf{Q} \times \mathbf{Q}$ *given by:*

$$E = \left\{(q,q') \in \mathbf{Q} \times \mathbf{Q} \ : \ \left(q|_{Q_1}, q'|_{Q_1}\right) \in E_1 \vee \left(q|_{Q_2}, q'|_{Q_2}\right) \in E_2\right\}$$

- $G : E \to 2^{\mathbf{X}\times\mathbf{V}}$ *given by:*

$$G(q,q') = \{(x,v) \in \mathbf{X} \times \mathbf{V} \ : \ e_i = \left(q|_{Q_i}, q'|_{Q_i}\right) \in E_i \Rightarrow \left(x|_{X_i}, v|_{V_{ii}}, h_{ji}\left(q|_{Q_j}, x|_{X_2}\right)\right) \in G_i(e_i) \ \textit{for } i,j = 1,2, i \neq j\}$$

- $R : E \times \mathbf{X} \times \mathbf{V} \to 2^{\mathbf{X}}$ *defined by:*

$$R(q,q',x,v) = \{x' \in \mathbf{X} \ : \ e_i = \left(q|_{Q_i}, q'|_{Q_i}\right) \in E_i \Rightarrow x'|_{X_i} \in R_i\left(e_i, x|_{X_i}, v|_{V_{ii}}, h_{ji}\left(q|_{Q_j}, x|_{X_2}\right)\right) \ \textit{for } i,j = 1,2, i \neq j\}$$

Remarks:

1. Both interleaving and synchronous transitions are allowed.
2. A transition is forced for the composition if a transition is forced in at least one of the constituents.
3. A transition is enabled for the composition if a transition is enabled in at least one of the constituents.
4. For an execution $\chi = (\tau, q, x, v, y)$ of the composition $H = H_1 \| H_2$, let $\chi|_{H_i}$ denote the restriction of the execution to the variables of H_i, i.e. the collection $\chi_i = (\tau_i, q_i, x_i, v_i, y_i)$ where $\tau_i \in \mathcal{T}$, $q_i : \tau_i \to \mathbf{Q}_i$, $x_i : \tau_i \to \mathbf{X}_i$, $v_i : \tau_i \to \mathbf{V}_i$ and $y_i : \tau_i \to \mathbf{Y}_i$ defined by $\tau_i = \tau$, and for all $t \in \tau$, $q_i(t) = q(t)|_{Q_i}$, $x_i(t) = x(t)|_{X_i}$, $v_i(t) = v(t)|_{V_i}$, $y_i(t) = y(t)|_{Y_i}$.

Proposition 3.5 *Let $H = H_1 \| H_2$ be the composition of two compatible hybrid automata H_1 and H_2. Then $\mathcal{H} = \{\chi \ : \ \chi|_{H_i} \in \mathcal{H}_i, \ \textit{for } i = 1,2\}$.*

Example: In the steam boiler example, since the only shared variables are u_1 and u_2:

Proposition 3.6 *B, P_1, and P_2 are compatible.*

The composition $H = B \| P_1 \| P_2$ can therefore be defined. It is an open hybrid automaton with:

- $Q = \{q_B, q_1, q_2\}$;
- $X = \{w, r, T_1, T_2\}$;
- $V = \{c_1, c_2, d\}$, with $V = V_C \cup V_D$, $V_C = \{d\}$ and $V_D = \{c_1, c_2\}$;
- $Y_B = \{y_1, y_2, u_1, u_2\}$;
- ...

3.14 Renaming and Hiding

- Somewhat cumbersome to define each variable individually. Renaming allows us to define a single automaton, and then use it in different compositions by changing the name of its variables. This operation does not change the dynamics of the system, only the way it interacts with other systems.
- It may sometimes be desirable to eliminate output variables, especially after it has been "composed" with an input variable of another automaton. This operation does not change the dynamics of the automaton, it just affects its external behavior.
- See [8, 7] for a formal discussion.

3.15 Receptivity

- Zenoness more difficult to define for open hybrid automata.
- *Receptivity*: An open automaton is receptive if it accepts an admissible (time divergent) execution for all input trajectories with a finite number of transitions.
- See [31, 59, 8] for a formal discussion.

3.16 Exercises

Problem 1: Let $x_0, x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, $A \in \mathbb{R}^{n \times n}$ $B \in \mathbb{R}^{n \times m}$, and u a piecewise continuous function of time. Consider the matrix exponential function defined by its Taylor series expansion $e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \ldots$, where I is the identity $n \times n$ matrix.
(a) Consider the autonomous, linear dynamical system:

$$\dot{x}(t) = Ax(t), \quad x(0) = x_0 \tag{3.2}$$

For an arbitrary $T > 0$ show that:

$$x(t) = e^{At} x_0 \tag{3.3}$$

is an execution of (3.2) for $t \in [0, T]$. Can there be another execution, x', with $x'(t) \neq x(t)$ for some $t \in [0, T]$?
(b) Consider the linear control system:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0 \tag{3.4}$$

For an arbitrary $T > 0$ show that:

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \tag{3.5}$$

is an execution of (3.4) for $t \in [0, T]$. Can there be another execution, x', with $x'(t) \neq x(t)$ for some $t \in [0, T]$?
(Hint: Use the Leibniz rule.)

Problem 2: Consider the finite automaton, $M = (\mathbf{Q}, \Sigma, \Delta, q_0, F)$, with $\mathbf{Q} = \{p_0, p_1, p_2\}$, $\mathbf{\Sigma} = \{a, b\}$, $\mathbf{\Delta} = \{(p_0, a, p_1), (p_1, b, p_0), (p_1, b, p_2), (p_2, a, p_0)\}$, $q_0 = \{p_0\}$, $F = \{p_0\}$. Construct an equivalent deterministic finite automaton, M'. Your answer should include a justification of the construction, not just a picture!
(Hint: The fact that if M has $|Q|$ states, M' will in general have $2^{|Q|}$ states suggests a relation between states of M' and sets of states of M.)

Problem 3: Consider the bouncing ball system of Figure 3.22, where $x_1, x_2 \in \mathbb{R}$ and $c \in [0, 1]$. Assume that initially $x_1 \geq 0$.
(a) Write down a model of the system in the hybrid automaton modeling formalism.
(b) Show that the bouncing ball automaton has transverse invariants.
(c) Show that the bouncing ball automaton is non-blocking.
(d) Show that the bouncing ball automaton is deterministic.
(e) Show that, if $c \in [0, 1)$, all execution of the bouncing ball automaton are Zeno.

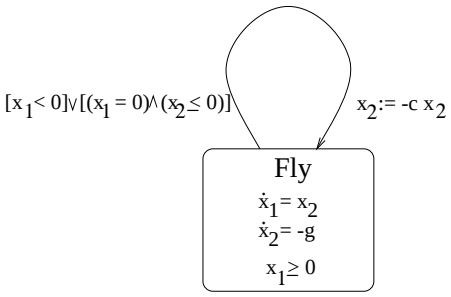


Figure 3.22: Bouncing ball

Problem 4: Consider a hybrid automaton, H, such that the set $\cup_{q\in\mathbf{Q}}(q, I(q))$ is open.
(a) Show that if:

1. for all $q \in \mathbf{Q}$, $I(q)^c \subseteq \cup_{(q,q')\in E} G(q, q')$; and
2. for all $(q, q') \in E$ and for all $x \in G(q, q')$, $R((q, q'), x) \neq \emptyset$,

then H is non-blocking.
(b) Show that if:

1. for all $q \in \mathbf{Q}$, $I(q)^c \supseteq \cup_{(q,q')\in E} G(q, q')$;
2. for all $(q, q'), (q, q'') \in E$ with $q' \neq q''$, $G(q, q') \cap G(q, q'') = \emptyset$; and
3. for all $(q, q') \in E$ and for all $x \in G(q, q')$, $|R((q, q'), x)| \leq 1$,

then H is deterministic.
(c) Deduce an existence and uniqueness theorem for hybrid automata with open invariant sets.

Problem 5: Consider two compatible hybrid automata, H_1 and H_2, and let $H = H_1 \| H_2$ denote their composition. Show that $\mathcal{H} = \{\chi \in \mathrm{Hyb}(\mathrm{Var}(H)) \ : \ \chi|_{\mathrm{Var}(H_i)} \in \mathcal{H}_i, \text{ for } i = 1, 2\}$.

Problem 6: Show that a sequence property (W, P) is both a safety and a liveness property if and only if $P(\chi) = \mathrm{True}$ for all $\chi \in \mathrm{Hyb}(W)$.

Problem 7: Consider a sequence property, (W, P), such that $\{\chi \in \mathrm{Hyb}(W) \ : \ P(\chi) = \mathrm{True}\} \neq \emptyset$. Show that there exist a safety property (W, P_1) and a liveness property (W, P_2) such that $P(\chi) = \mathrm{True}$ if and only if $P_1(\chi) = \mathrm{True}$ and $P_2(\chi) = \mathrm{True}$.

Problem 8: Consider a hybrid automaton, $H = (Q, X, \mathrm{I}nit, f, I, E, G, R)$ with transverse invariants, and a set:

$$\mathrm{I}nv = \{(q, x) \in \mathbf{Q} \times \mathbf{X} \ : \ s(q, x) \geq 0\}$$

where s is a function analytic in x. Let $\mathrm{I}nv_q = \{x \in \mathbf{X} \ : \ (q, x) \in \mathrm{I}nv\}$. Assume that for all $(q, x) \in \mathrm{I}nv$, $n_{(s,f)}(q, x) < \infty$ and let:

$$\mathrm{Out}_{\mathrm{I}nv} = \left\{(q, x) \in \mathrm{I}nv \ : \ L_f^{n_{(s,f)}(q,x)} s(q, x) < 0\right\}$$

Show that if:

1. $\mathrm{I}nit \subseteq \mathrm{I}nv$
2. For all $(q, q') \in E$ and $(q, x) \in \mathrm{R}each(H)$, $x \in G(q, q') \cap \mathrm{I}nv_q \Rightarrow R(q, q', x) \subseteq \mathrm{I}nv_{q'}$.
3. $(q, x) \in \mathrm{Out}_{\mathrm{I}nv} \cap \mathrm{R}each(H) \Rightarrow x \in \mathrm{Out}(q)$.

then H satisfies $(\mathrm{Var}(H), \Box \mathrm{I}nv)$ (or equivalently $\mathrm{R}each(H) \subseteq \mathrm{I}nv$.

Problem 9: Let H be the water tank automaton of Lecture 5. Consider:

$$F = \{x \in \mathbb{R}^2 \ : \ x_1 \geq r_1 \wedge x_2 \geq r_2\}$$

Show that H satisfies $(X, \Box F)$. What does this tell you about the limitations of deductive proofs?

Chapter 4

Executions of Hybrid Systems

4.1 Modelling Issues

Powerful modelling languages, such as hybrid automata, allow one to model a very wide variety of physical phenomena, but also make it possible to produce models that are unreasonable, either physically or mathematically. 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. Such systems are known as *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. For example, the executions of hybrid models can take an infinite number of discrete transitions in finite time. Such executions are known as *Zeno* executions. One can argue that physical systems do not exhibit Zeno behaviour. However, modelling abstraction can often lead to Zeno models of physical systems. Since abstraction is crucial for handling complex systems, understanding when it leads to Zeno hybrid systems is important.

Another issue that arises in hybrid modelling is lack of uniqueness of solutions. Hybrid systems that accept multiple executions for a single initial state are known as *non-deterministic*. It is often desirable to retain some level of non-determinism in a hybrid model, since it can be used to model uncertainty (recall the thermostat example of Chapter **??**). This, however, requires additional care when designing controllers for such systems, or when developing arguments about their performance. A common practice in continuous dynamical systems is to base proofs on arguments about *the solution* of the system. This is motivated by the fact that, under fairly general assumptions (Lipschitz continuity, recall Theorem 2.1), continuous dynamical system have unique solutions. This proof technique is inadequate for non-deterministic systems. Instead one needs to develop arguments that hold for *all solutions* of the system.

Finally, hybrid systems are especially challenging from the point of view of simulation. The problems faced by the developers of simulation algorithms are intimately related to the

modelling problems discussed so far.

- *Existence*: simulation algorithms may run into trouble if the simulated model has no solutions. Incorporating tests for existence in the simulation packages can alleviate this problem. More challenging is the case of Zeno executions. In this case, unless special care is taken, the simulation may grind to a halt, or produce spurious results.

- *Uniqueness*: Non-determinism introduces further complications from the point of view of simulation. 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). Probabilistic methods have also been proposed for dealing with non-determinism in the context of simulation.

- *Discontinuity*: Lack of continuity of the solution with respect to initial conditions, an inherent characteristic of hybrid systems, can also lead to problems, both theoretical and practical. The most common problem is event detection (guard crossing).

- *Composability*: When simulating large scale systems (e.g. the Automated Highway System of Chapter **??**), one would like to be able to build up the simulation by composing different components (e.g. models for the motion of each vehicle). It may also be desirable to add components to the simulation on-line (e.g. to model vehicles joining the highway), eliminate components (e.g. to model vehicles leaving the highway), or redefine the interactions between components (e.g. to model vehicles changing lanes). Object oriented modelling languages have been developed to address these needs.

4.2 Two Fundamental Concepts

Reachability is a fundamental concept in the study of hybrid systems (and dynamical systems in general). Roughly speaking, a state, $(\hat{q}, \hat{x}) \in Q \times X$ of a hybrid automaton H is called reachable if the hybrid automaton can find its way to $(\hat{q}, \hat{x})$ while moving along one of its executions. The importance of the concept of reachability is difficult to overstate. In the next section we will show how reachability plays a central role in the derivation of existence and uniqueness conditions for executions. Reachability will also turn out to be a key concept in the study of safety properties for hybrid systems.

More formally,

Definition 4.1 (Reachable State) *A state $(\hat{q}, \hat{x}) \in Q \times X$ of a hybrid automaton H is called reachable if there exists a finite execution (τ, q, x) ending in $(\hat{q}, \hat{x})$, i.e. $\tau = \{[\tau_i, \tau_i']\}_0^N$, $N < \infty$, and $(q_N(\tau_N'), x_N(\tau_N')) = (\hat{q}, \hat{x})$.*

We will use R$each \subseteq Q \times X$ to denote the set of all states reachable by H. Clearly, I$nit \subseteq$ R$each$.

Exercise 4.1 *Why are all initial states reachable?*

Another important concept in the study of existence of executions for hybrid automata is the set of states from which continuous evolution is impossible. We will call these states *transition states*. For $(\hat{q}, \hat{x}) \in Q \times X$ and some $\epsilon > 0$, consider the solution, $x(\cdot) : [0, \epsilon) \to \mathbb{R}^n$ of the differential equation

$$\frac{dx}{dt} = f(\hat{q}, x) \text{ with } x(0) = \hat{x}. \tag{4.1}$$

Notice that, under the assumption that f is Lipschitz continuous in x, the solution to equation (4.1) exists and is unique (Theorem 2.1). The states, $Trans \subseteq Q \times X$, from which continuous evolution is impossible are

$$Trans = \{(\hat{q}, \hat{x}) \in Q \times X \mid \forall \epsilon > 0 \; \exists t \in [0, \epsilon) \text{ such that } (\hat{q}, x(t)) \notin Dom(\hat{q})\}.$$

In words, $Trans$ is the set of states for which continuous evolution along the differential equation forces the system to exit the domain instantaneously.

The exact characterisation of the set $Trans$ may be quite involved. Clearly, continuous evolution is impossible for all states outside the domain (refer to Definition 3.4). Therefore, for each discrete state $q \in Q$, states in the complement of the domain of q (i.e. the set of all x outside $Dom(q)$, denoted by $Dom(q)^c$) must belong to $Trans$. Mathematically this can be written as

$$\bigcup_{q \in Q} \{q\} \times Dom(q)^c \subseteq Trans$$

If $Dom(q)$ is a closed set (i.e. it contains its boundary), then $Trans$ may also contain pieces of the boundary of the domain. In the examples considered in this class, it is usually straight forward to figure out what these parts are going to be.

Example (Water Tank (continued)) Consider again the water tank automaton, and assume that

$$0 < v_1, v_2 < w.$$

We will show how to compute the sets $Reach$ and $Trans$ for this system.

First of all $Reach$ must contain all initial states. Therefore

$$Reach \supseteq \{q_1, q_2\} \times \{x \in \mathbb{R}^2 \mid (x_1 \geq r_1) \wedge (x_2 \geq r_2)\} \tag{4.2}$$

Can it contain any other states? It turns out that it can not. To see why, we will show using induction that the state remains in the set $Init$.

Consider an arbitrary initial state $(\hat{q}, \hat{x}) \in Init$ and an arbitrary execution (τ, q, x) starting at $(\hat{q}, \hat{x})$. The fact that $(\hat{q}, \hat{x}) \in Init$ provides the base case for the induction argument. Assume that for some i, $(q_i(\tau_i), x_i(\tau_i)) \in Init$, and consider the case where $q_i(\tau_i) = q_1$ (the case $q_i(\tau_i) = q_2$ is similar). If $\tau_i' > \tau_i$, then continuous evolution takes place from $(q_i(\tau_i), x_i(\tau_i))$. Along this evolution, the first component of the continuous state increases (because $q_i(\tau_i) = q_1$, therefore $\dot{x}_1 = w - v_1$ and $v_1 < w$). The second component of the continuous state, on the other hand, decreases, but remains above r_2. This is because, by the definition of an execution,

$$x_i(t) \in Dom(q_1) = \{x \in \mathbb{R}^2 \mid x_2 \geq r_2\}$$

for all $t \in [\tau_i, \tau_i']$. Therefore, $(q_i(t), x_i(t))$ remains in $Init$ along continuous evolution.

If $\tau_i' = \infty$, or if $[\tau_i, \tau_i']$ is the last interval in τ we are done! Otherwise, a discrete transition takes place from $(q_i(\tau_i'), x_i(\tau_i'))$. But the reset relation, R, leaves x unaffected, therefore,

$$(q_{i+1}(\tau_{i+1}), x_{i+1}(\tau_{i+1})) \in \mathrm{I}nit$$

The last statement is true even if $\tau_i = \tau_i'$.

Summarising, if $(q_i(\tau_i), x_i(\tau_i)) \in \mathrm{I}nit$, then $(q_i(t), x_i(t)) \in \mathrm{I}nit$ for all $t \in [\tau_i, \tau_i']$. Moreover, $(q_{i+1}(\tau_{i+1}), x_{i+1}(\tau_{i+1})) \in \mathrm{I}nit$. Therefore, by induction on i, $(q_i(t), x_i(t)) \in \mathrm{I}nit$ for all i and all t and

$$\mathrm{R}each \subseteq \{q_1, q_2\} \times \{x \in \mathbb{R}^2 \mid (x_1 \geq r_1) \wedge (x_2 \geq r_2)\} \tag{4.3}$$

Equations (4.2) and (4.3) together imply that

$$\mathrm{R}each = \{q_1, q_2\} \times \{x \in \mathbb{R}^2 \mid (x_1 \geq r_1) \wedge (x_2 \geq r_2)\}$$

To establish the set $\mathrm{T}rans$ for the water tank system, notice that continuous evolution is impossible if $q = q_1$ and $x_2 < r_2$ (the inflow will get immediately switched to tank 2) or if $q = q_2$ and $x_1 < r_1$. Therefore,

$$\mathrm{T}rans \supseteq \big(\{q_1\} \times \{x \in \mathbb{R}^2 \mid x_2 < r_2\}\big) \cup \big(\{q_2\} \times \{x \in \mathbb{R}^2 \mid x_1 < r_1\}\big)$$

On the other hand, continuous evolution is possible if $q = q_1$ and $x_2 > r_2$, or if $q = q_2$ and $x_1 > r_1$. Therefore

$$\mathrm{T}rans \subseteq \big(\{q_1\} \times \{x \in \mathbb{R}^2 \mid x_2 \leq r_2\}\big) \cup \big(\{q_2\} \times \{x \in \mathbb{R}^2 \mid x_1 \leq r_1\}\big)$$

How about if $q = q_1$ and $x_2 = r_2$? If continuous evolution was to take place from this state, x_2 would immediately go below r_2. This is because $q = q_1$ implies that $\dot{x}_2 = -v_2 < 0$ (recall that $v_2 > 0$). This, however, would imply that the state would leave the domain $\mathrm{D}om(q_1)$, which is impossible along continuous evolution. Therefore, continuous evolution is also impossible from states where $q = q_1$ and $x_2 = r_2$ (and, by a symmetric argument, states where $q = q_2$ and $x_1 = r_1$). Overall,

$$\mathrm{T}rans = \big(\{q_1\} \times \{x \in \mathbb{R}^2 \mid x_2 \leq r_2\}\big) \cup \big(\{q_2\} \times \{x \in \mathbb{R}^2 \mid x_1 \leq r_1\}\big).$$

■

4.3 Local Existence and Uniqueness

Next, we turn our attention to questions of existence of executions. We give some conditions under which infinite executions exist for all initial states, and conditions under which these executions are unique.

Definition 4.2 (Non-Blocking and Deterministic) *A hybrid automaton H is called non-blocking if for all initial states $(\hat{q}, \hat{x}) \in \mathrm{I}nit$ there exists an infinite execution starting at $(\hat{q}, \hat{x})$. It is called deterministic if for all initial states $(\hat{q}, \hat{x}) \in \mathrm{I}nit$ there exists at most one maximal execution starting at $(\hat{q}, \hat{x})$.*

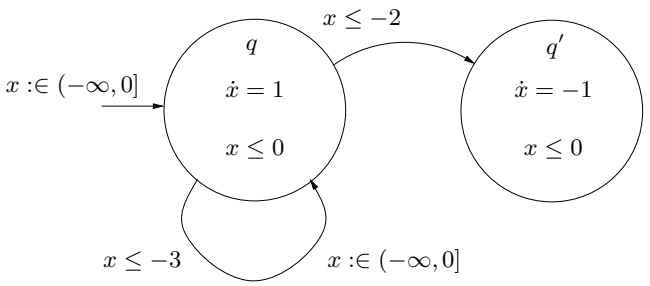


Figure 4.1: Examples of blocking and non-determinism.

Roughly speaking, the non-blocking property implies that infinite executions exist for all initial states, while the deterministic property implies that the infinite executions (if they exist) are unique. As we have seen, continuous dynamical systems described by differential equations have both these properties if the vector field f is assumed to be Lipschitz continuous (Theorem 2.1). In hybrid systems, however, more things can go wrong.

Consider, for example, the hybrid automaton of Figure 4.1. Let $(\hat{q}, \hat{x})$ denote the initial state, and notice that $\hat{q} = q$. If $\hat{x} = -3$, executions starting at $(\hat{q}, \hat{x})$ can either flow along the vector field $\dot{x} = 1$, or jump back to q reseting x anywhere in $(-\infty, 0]$, or jump to q' leaving x unchanged. If $\hat{x} = -2$ executions starting at $(\hat{q}, \hat{x})$ can either flow along the vector field, or jump to q'. If $\hat{x} = -1$ executions stating at $(\hat{q}, \hat{x})$ can only flow along the vector field. Finally, if $\hat{x} = 0$ there are no executions starting at $(\hat{q}, \hat{x})$, other than the trivial execution defined over $[\tau_0, \tau_0']$ with $\tau_0 = \tau_0'$. Therefore, the hybrid automaton of Figure 4.1 accepts no infinite executions for some initial states and multiple infinite executions for others.

Intuitively, a hybrid automaton is non-blocking if for all reachable states for which continuous evolution is impossible a discrete transition is possible. This fact is stated more formally in the following lemma.

Lemma 4.1 *A hybrid automaton, H, is non-blocking if for all $(\hat{q}, \hat{x}) \in \mathrm{R}each \cap \mathrm{T}rans$, there exists $\hat{q}' \in Q$ such that $(\hat{q}, \hat{q}') \in E$ and $\hat{x} \in G(\hat{q}, \hat{q}')$. If H is deterministic, then it is non-blocking if and only if this condition holds.*

Proof: Consider an initial state $(q_0, x_0) \in \mathrm{I}nit$ and assume, for the sake of contradiction, that there does not exist an infinite execution starting at (q_0, x_0). Let $\chi = (\tau, q, x)$ denote a maximal execution starting at (q_0, x_0), and note that τ is a finite sequence.

First consider the case $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{N-1}[\tau_N, \tau_N')$ and let $(q_N, x_N) = \lim_{t \to \tau_N'}(q_N(t), x_N(t))$. Note that, by the definition of execution and a standard existence argument for continuous dynamical systems, the limit exists and χ can be extended to $\widehat{\chi} = (\widehat{\tau}, \widehat{q}, \widehat{x})$ with $\widehat{\tau} = \{[\tau_i, \tau_i']\}_{i=0}^{N}$, $\widehat{q}_N(\tau_N') = q_N$, and $\widehat{x}_N(\tau_N') = x_N$. This contradicts the assumption that χ is maximal.

Now consider the case $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{N}$, and let $(q_N, x_N) = (q_N(\tau_N'), x_N(\tau_N'))$. Clearly, $(q_N, x_N) \in \mathrm{R}each$. If $(q_N, x_N) \notin \mathrm{T}rans$, then there exists $\epsilon > 0$ such that χ can be extended to $\widehat{\chi} = (\widehat{\tau}, \widehat{q}, \widehat{x})$ with $\widehat{\tau} = \{[\tau_i, \tau_i']\}_{i=0}^{N-1}[\tau_N, \tau_N' + \epsilon)$, by continuous evolution. If, on the other hand $(q_N, x_N) \in \mathrm{T}rans$, then by assumption there exists $(q', x') \in Q \times X$ such that $(q_N, q') \in E$, $x_N \in G(q_N, q')$ and $x' \in R(q_N, q', x_N)$. Therefore, χ can be extended to $\widehat{\chi} = (\widehat{\tau}, \widehat{q}, \widehat{x})$ with $\widehat{\tau} = \{[\tau_i, \tau_i']\}_{i=0}^{N+1}$, $\tau_{N+1} = \tau_{N+1}' = \tau_N'$, $q_{N+1}(\tau_{N+1}) = q'$, $x_{N+1}(\tau_{N+1}) = x'$

by a discrete transition. In both cases the assumption that χ is maximal is contradicted.

This argument also establishes the "if" of the second part. For the "only if", consider a deterministic hybrid automaton that violates the conditions, i.e., there exists $(q', x') \in \mathrm{R}each$ such that $(q', x') \in \mathrm{T}rans$, but there is no $\widehat{q}' \in Q$ with $(q', \widehat{q}') \in E$ and $x' \in G(q', \widehat{q}')$. Since $(q', x') \in \mathrm{R}each$, there exists $(q_0, x_0) \in \mathrm{I}nit$ and a finite execution, $\chi = (\tau, q, x)$ starting at (q_0, x_0) such that $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{N}$ and $(q', x') = (q_N(\tau_N'), x_N(\tau_N'))$.

We first show that χ is maximal. Assume first that there exists $\widehat{\chi} = (\widehat{\tau}, \widehat{q}, \widehat{x})$ with $\widehat{\tau} = \{[\tau_i, \tau_i']\}_{i=0}^{N-1}\ [\tau_N, \tau_N' + \epsilon)$ for some $\epsilon > 0$. This would violate the assumption that $(q', x') \in \mathrm{T}rans$. Next assume that there exists $\widehat{\chi} = (\widehat{\tau}, \widehat{q}, \widehat{x})$ with $\widehat{\tau} = \tau[\tau_{N+1}, \tau_{N+1}']$ with $\tau_{N+1} = \tau_N'$. This requires that the execution can be extended beyond (q', x') by a discrete transition, i.e., there exists $(\widehat{q}', \widehat{x}') \in Q \times X$ such that $(q', \widehat{q}') \in E$, $x' \in G(q', \widehat{q}')$ and $\widehat{x}' \in R(q', \widehat{q}', x')$. This would contradict our original assumptions. Overall, χ is maximal.

Now assume, for the sake of contradiction that H is non-blocking. Then, there exists an infinite (and therefore maximal) χ' starting at (q_0, x_0). But $\chi \neq \chi'$ (as the former is finite and the latter infinite). This contradicts the assumption that H is deterministic. ∎

Intuitively, a hybrid automaton may be non-deterministic if either there is a choice between continuous evolution and discrete transition, or if a discrete transition can lead to multiple destinations (recall that continuous evolution is unique by Theorem 2.1). More specifically, the following lemma states that a hybrid automaton is deterministic if and only if (1) whenever a discrete transition is possible continuous evolution is impossible, and (2) discrete transitions have unique destinations.

Lemma 4.2 *A hybrid automaton, H, is deterministic if and only if for all $(\hat{q}, \hat{x}) \in \mathrm{R}each$*

1. *if $\hat{x} \in G(\hat{q}, \hat{q}')$ for some $(\hat{q}, \hat{q}') \in E$, then $(\hat{q}, \hat{x}) \in \mathrm{T}rans$;*
2. *if $(\hat{q}, \hat{q}') \in E$ and $(\hat{q}, \hat{q}'') \in E$ with $\hat{q}' \neq \hat{q}''$ then $\hat{x} \notin G(\hat{q}, \hat{q}') \cap G(\hat{q}, \hat{q}'')$; and,*
3. *if $(\hat{q}, \hat{q}') \in E$ and $x \in G(\hat{q}, \hat{q}')$ then $R(\hat{q}, \hat{q}', \hat{x}) = \{\hat{x}'\}$, i.e. the set contains a single element, $\hat{x}'$.*

Proof: For the "if" part, assume, for the sake of contradiction, that there exists an initial state $(q_0, x_0) \in \mathrm{I}nit$ and two maximal executions $\chi = (\tau, q, x)$ and $\widehat{\chi} = (\widehat{\tau}, \widehat{q}, \widehat{x})$ starting at (q_0, x_0) with $\chi \neq \widehat{\chi}$. Let $\bar{\chi} = (\bar{\tau}, \bar{q}, \bar{x})$ denote the maximal common prefix of χ and $\widehat{\chi}$. Such a prefix exists as the executions start at the same initial state. Moreover, $\bar{\chi}$ is not infinite, as $\chi \neq \widehat{\chi}$. As in the proof of Lemma 4.1, $\bar{\tau}$ can be assumed to be of the form $\bar{\tau} = \{[\bar{\tau}_i, \bar{\tau}_i']\}_{i=0}^{N}$. Let $(q_N, x_N) = (q_N(\bar{\tau}_N'), x_N(\bar{\tau}_N')) = (\widehat{q}_N(\bar{\tau}_N'), \widehat{x}_N(\bar{\tau}_N'))$. Clearly, $(q_N, x_N) \in \mathrm{R}each$. We distinguish the following four cases:

Case 1: $\bar{\tau}_N' \notin \{\tau_i'\}$ and $\bar{\tau}_N' \notin \{\widehat{\tau}_i'\}$, i.e., $\bar{\tau}_N'$ is not a time when a discrete transition takes place in either χ or $\widehat{\chi}$. Then, by the definition of execution and a standard existence and uniqueness argument for continuous dynamical systems, there exists $\epsilon > 0$ such that the prefixes of χ and $\widehat{\chi}$ are defined over $\{[\bar{\tau}_i, \bar{\tau}_i']\}_{i=0}^{N-1}[\bar{\tau}_N, \bar{\tau}_N' + \epsilon)$ and are identical. This contradicts the fact that $\bar{\chi}$ is maximal.

Case 2: $\bar{\tau}_N' \in \{\tau_i'\}$ and $\bar{\tau}_N' \notin \{\widehat{\tau}_i'\}$, i.e., $\bar{\tau}_N'$ is a time when a discrete transition takes place in χ but not in $\widehat{\chi}$. The fact that a discrete transition takes place from (q_N, x_N) in χ

indicates that there exists $q' \in Q$ such that $(q_N, q') \in E$ and $x_N \in G(q_N, q')$. The fact that no discrete transition takes place from (q_N, x_N) in $\widehat{\chi}$ indicates that there exists $\epsilon > 0$ such that $\widehat{\chi}$ is defined over $\{[\bar{\tau}_i, \bar{\tau}'_i]\}_{i=0}^{N-1}[\bar{\tau}_N, \bar{\tau}'_N + \epsilon)$. A necessary condition for this is that $(q_N, x_N) \notin \mathrm{T}rans$. This contradicts condition 1 of the lemma.

Case 3: $\bar{\tau}'_N \notin \{\tau'_i\}$ and $\bar{\tau}'_N \in \{\widehat{\tau}'_i\}$, symmetric to Case 2.

Case 4: $\bar{\tau}'_N \in \{\tau'_i\}$ and $\bar{\tau}'_N \in \{\widehat{\tau}'_i\}$, i.e., $\bar{\tau}'_N$ is a time when a discrete transition takes place in both χ and $\widehat{\chi}$. The fact that a discrete transition takes place from (q_N, x_N) in both χ and $\widehat{\chi}$ indicates that there exist (q', x') and $(\widehat{q}', \widehat{x}')$ such that $(q_N, q') \in E$, $(q_N, \widehat{q}') \in E$, $x_N \in G(q_N, q')$, $x_N \in G(q_N, \widehat{q}')$, $x' \in R(q_N, q', x_N)$, and $\widehat{x}' \in R(q_N, \widehat{q}', x_N)$. Note that by condition 2 of the lemma, $q' = \widehat{q}'$, hence, by condition 3, $x' = \widehat{x}'$. Therefore, the prefixes of χ and $\widehat{\chi}$ are defined over $\{[\bar{\tau}_i, \bar{\tau}'_i]\}_{i=0}^{N}[\bar{\tau}_{N+1}, \bar{\tau}'_{N+1}]$, with $\bar{\tau}_{N+1} = \bar{\tau}'_{N+1} = \bar{\tau}'_N$, and are identical. This contradicts the fact that $\bar{\chi}$ is maximal and concludes the proof of the "if" part.

For the "only if" part, assume that there exists $(q', x') \in \mathrm{R}each$ such that at least one of the conditions of the lemma is violated. Since $(q', x') \in \mathrm{R}each$, there exists $(q_0, x_0) \in \mathrm{I}nit$ and a finite execution, $\chi = (\tau, q, x)$ starting at (q_0, x_0) such that $\tau = \{[\tau_i, \tau'_i]\}_{i=0}^{N}$ and $(q', x') = (q_N(\tau'_N), x_N(\tau'_N))$. If condition 1 is violated, then there exist $\widehat{\chi}$ and $\tilde{\chi}$ with $\widehat{\tau} = \{[\tau_i, \tau'_i]\}_{i=0}^{N-1}[\tau_N, \tau'_N + \epsilon)$, $\epsilon > 0$, and $\tilde{\tau} = \tau[\tau_{N+1}, \tau'_{N+1}]$, $\tau_{N+1} = \tau'_N$, such that $\chi \sqsubset \widehat{\chi}$ and $\chi \sqsubset \tilde{\chi}$. If condition 2 is violated, there exist $\widehat{\chi}$ and $\tilde{\chi}$ with $\widehat{\tau} = \tilde{\tau} = \tau[\tau_{N+1}, \tau'_{N+1}]$, $\tau_{N+1} = \tau'_{N+1} = \tau'_N$, and $\widehat{q}_{N+1}(\tau_{N+1}) \neq \tilde{q}_N(\tau_{N+1})$, such that $\chi \sqsubset \widehat{\chi}$, $\chi \sqsubset \tilde{\chi}$. Finally, if condition 3 is violated, then there exist $\widehat{\chi}$ and $\tilde{\chi}$ with $\widehat{\tau} = \tilde{\tau} = \tau[\tau_{N+1}, \tau'_{N+1}]$, $\tau_{N+1} = \tau'_{N+1} = \tau'_N$, and $\widehat{x}_{N+1}(\tau_{N+1}) \neq \tilde{x}_{N+1}(\tau_{N+1})$, such that $\chi \sqsubset \widehat{\chi}$, $\chi \sqsubset \tilde{\chi}$. In all three cases, let $\overline{\widehat{\chi}}$ and $\overline{\tilde{\chi}}$ denote maximal executions of which $\widehat{\chi}$ and $\tilde{\chi}$ are prefixes, respectively. Since $\widehat{\chi} \neq \tilde{\chi}$, it follows that $\overline{\widehat{\chi}} \neq \overline{\tilde{\chi}}$. Therefore, there are at least two maximal executions starting at (q_0, x_0) and thus H is non-deterministic. ∎

The following theorem is a direct consequence of Lemmas 4.1 and 4.2.

Theorem 4.1 (Existence and Uniqueness) *A hybrid automaton H accepts a unique infinite execution for each initial state if it satisfies all the conditions of Lemmas 4.1 and 4.2.*

Important Note: The conditions of the lemmas involve the set of reachable states, R*each*. This is needed only to make the conditions necessary. If all we are interested in is establishing that a hybrid automaton accepts an infinite executions for all initial states, or that infinite executions are unique, it suffices to show that the conditions of the lemmas hold for all states (as opposed to all *reachable* states). This can make our life considerably easier, since calculating the set of reachable states is sometimes hard.

Example (Water Tank (continued)) Consider again the water tank automaton with $0 < v_1, v_2 < w$. Recall that

$$\mathrm{R}each = \{(q, x) \in \mathbf{Q} \times \mathbb{R}^2 \mid x_1 \geq r_1 \wedge x_2 \geq r_2\},$$
$$\mathrm{T}rans = \{q_1\} \times \{x \in \mathbb{R}^2 \mid x_2 \leq r_2\} \cup \{q_2\} \times \{x \in \mathbb{R}^2 \mid x_1 \leq r_1\}.$$

Therefore,

$$\begin{aligned}\mathrm{R}each \cap \mathrm{T}rans = &\{q_1\} \times \{x \in \mathbb{R}^2 \mid x_1 \geq r_1 \wedge x_2 = r_2\} \cup \\ &\{q_2\} \times \{x \in \mathbb{R}^2 \mid x_2 \geq r_2 \wedge x_1 = r_1\}.\end{aligned}$$

Consider an arbitrary state $(\hat{q}, \hat{x}) \in \mathrm{R}each \cap \mathrm{T}rans$ (in fact the argument holds for any state $(\hat{q}, \hat{x}) \in \mathrm{T}rans$, see "important note" above). Notice that, if $\hat{q} = q_1$, then

$$\hat{x} \in \{x \in \mathbb{R}^2 \mid (x_1 \geq r_1) \wedge (x_2 = r_2)\} \subseteq G(q_1, q_2).$$

Likewise, if $q = q_2$, then $x \in G(q_1, q_2)$. Therefore, the condition of Lemma 4.1 is satisfied, and the water tank system is non-blocking.

Next, consider an arbitrary reachable state $(\hat{q}, \hat{x}) \in \mathrm{R}each$ (in fact the argument holds for any state $(\hat{q}, \hat{x}) \in Q \times X$). Assume that $\hat{q} = q_1$ (a similar argument holds if $\hat{q} = q_2$).

1. If $\hat{x} \in G(q_1, q_2) = \{x \in \mathbb{R}^2 \mid x_2 \leq r_2\}$, then $x_2 \leq r_2$. Therefore $(\hat{q}, \hat{x}) \in \mathrm{T}rans$.
2. Only one discrete transition is possible from q_1 (namely $(q_1, q_2) \in E$).
3. $R(q_1, q_2, \hat{x}) = \{\hat{x}\}$ contains one element.

Therefore, the conditions of Lemma 4.2 are also satisfied. By Theorem 4.1, the water tank automaton accepts a unique infinite execution for each initial state. ∎

4.4 Zeno Executions

The conditions of Theorem 4.1 ensure that a hybrid automaton accepts infinite executions for all initial states. They do not, however, ensure that the automaton accepts executions defined over arbitrarily long time horizons. The Lipschitz assumption on f eliminates the possibility of escape to infinity in finite time along continuous evolution (c.f. finite escape example in Chapter **??**). However, the infinite executions may be such that the state takes an infinite number of discrete transitions in finite time. Executions with this property are known as Zeno executions.

The name "Zeno" comes from the ancient Greek philosopher, Zeno of Elea. Born around 490BC, Zeno was a philosopher and one of the founders of the Eleatic school. He was a student of Parmenides, whose teachings rejected the ideas of plurality and transition as illusions generated by our senses. The main contribution of Zeno was a series of paradoxes designed to support the view of his mentor by showing that accepting plurality and motion leads to logical contradictions. One of the better known ones is the race of Achilles and the turtle.

> Achilles, a renowned runner, was challenged by the turtle to a race. Being a fair sportsman, Achilles decided to give the turtle a 100 meter head-start. To overtake the turtle, Achilles will have to first cover half the distance separating them, i.e. 50 meters. To cover the remaining 50 meters, he will first have to cover half that distance, i.e. 25 meters, and so on. There are an infinite number of such segments and to cover each one of them Achilles needs a non zero amount of time. Therefore, Achilles will never overtake the turtle.

This paradox may seem simple minded, but it was not until the beginning of the 20^{th} century that it was resolved satisfactorily by mathematicians and philosophers. And it was

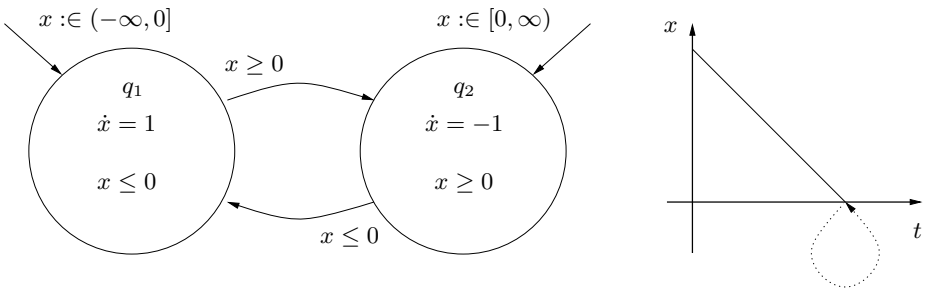


Figure 4.2: Chattering system.

not until the end of the 20^{th} century that it turned out to be a practical problem, in the area of hybrid systems.

The Zeno phenomenon is notoriously difficult to characterise and eliminate in hybrid systems. In this class we will not examine the properties of Zeno executions in detail. We will only give some examples of hybrid automata that admit Zeno behaviour.

Example (Chattering System) Consider the hybrid automaton of Figure 4.2. It is easy to show that this hybrid automaton accepts a unique infinite execution for all initial states. However, all infinite executions are Zeno. An execution starting in x_0 at time τ_0 reaches $x = 0$ in finite time $\tau_0' = \tau_0 + |x_0|$ and takes an infinite number of transitions from then on, without time progressing further.

This is a phenomenon known in continuous dynamical system as *chattering*. A bit of thought in fact reveals that this system is the same as the example used to demonstrate absence of solutions in Chapter **??**. In the control literature the "Zenoness" of such chattering systems is sometimes eliminated by allowing weaker solution concepts, such as sliding solutions (also known as Filippov solutions). A more thorough treatment of this topic can be found in [30, 80]. ■

Example (Non-analytic Domain) Consider the hybrid automaton of Figure 4.3. Assume that the function $\rho : \mathbb{R} \to \mathbb{R}$ that determines the boundary of the domain is of the form

$$\rho(x) = \begin{cases} \sin\left(\frac{1}{x^2}\right) \exp\left(-\frac{1}{x^2}\right) & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases}$$

It is easy to check that the automaton is non-blocking and deterministic.

For any $\epsilon > 0$, ρ has an infinite number of zero crossings in the interval $(-\epsilon, 0]$. Therefore, the execution of the hybrid automaton with initial state (q_1, x_0) will take an infinite number of discrete transitions before time $\tau_0 + |x_0|$ (notice that $x_0 < 0$). ■

Example (Water Tank (continued)) We have already shown that the water tank hybrid automaton accepts a unique infinite execution for each initial state if $0 < v_1, v_2 < w$. If in addition the inflow is less than the sum of the outflows ($w < v_1 + v_2$), then all infinite executions are Zeno. It is easy to show that the execution starting at time τ_0 takes an infinite number of transitions by time

$$\tau_0 + \frac{x_1(\tau_0) + x_2(\tau_0) - r_1 - r_2}{v_1 + v_2 - w}$$

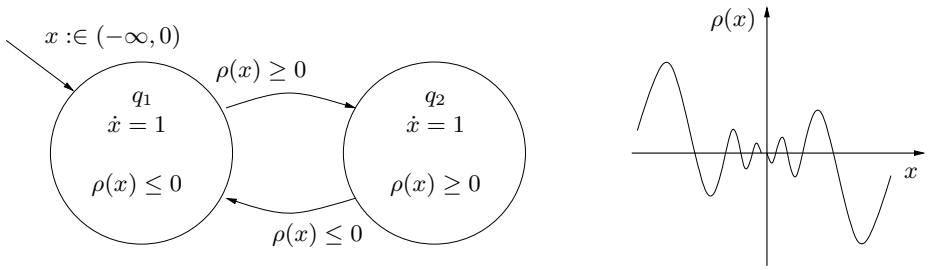


Figure 4.3: System with a smooth, non-analytic domain.

■

4.5 Bibliography and Further Reading

The simulation of hybrid systems presents special challenges, that need particular attention. Nowadays general purpose simulation packages such as Matlab and Simulink can deal adequately with most complications (this was not always the case!) Specialised packages have also been developed that allow accurate simulation of hybrid systems (at least to the extent that this is possible in theory). For references see [11, 64, 63, 10]. See also [27] for a compositional language for hybrid system simulation.

The fundamental properties of existence and uniqueness of solutions, continuity with respect to initial conditions, etc. naturally attracted the attention of researchers in hybrid systems from fairly early on. The majority of the work in this area concentrated on developing conditions for well-posedness (existence and uniqueness of solutions) for special classes of hybrid systems: piecewise linear systems [41, 48], complementarity systems [81, 33], mixed logic dynamical systems [35], etc. The discussion in these notes is based on [56, 55, 42].

Continuity of the solutions with respect to initial conditions and parameters has also been studied, but somewhat less extensively. Motivated by questions of simulation, Tavernini [74] established a class of hybrid systems that have the property of continuous dependence of solutions for almost every initial condition. More recently, an approach to the study of continuous dependence on initial conditions based on the Skorohod topology was proposed [21]. The Skorohod topology, used in stochastic processes for the space of cadlag functions [16], is mathematically appealing, but tends to be cumbersome to work with in practice. [56] presents a more practical (but still limited) approach to the question of continuity.

Zeno executions are treated in [5, 15, 8] from a computer science perspective and [42, 43, 34, 73, 87] from a control theory perspective. [42, 62] attempt to define extensions of Zeno execution beyond the Zeno time, motivated by the classical definition of "sliding" flows for discontinuous vector fields.

Chapter 4

Verification techniques for Hybrid Systems

4.1 Reachability and Sequence Properties

- After a short aside on stability, we return to sequence properties, to study reachability.
- The problem we will address is, given a hybrid automaton H compute R$each(H)$.
- If we can solve this problem we can also answer questions about safety properties.

Proposition 4.1 *H satisfies $(Q \cup X, \Box G)$ for $G \subseteq \mathbf{Q} \times \mathbf{X}$ if and only if* R$each(H) \subseteq G$.

- The interpretation is that G is a "good" set of state you would always like to stay in. Equivalently $B = G^c$ is a "bad" set of states you would like to keep away from.
- Different methods have been proposed for solving the reachability nproblem:
  1. **Optimal Control**: The role of the "control" is often played by the non-determinism of the system.
  2. **Deductive Techniques**: Establish invariants to bound R$each(H)$.
  3. **Model Checking Techniques**: Automatically compute R$each(H)$. Requires one to be able to "compute" with sets of states. Class of systems to which it applies is inherently limited.
  4. **Approximation**: Works with all of the above
     - For optimal control, approximate by sets and dynamics for which optimal control problems are easier to solve (e.g. ellipsoidal sets and linear dynamics)
     - Deductive techniques are inherently based on over-approximation
     - For model checking, approximate by a system you can compute with.
     - Also possible to do "brute force" over-approximation, based, for example, on gridding.
- In all cases you stop once your question has been answered. For example, if your question was "does H satisfy $(Q \cup X, \Box G)$", you stop once you find a reachable state outside G.

- For approximation, you typically "over-approximate".
- All methods are supported by computational tools:
  1. typically uses optimal control and convex optimization tools
  2. typically uses theorem provers
  3. typically uses model checkers
  4. done using "gridding" or polynomial manipulation packages.
- **Simulation** can also be used for reachability investigations. It is not a formal method however since:
  - It is a shot in the dark: we simulate. If B is reached fine, else we have to choose another initial condition and try again.
  - No termination guarantee.

  Still it is the best we can do for many classes of hybrid systems.

4.2 General Transition Systems

Definition 4.1 (Transition System) *A transition system is a collection* $T = (S, \Sigma, \rightarrow, S_0, S_F)$*, where*

- S *is a set of states;*
- Σ *is an alphabet of events;*
- $\rightarrow \subseteq S \times \Sigma \times S$ *is a transition relation;*
- $S_0 \subseteq S$ *is a set of initial states; and,*
- $S_F \subseteq S$ *is a set of final states.*

Example: A finite automaton $M = (\mathbf{Q}, \Sigma, \Delta, q_0, F)$, is a transition system with:

- $S = \mathbf{Q}$
- Σ the same
- $\rightarrow = \Delta$
- $S_0 = \{q_0\}$
- $S_F = F$

Example: A hybrid automaton $H = (Q, X, \mathrm{I}nit, f, I, E, G, R)$ and a safety property $(Q \cup X, \Box G)$ form a transition system with:

- $S = \mathbf{Q} \times \mathbf{X}$

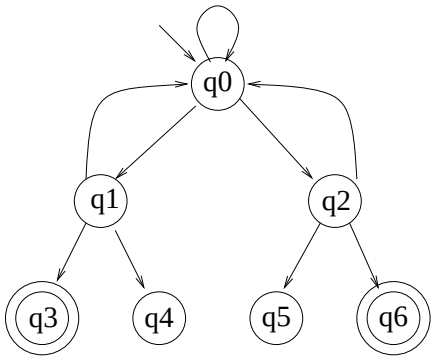


Figure 4.1: Reachability example for finite automata

- $\Sigma = E \cup \{\tau\}$
- $\rightarrow =$ { discrete transitions } $\cup$ { continuous evolution}, all of which are characterized by G, I and R
- $S_0 = \mathrm{I}nit$
- $S_F = G^c$

Problem 4.1 (Reachability) *Given a transition system T, is any state $s_f \in S_F$ reachable from a state $s_0 \in S_0$ by a sequence of transitions?*

Remarks:

- For finite automata we can always "decide" reachability problems by brute force.

Example: Consider for example the finite automaton of Figure 4.1. We can start "exploring" from q_0 and keep going until we either visit a state in F or have nowhere else to go (have visited all reachable states).

- More formally, the set of reachable states for a transition system can be computed using the following "algorithm":

Algorithm 4.0 (Reachability)

Initialization:
$\mathrm{R}each_0 = S_0$
$\mathrm{R}each_{-1} = \emptyset$
$i = 0$
while $\mathrm{R}each_i \neq \mathrm{R}each_{i-1}$ **do**
begin
$\mathrm{R}each_{i+1} = \mathrm{R}each_i \cup \{s' \in S \ : \ \exists s \in \mathrm{R}each_i, \sigma \in \Sigma \text{ with } (s, \sigma, s') \in \rightarrow\}$
$i = i + 1$
end

- If the algorithm can be implemented, it terminates and upon termination $\mathrm{R}each_i \cap S_F = \emptyset$, then the answer to the reachability problem is no.
- For the example of Figure 4.1, $\mathrm{R}each_{-1} = \emptyset$, $\mathrm{R}each_0 = \{q_0\}$, $\mathrm{R}each_1 = \{q_0, q_1, q_2\}$ and $\mathrm{R}each_2 = \mathbf{Q}$.
- For finite automata the algorithm *can be implemented* and *always terminates*.
- What properties of finite automata allow us to do this?
  1. We can represent states in a finite way (by enumeration).
  2. We can represent transitions among states in a finite way (by enumeration)
  3. We are guaranteed that if we start exploring a particular execution, after a finite number of steps we will either reach a state we visited already, or a state from which we have nowhere else to go, or a final state.
- **Remark**: Enumeration is a fairly naive way of going about reachability analysis. In practice, sets of states are not enumerated, but are represented more compactly, e.g. by BDD's.

4.3 Bisimulation

- In the example of Figure 4.1, q_1 and q_2 have very similar properties, since they can both be reached from q_0 by a and all subsequent executions look similar.
- Suggests that these states are in some sense "equivalent".
- Try to make this statement more precise by introducing the notion of bisimulation.
- Consider the set of states S. A relation on S is a subset of $S \times S$.

Definition 4.2 (Equivalence Relation) *A relation $\sim \subseteq S \times S$ is called an equivalence relation if it is:*

1. *Reflexive: $(s, s) \in \sim$ for all $s \in S$;*
2. *Symmetric: $(s, s') \in \sim$ implies that $(s', s) \in \sim$; and,*
3. *Transitive: $(s, s') \in \sim$ and $(s', s'') \in \sim$ imply $(s, s'') \in \sim$.*

- For simplicity we write $s \sim s'$ instead of $(s, s') \in \sim$ and say s is equivalent to s'.
- Examples of equivalence relations: Equality, $S \times S$
- An equivalence relation partitions S to a number of *equivalence classes*:

$$S = \bigcup_i S_i$$

  such that for all $s, s' \in S$, $s, s' \in S_i$ if and only if $s \sim s'$.

- Equivalence classes cover S (by symmetry)
- Equivalence classes are disjoint (by transitivity)
- For equality the equivalence classes are the singletons, for $S \times S$ there is only one equivalence class, S itself.
- Given an equivalence relation $\sim$, let $S/\sim = \{S_i\}$ denote the quotient space, i.e. the set consisting of all equivalence classes.
- Given a set $P \subseteq S$, let $P/\sim$ represent the part of the quotient space with which P overlaps:
$$P/\sim = \{S_i \ : \ S_i \cap P = \emptyset\} \subseteq S/\sim$$
- If S are the states of a transition system, $T = (S, \Sigma, \rightarrow, S_0, S_F)$, define the *quotient transition system* as
$$T/\sim = (S/\sim, \Sigma, \rightarrow_\sim, S_0/\sim, S_F/\sim)$$
where for $S_1, S_2 \in S/\sim$, $(S_1, \sigma, S_2) \in \rightarrow_\sim$ if and only if there exist $s_1 \in S_1$ and $s_2 \in S_2$ such that $(s_1, \sigma, s_2) \in \rightarrow$.
- Notice that the quotient transition system may be "non-deterministic", even if the original system is.
- For $\sigma \in \Sigma$ define the $\mathrm{Pre}_\sigma : 2^S \rightarrow 2^S$ operator as:
$$\mathrm{Pre}_\sigma(P) = \{s \in S \ : \ \exists s' \in P \text{ such that } (s, \sigma, s') \in \rightarrow\}$$

Definition 4.3 (Bisimulation) *Given $T = (S, \Sigma, \rightarrow, S_0, S_F)$, and $\sim$ an equivalence relation over S, $\sim$ is called a bisimulation if:*

1. *S_0 is a union of equivalence classes;*
2. *S_F is a union of equivalence classes;*
3. *For all $\sigma \in \Sigma$, if P is a union of equivalence classes $\mathrm{Pre}_\sigma(P)$ is also a union of equivalence classes.*

- If $\sim$ is a bisimulation, T and $T/\sim$ are called bisimilar.
- Equality is a bisimulation. $S \times S$ is in general not a bisimulation.
- Figure 4.3 shows a bisimulation for the example of Figure 4.1.
- Why is this an improvement?
    1. No need to enumerate all the states, therefore may have a computational advantage.
    2. Extends to systems with infinite states. If the bisimulation quotient can be computed and is finite, then the reachability computation is decidable.

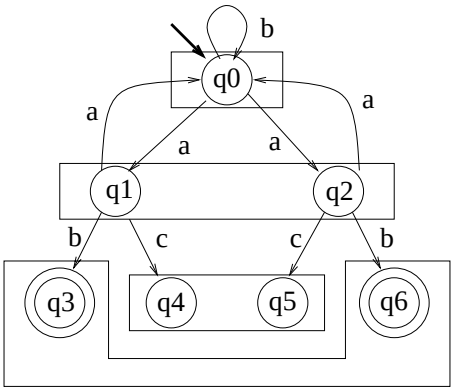


Figure 4.2: Bisimulation of previous example

Definition 4.4 (Bisimulation) *Given $T = (S, \Sigma, \rightarrow, S_0, S_F)$, and $\sim$ an equivalence relation over S, $\sim$ is called a bisimulation if:*

1. *S_0 is a union of equivalence classes;*
2. *S_F is a union of equivalence classes;*
3. *For all $\sigma \in \Sigma$, if P is a union of equivalence classes $\mathrm{Pre}_\sigma(P)$ is also a union of equivalence classes.*

- If $\sim$ is a bisimulation, T and $T/\sim$ are called bisimilar.
- Equality is a bisimulation. $S \times S$ is in general not a bisimulation.
- In a sense bisimilar transition systems generate the same sequences of transitions (language).
- Therefore, if $\sim$ is a bisimulation, we need not distinguish between the elements of an equivalence class.
- More specifically, if a state in S_F is reachable from a state in S_0, a state in $S_F/\sim$ is reachable from a state in $S_F/\sim$.

Proposition 4.2 *$\sim$ is a bisimulation if and only if:*

1. *$(s_1 \sim s_2) \wedge (s_1 \in S_0) \Rightarrow (s_2 \in S_0)$;*
2. *$(s_1 \sim s_2) \wedge (s_1 \in S_F) \Rightarrow (s_2 \in S_F)$; and,*
3. *$(s_1 \sim s_2) \wedge ((s_1, \sigma, s_1') \in \rightarrow) \Rightarrow \exists s_2'$ such that $(s_1' \sim s_2') \wedge ((s_2, \sigma, s_2') \in \rightarrow)$.*

Proof: ($\Rightarrow$):

1. If for all $s_1 \in S_0$, $s_1 \sim s_2$ implies $s_2 \in S_0$, S_0 must be a union of equivalence classes.
2. Similarly for S_F.

3. If P is an equivalence class and $s_1 \in Pre_\sigma(P)$, then $s_2 \in Pre_\sigma(P)$ for all $s_2 \sim s_1$. Hence $Pre_\sigma(P)$ must be a union of equivalence classes.

($\Leftarrow$): similar ∎

Aside: More generally, two transition systems, $T = (S, \Sigma, \rightarrow, S_0, S_F)$ and $T' = (S', \Sigma, \rightarrow', S'_0, S'_F)$ are called *bisimilar* if there exists a relation $\sim \subseteq S \times S'$ such that:

1. $(s_1 \sim s_2) \wedge (s_1 \in S_0) \Rightarrow (s_2 \in S'_0)$;
2. $(s_1 \sim s_2) \wedge (s_2 \in S'_0) \Rightarrow (s_1 \in S_0)$;
3. $(s_1 \sim s_2) \wedge (s_1 \in S_F) \Rightarrow (s_2 \in S'_F)$;
4. $(s_1 \sim s_2) \wedge (s_2 \in S'_F) \Rightarrow (s_1 \in S_F)$;
5. $(s_1 \sim s_2) \wedge ((s_1, \sigma, s'_1) \in \rightarrow) \Rightarrow \exists s'_2$ such that $(s'_1 \sim s'_2) \wedge ((s_2, \sigma, s'_2) \in \rightarrow')$.
6. $(s_1 \sim s_2) \wedge ((s_2, \sigma, s'_2) \in \rightarrow') \Rightarrow \exists s'_1$ such that $(s'_1 \sim s'_2) \wedge ((s_1, \sigma, s'_1) \in \rightarrow)$.

Three of the above condition are taken care of automatically if $T' = T/\sim$. Again T and T' being bisimilar can be interpreted as T and T' accepting the same sequences of events.

4.4 Computing Bisimulations

- Bisimulations look useful since they preserve the language of the transition system.
- How does one find a bisimulation?

Algorithm 4.1 (Bisimulation)

Initialization:
$S/\sim = \{S_0, S_F, S \setminus (S_0 \cup S_F)$
while $\exists P, P' \in S/\sim$ and $\sigma \in \Sigma$ such that $P \cap Pre_\sigma(P') \neq P$ and $P \cap Pre_\sigma(P') \neq \emptyset$ **do**
begin
$P_1 = P \cap Pre_\sigma(P')$
$P_2 = P \setminus Pre_\sigma(P')$
$S/\sim = (S/\sim \setminus \{P\}) \cup \{P_1, P_2\}$
end

- If the algorithm terminates, $\sim$ is a bisimulation, since:
  1. S_0 is a union of equivalence classes (note that $S_0 \in S/\sim$ initially, an we only split sets)
  2. S_F is a union of equivalence classes (for the same reason).
  3. Termination implies that for all $P' \in S/\sim$ and for all σ, $P \cap Pre_\sigma(P')$ is either equal to P or equal to $\emptyset$, therefore, $Pre_\sigma(P')$ is a union of equivalence classes.

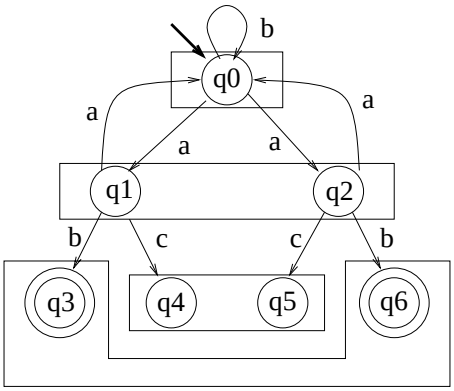


Figure 4.3: Bisimulation of previous example

- This is again a pseudo algorithm. Implementation and termination for general transition systems are not obvious. For finite state systems we can implement the algorithm and guarantee that it terminates because we can enumerate the states for the finite state system.
- Figure 4.3 shows the results of applying this algorithm to the finite state example of Lecture 12.

- Why is this an improvement?
  1. No need to enumerate all the states, therefore may have a computational advantage.
  2. Extends to systems with infinite states. If the bisimulation quotient can be computed and is finite, then the reachability computation is decidable.

4.5 Bisimulations of Timed Automata

- Consider $X = \{x_1, \ldots, x_n\}$ a finite collection of variables, each of which takes values in $\mathbb{R}$.
- Let $x = (x_1, \ldots, x_n) \in Re^n = \mathbf{X}$ denote a valuation for all $x_i \in X$.

Definition 4.5 (Clock Constraints) *The set, $\Phi(X)$, of clock constraints of X, is a set of logical expressions defined inductively by $\delta \in \Phi(X)$ if:*

$$\delta := (x_i \leq c) | (x_i \geq c) | \neg\delta | \delta_1 \wedge \delta_2$$

where $x_i \in X$ and $c \geq 0$ is a rational number.

- Examples: let $X = \{x_1, x_2\}$.
  - $(x_1 \leq 1) \in \Phi(X)$
  - $(0 \leq x_1 \leq 1) \in \Phi(X)$, since $(0 \leq x_1 \leq 1) \Leftrightarrow (x_1 \geq 0) \wedge (x_1 \leq 1)$

- $(x_1 = 1) \in \Phi(X)$, since $(x_1 = 1) \Leftrightarrow (x_1 \geq 1) \wedge (x_1 \leq 1)$
- $(x_1 < 1) \in \Phi(X)$, since $(x_1 < 1) \Leftrightarrow (x_1 \leq 1) \wedge \neg(x_1 \geq 1)$
- True $\in \Phi(X)$, since True $\Leftrightarrow \neg((x_1 = 1) \wedge \neg(x_1 = 1))$
- $(x_1 \leq x_2) \notin \Phi(X)$

- Given $\delta \in \Phi(X)$, we say $x \in \mathbf{X}$ *satisfies* δ is $\delta(x) =$ True.
- To each $\delta \in \Phi(X)$ we can associate a set:

$$\hat{\delta} = \{x \in \mathbf{X} \ : \ \delta(x) = \text{True}\}$$

- The original definition of a timed automaton, found in [4] is given below. Some extensions will be discussed in Lecture 14.

Definition 4.6 (Timed Automaton) *A timed automaton is a hybrid automaton* $H = (Q, X, \text{I}nit, f, I, E, G, R)$*, where*

- Q *is a set of discrete variables,* $\mathbf{Q} = \{q_1, \ldots, q_m\}$*;*
- $X = \{x_1, \ldots, x_n\}$*,* $\mathbf{X} = \mathbb{R}^n$*;*
- $\text{I}nit = \{\{q_i\} \times \widehat{\text{I}nit_{q_i}}\}_{i=1}^m$ *where* $\text{I}nit_{q_i} \in \Phi(X)$*;*
- $f(q, x) = (1, \ldots, 1)$ *for all* (q, x)*;*
- $\text{I}nv(q) = \mathbf{X}$ *for all* $q \in \mathbf{Q}$*;*
- $E \subset \mathbf{Q} \times \mathbf{Q}$*;*
- $G(e) = \widehat{G_e}$ *where* $G_e \in \Phi(X)$*, for all* $e = (q, q') \in E$*; and*
- *For all* e*,* $R(e, x)$ *either leaves* x_i *unaffected or resets it to* 0*.*

- Notice that R is single valued.
- Example
- Consider a set of final states of the form $F = \{\{q_i\} \times \widehat{F_{q_i}}\}_{i=1}^m$ where $F_{q_i} \in \Phi(X)$.
- A timed automaton together with a set of final states, F, can be viewed as a transition system, $T = (S, \Sigma, \rightarrow, S_0, S_F)$, with:
  - $S = \mathbf{Q} \times \mathbf{X}$;
  - $\Sigma = E \cup \{\tau\}$, where τ is a symbol denoting time passage;
  - $((q, x), e, (q', x')) \in \rightarrow$ if $e = (q, q') \in E$, $G_e(x) =$ True and $x' \in R(e, x)$.
  - $((q, x), \tau, (q', x')) \in \rightarrow$ if $q = q'$, and there exists $t \geq 0$ such that $x' = x + t(1, \ldots, 1)$.
  - $S_0 = \text{I}nit$; and,
  - $S_F = F$.
- The claim is that every timed automaton is bisimilar to a finite state system! This will be proved in Lecture 14.

4.6 Recall that ...

- Let $X = \{x_1, \ldots, x_n\}$ be a finite collection of variables, each of which takes values in $\mathbb{R}$.
- Let $x = (x_1, \ldots, x_n) \in Re^n = \mathbf{X}$ denote a valuation for all $x_i \in X$.

Definition 4.7 (Clock Constraints) *The set, $\Phi(X)$, of clock constraints of X, is a set of logical expressions defined inductively by $\delta \in \Phi(X)$ if:*

$$\delta := (x_i \leq c) | (x_i \geq c) | \neg\delta | \delta_1 \wedge \delta_2$$

where $x_i \in X$ and $c \geq 0$ is a rational number.

- To each $\delta \in \Phi(X)$ we can associate a set:

$$\hat{\delta} = \{x \in \mathbf{X} \ : \ \delta(x) = \text{True}\}$$

- The original definition of a timed automaton, found in [4], is given below.

Definition 4.8 (Timed Automaton) *A timed automaton is a hybrid automaton $H = (Q, X, \text{I}nit, f, I, E, G, R)$, where*

- *Q is a set of discrete variables, $\mathbf{Q} = \{q_1, \ldots, q_m\}$;*
- *$X = \{x_1, \ldots, x_n\}$, $\mathbf{X} = \mathbb{R}^n$;*
- *$\text{I}nit = \{\{q_i\} \times \widehat{\text{I}nit_{q_i}}\}_{i=1}^m$ where $\text{I}nit_{q_i} \in \Phi(X)$;*
- *$f(q, x) = (1, \ldots, 1)$ for all (q, x);*
- *$\text{I}nv(q) = \mathbf{X}$ for all $q \in \mathbf{Q}$;*
- *$E \subset \mathbf{Q} \times \mathbf{Q}$;*
- *$G(e) = \widehat{G_e}$ where $G_e \in \Phi(X)$, for all $e = (q, q') \in E$; and*
- *For all e, $R(e, x)$ either leaves x_i unaffected or resets it to 0.*
- Notice that R is single valued.

Example: As an example consider the timed automaton of Figure 4.6.

- $Q = \{q\}$, $\mathbf{Q} = \{q_1, q_2\}$;
- $X = \{x_1, x_2\}$, $\mathbf{X} = \mathbb{R}^2$;
- $\text{I}nit = \{(q_1, 0, 0)\}$;
- $f(q, x) = (1, 1)$ for all (q, x);
- $\text{I}nv(q) = \mathbb{R}^2$ for all $q \in \mathbf{Q}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbb{R}^2 \ : \ (x_1 \leq 3) \wedge (x_2 \geq 2)\}$, $G(q_2, q_1) = \{x \in \mathbb{R}^2 \ : \ (x_1 \leq 1)\}$;
- $R(q_1, q_2, x) = \{(0, x_2)\}$, $R(q_2, q_1, x) = \{(x_1, 0)\}$

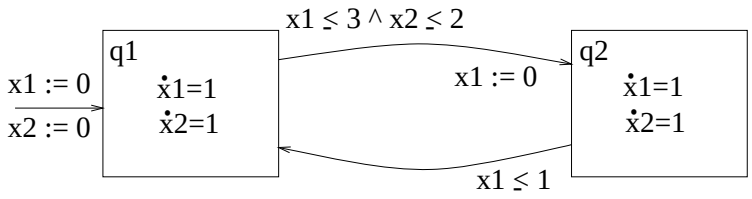


Figure 4.4: Example of a timed automaton

4.6.1 Timed Automata and Transition Systems

- Consider a set of final states of the form $F = \{\{q_i\} \times \widehat{F_{q_i}}\}_{i=1}^{m}$ where $F_{q_i} \in \Phi(X)$.
- A timed automaton together with a set of final states, F, can be viewed as a transition system, $T = (S, \Sigma, \rightarrow, S_0, S_F)$, with:
  - $S = \mathbf{Q} \times \mathbf{X}$;
  - $\Sigma = E \cup \{\tau\}$, where τ is a symbol denoting time passage;
  - $((q, x), e, (q', x')) \in \rightarrow$ if $e = (q, q') \in E$, $G_e(x) =$ True and $x' \in R(e, x)$.
  - $((q, x), \tau, (q', x')) \in \rightarrow$ if $q = q'$, and there exists $t \geq 0$ such that $x' = x + t(1, \ldots, 1)$.
  - $S_0 = \mathrm{I}nit$; and,
  - $S_F = F$.

4.7 Timed Automata are Bisimilar to Finite Systems

- Without loss of generality, all constants can be assumed to be integers.
- Let T be the transition system defined by a timed automaton, H.
- Consider an arbitrary $\lambda > 0$, rational.
- Let H_λ denote the timed automaton obtained by replacing all constants, c, in H by λc.
- Let T_λ denote the transition system associated with H_λ.

Proposition 4.3 *T and T_λ are bisimilar.*

Proof: Consider the relation $(q, x) \sim (q, \lambda x)$. Note that, since $\lambda > 0$, $(x_i \leq c) \Leftrightarrow (\lambda x \leq \lambda c)$ and $(x_i \geq c) \Leftrightarrow (\lambda x \geq \lambda c)$. Therefore:

$$(q, x) \in \mathrm{I}nit \quad \Leftrightarrow \quad (q, \lambda x) \in \mathrm{I}nit_\lambda \tag{4.1}$$

$$(q, x) \in F \quad \Leftrightarrow \quad (q, \lambda x) \in F_\lambda \tag{4.2}$$

$$(q, x) \xrightarrow{e} (q', x') \quad \Leftrightarrow \quad (q, \lambda x) \xrightarrow{e} (q', \lambda x') \tag{4.3}$$

$$(q, x) \xrightarrow{\tau} (q', x') \quad \Leftrightarrow \quad (q, \lambda x) \xrightarrow{\tau} (q', \lambda x') \tag{4.4}$$

For the discrete transition, $(q, x) \xrightarrow{e} (q', x')$ if $e = (q, q') \in E$, $Ge(x) =$ True and $x' \in R(e, x)$. Therefore, $(q, \lambda x) \xrightarrow{e} (q', \lambda x')$, since $e = (q, q') \in E$, $G_{e_\lambda}(\lambda x) =$ True and $\lambda x' \in$

$R_\lambda(e, \lambda x)$. For the continuous transition, recall that $(q, x) \xrightarrow{\tau} (q', x')$ if $q = q'$, and there exists $t \geq 0$ such that $x' = x + t(1, \ldots, 1)$. Therefore, $(q, \lambda x) \xrightarrow{\tau} (q', \lambda x')$ since $q = q'$ and $\lambda x' = \lambda x + \lambda t(1, \ldots, 1)$. ∎

- We can therefore assume all constants are integers. If they are not, we let λ be a common multiple of their denominators and consider the bisimilar system T_λ.
- Let c_i denote the largest constant with which x_i is compared.
- In the above example, $c_1 = 3$ and $c_2 = 2$.
- Let $\lfloor x_i \rfloor$ denote the integer part of x_i and $\langle x_i \rangle$ denote the fractional part of x_i. In other words, $x_i = \lfloor x_i \rfloor + \langle x_i \rangle$, $\lfloor x_i \rfloor \in \mathbb{Z}$ and $\langle x_i \rangle \in [0, 1)$.
- Consider the relation $\sim \subseteq \mathbf{Q} \times \mathbf{X}$ with $(q, x) \sim (q', x')$ if:
  1. $q = q'$;
  2. for all x_i, $\lfloor x_i \rfloor = \lfloor x_i' \rfloor$ or $(\lfloor x_i \rfloor > c_i) \wedge (\lfloor x_i' \rfloor > c_i)$
  3. for all x_i, x_j with $x_i \leq c_i$ and $x_j \leq c_j$
  $$(\langle x_i \rangle \leq \langle x_j \rangle) \Leftrightarrow (\langle x_i' \rangle \leq \langle x_j' \rangle)$$
  4. for all x_i with $x_i \leq c_i$,
  $$(\langle x_i \rangle = 0) \Leftrightarrow (\langle x_i' \rangle = 0)$$

Proposition 4.4 *$\sim$ is an equivalence relation.*

Proof: Probably part of homework 3! ∎

- What do the equivalence classes look like?
- The equivalence classes are either open triangles, open line segments, open parallelograms or points.
- For the example introduced above, they are shown in Figure 4.7.
- Notice that the number of classes is $2(12\text{points} + 30\text{lines} + 18\text{open sets})$. Quite a few, but definitely finite!

Proposition 4.5 *$\sim$ is a bisimulation.*

4.8 Rectangular Hybrid Automatas

So far we have:

- Introduced timed automata.
- Showed that timed automata can be viewed as transition systems.

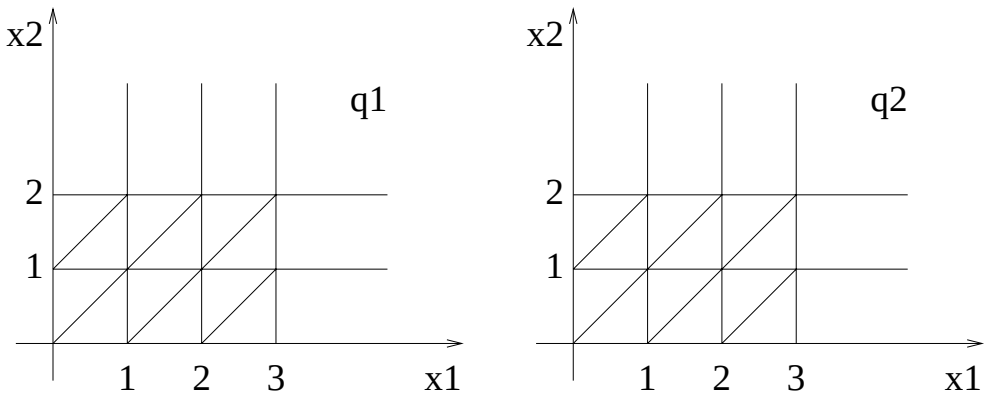


Figure 4.5: Equivalence classes for the example

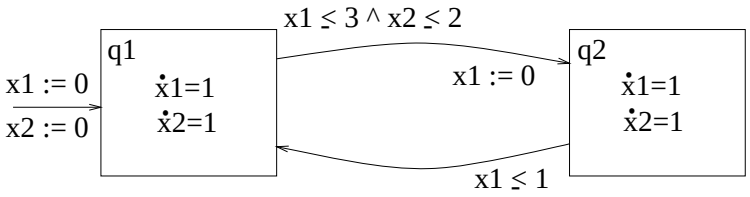


Figure 4.6: Example of a timed automaton

- Showed that without loss of generality all constants can be considered as integers.
- Introduced an equivalence relation, $\sim \subseteq \mathbf{Q} \times \mathbf{X} \times \mathbf{Q} \times \mathbf{X}$ with $(q, x) \sim (q', x')$ if:
  1. $q = q'$;
  2. for all x_i, $\lfloor x_i \rfloor = \lfloor x_i' \rfloor$ or $(x_i > c_i) \wedge (x_i' > c_i)$
  3. for all x_i, x_j with $x_i \leq c_i$ and $x_j \leq c_j$
  $$(\langle x_i \rangle \leq \langle x_j \rangle) \Leftrightarrow (\langle x_i' \rangle \leq \langle x_j' \rangle)$$
  4. for all x_i with $x_i \leq c_i$,
  $$(\langle x_i \rangle = 0) \Leftrightarrow (\langle x_i' \rangle = 0)$$
- Here $\lfloor x_i \rfloor$ denotes the integer part and $\langle x_i \rangle$ the fractional part of x_i and c_i denotes the largest constant with which x_i is compared in the guards, initial and final states of the timed automaton.
- In $\mathbb{R}^2$ the equivalence classes of $\sim$ are open triangles, open rectangles, open line segments (parallel to the axes and at 45 degrees), and points.
- For the timed automaton of Figure 4.6 the equivalence classes are shown in Figure 4.7.

Now, we continue with

Proposition 4.6 *$\sim$ is a bisimulation.*

Proof: We need to show:

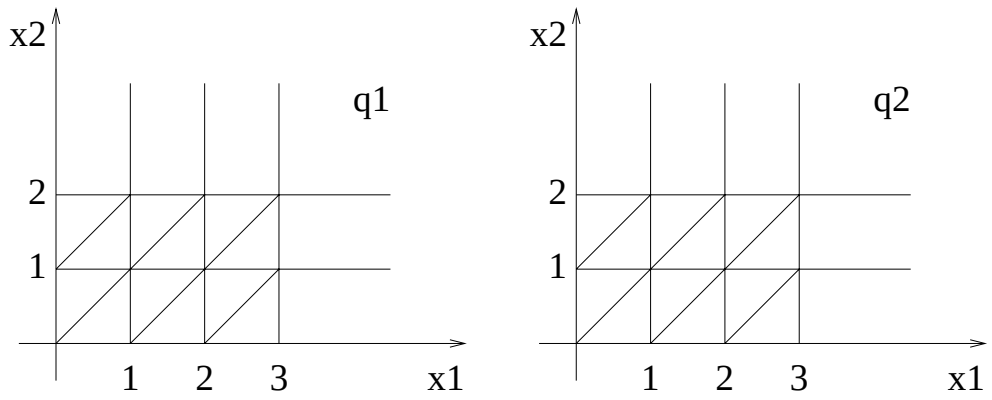


Figure 4.7: Equivalence classes for the example

1. I*nit* is a union of equivalence classes.
2. F is a union of equivalence classes.
3. If P is an equivalence class and $e \in E$, $\mathrm{P}re_e(P)$ is a union of equivalence classes.
4. If P is an equivalence class, $\mathrm{P}re_\tau(P)$ is a union of equivalence classes.

The proof will be somewhat informal. ∎

- First, note that if $\delta \in \Phi(X)$,

  $$\hat{\delta} = \{x \in \mathbf{X} \ : \ \delta(x) = \text{True}\}$$

  is "a union of equivalence classes" (for the X variables only). Recall that $\hat{\delta}$ can be written as the product of unions and intersections of sets of the form:

  $$\{x_i \geq c\}, \{x_i \leq c\}, \{x_i < c\}, \{x_i > c\}, \{x_i = c\}$$

  where c is an integer constant. All these sets are unions of equivalence classes for the X variables.

- This takes care of the requirements on I*nit* and F.

- To deal with $\mathrm{P}re_e(P)$, let:

  $$R^{-1}(e, P) = \{(q, x) \in \mathbf{Q} \times \mathbf{X} \ : \ \exists (q', x') \in P \text{ with } e = (q, q'), x' \in R(e, x)\}$$

- Notice that:

  $$\mathrm{P}re_{(q,q')}(P) = R^{-1}(q, q', P) \cap (\{q\} \times G(q, q'))$$

  Proposition 4.7 *If P is an equivalence class, $R^{-1}(e, P)$ is a union of equivalence classes. Hence $\mathrm{P}re_e(P)$ is a union of equivalence classes.*

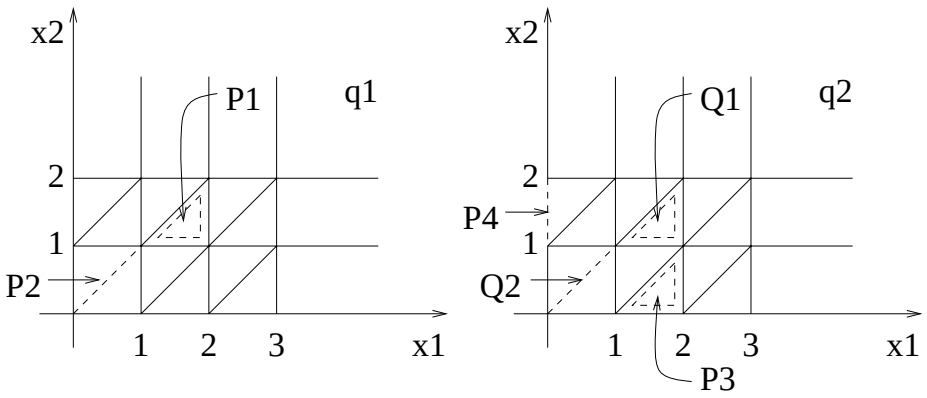


Figure 4.8: Examples of Pre_e computation

- Easier to demonstrate by examples. For the timed automaton of Figure 4.6, consider the equivalence classes $P_1, \ldots, P_4$ shown in Figure 4.8. Notice that:

$$\begin{aligned}
Pre_{e_1}(P_1) &= \emptyset, \\
Pre_{e_2}(P_1) &= Q_1 \cap (\{q_2\} \times \{x_1 \leq 1\}) = \emptyset \\
Pre_{e_1}(P_2) &= \emptyset, \\
Pre_{e_2}(P_2) &= Q_2 \cap (\{q_2\} \times \{x_1 \leq 1\}) = Q_2 \\
Pre_{e_1}(P_3) &= \emptyset \cap (\{q_1\} \times \{x_1 \leq 1 \wedge x_2 \leq 2\}) = \emptyset, \\
Pre_{e_2}(P_3) &= \emptyset \\
Pre_{e_1}(P_4) &= \{q_1\} \times (\{x_1 \geq 0 \wedge 1 < x_2 < 2\} \cap \{x_1 \leq 1 \wedge x_2 \leq 2\}) = \{q_1\} \times \{0 \leq x_1 \leq 3 \wedge 1 < x_2 < 2\}, \\
Pre_{e_2}(P_4) &= \emptyset
\end{aligned}$$

  In all cases the result is a union of equivalence classes.

- Finally, notice that

$$Pre_\tau(P) = \{(q, x) \in \mathbf{Q} \times \mathbf{X} \ : \ \exists (q', x') \in P, t \geq 0 \text{ with } q = q', x' = x + t(1, \ldots, 1)\}$$

  These are all points that if we move in the $(1, \ldots, 1)$ direction we will eventually reach P. If P is an equivalence class, this set is also a union of equivalence classes.

- For example, in Figure 4.9, $Pre_\tau(P) = P \cup P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5$

4.9 Complexity Estimates and Generalizations

- The above discussion indicates that reachability questions for timed automata can be answered on a finite state system, defined on the quotient space of the bisimulation $\sim$ (also known as the *region graph*).

- What is the "size" of this finite state system?

- For the example, the number of equivalence classes is 2(12points+30lines+18open sets) = 120. Quite a few!

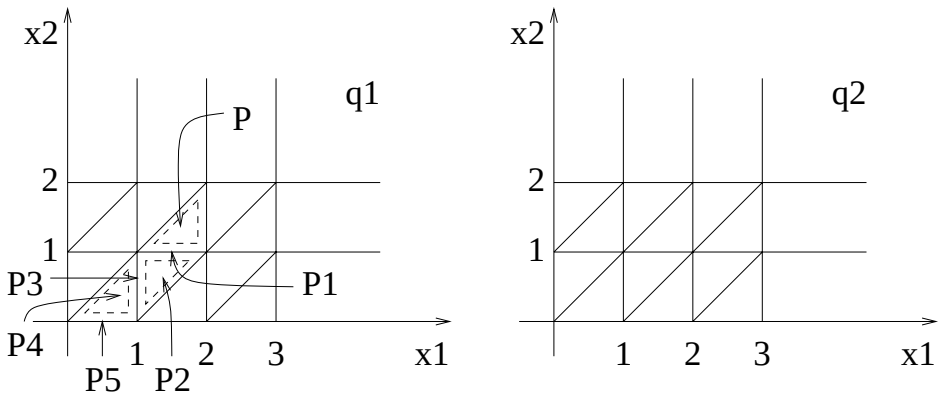


Figure 4.9: Examples of Pre_τ computation

- For an arbitrary system, one can expect up to $m(n!)(2^n)\prod_{i=1}^{n}(2c_i+2)$ discrete states.
- Of course, in general, one need not construct the entire region graph:
  1. On the fly reachability: run the reachability algorithm (Lecture 12). Typically it will terminate, without constructing the entire region graph.
  2. Construct coarser bisimulations: run the bisimulation algorithm (Lecture 13). Typically the bisimulation generated will have fewer equivalence classes than the region graph.
- Still the problem is PSPACE complete!
- The finite bisimulation result is preserved under some simple extensions:
  1. $I(q) = \hat{I}_q$ for some $I_q \in \Phi(X)$.
  2. $f(q,x) = (k_1, \ldots, k_n)$ for some rational $k_1, \ldots, k_n$ and all (q,x).
  3. R mapping equivalence classes to equivalence classes.

4.10 Rectangular Hybrid Automata

- The largest class of systems of this form that is known to be decidable is the class of *initialized rectangular automata*.
- A set $R \subset \mathbb{R}^n$ is called a rectangle if $R = \prod_{i=1}^{n} R_i$ where R_i are intervals whose finite end points are rational.

  Definition 4.9 (Rectangular Automaton) *A rectangular automaton is a hybrid automaton $H = (Q, X, \mathrm{I}nit, f, I, E, G, R)$, where*

  - *Q is a set of discrete variables, $\mathbf{Q} = \{q_1, \ldots, q_m\}$;*
  - *$X = \{x_1, \ldots, x_n\}$, $\mathbf{X} = \mathbb{R}^n$;*
  - *$\mathrm{I}nit = \cup_{i=1}^{m}\{q_i\} \times \mathrm{I}nit(q_i)$ where $\mathrm{I}nit(q_i)$ is a rectangle;*
  - *$f(q,x) = F(q)$ for all (q,x), where $F(q)$ is a rectangle;*

- *$I(q)$ is a rectangle for all $q \in \mathbf{Q}$;*
- *$E \subset \mathbf{Q} \times \mathbf{Q}$;*
- *$G(e)$ is a rectangle for all $e = (q, q') \in E$; and*
- *For all e, $R(e, x)$ either leaves x_i unaffected or resets to an arbitrary value in a rectangle.*

4.11 Exercises

Definition 4.10 (Singular Automaton) *A compact, singular, initialized, rectangular automaton (singular automaton for short!) is a rectangular automaton such that:*

1. **Compact***: for all $e \in E$, $G(e)$ and $R(e, x)$ are compact, and for all $q \in \mathbf{Q}$, $\mathrm{I}nit(q)$ and $F(q)$ are compact, and $I(q) = \mathbf{X}$.*
2. **Singular***: for all $q \in \mathbf{Q}$, $F(q) = F_1(q) \times \ldots \times F_n(q)$ is a singleton, and for all $e \in E$, $R(e, x)$ is a singleton.*
3. **Initialized***: for all $e = (q, q') \in E$, if $F_i(q) \neq F_i(q')$ then $R_i(e, x) \neq \{x_i\}$.*

Definition 4.11 (Stopwatch Automaton) *A stopwatch automaton is a singular automaton such that for all $q \in \mathbf{Q}$, $F_i(q) \in \{0, 1\}$.*

Definition 4.12 (Generalized Timed Automaton) *A generalized timed automaton is a stopwatch automaton such that for all $q \in \mathbf{Q}$, $F_i(q) = 1$.*

Problem 1: Show that every singular automaton is bisimilar to an initialized stopwatch automaton. (Hint: as a bisimulation relation use a time scaling, as in the proof of Proposition 1, Lecture 14).

Problem 2: Show that every initialized stopwatch automaton is bisimilar to a generalized timed automaton. (Hint: introduce additional discrete states to store the values of stopwatches when they stop.) Given that generalized timed automata are bisimilar to finite automata (the bisimulation relation used for timed automata still works!), what can you conclude about the reachability problem for singular and stopwatch automata?

Problem 3: Consider the linear system:

$$\dot{x} = \begin{bmatrix} \frac{2}{3} & 0 \\ 0 & -1 \end{bmatrix} x$$

Let $Y = \{(y_1, y_2) \in \mathbb{R}^2 \ : \ y_1 = 4 \wedge y_2 = 3\}$

1. Characterize $\mathrm{P}re_\tau(Y)$ in terms of a first order formula in $\mathrm{OF}_{\exp}(\mathbb{R}) = \{\mathbb{R}, +, -, \cdot, <, 0, 1, \exp\}$.
2. Going through the procedure outlined in the notes, show that this formula can also be written in quantifier free form.

Problem 4: Consider the linear system:

$$\dot{x} = \begin{bmatrix} 0 & \frac{2}{3} \\ -\frac{2}{3} & 0 \end{bmatrix} x$$

Let again $Y = \{(y_1, y_2) \in \mathbb{R}^2 \ : \ y_1 = 4 \wedge y_2 = 3\}$

1. Characterize $Pre_\tau(Y)$ in terms of a first order formula in $\mathrm{OF}_{\mathrm{an}}(\mathbb{R}) = \{\mathbb{R}, +, -, \cdot, <, 0, 1, \{\hat{f}\}\}$. (Recall that each $\hat{f}$ represents an analytic function, $\hat{f} : \mathbb{R} \to \mathbb{R}$ with its domain restricted to a **compact set**).

2. Going through the procedure outlined in the notes, show that this formula can also be written in quantifier free form.

Chapter 5

Stability of Hybrid Systems

5.1 Stability Definitions

Consider a hybrid automaton H.

Definition 5.1 (Equilibrium) *$x = 0 \in \mathbb{R}^n$ is an equilibrium point of H if:*

1. *$f(q, 0) = 0$ for all $q \in \mathbf{Q}$, and*
2. *$((q, q') \in E) \wedge (0 \in G(q, q')) \Rightarrow R(q, q', 0) = \{0\}$.*

Proposition 5.1 *If $(q_0, 0) \in \mathrm{I}nit$ and $(\tau, q, x) \in \mathcal{H}_{(q_0,0)}$ then $x(t) = 0$ for all $t \in \tau$.*

- If the continuous part of the state starts on the equilibrium point, it stays there forever.
- One would like to characterize the notion that if the continuous state starts close to the equilibrium point it stays close, or even converges to it.
- Use the definitions of Lyapunov for this purpose.

Definition 5.2 (Stable Equilibrium) *Let $x = 0 \in \mathbb{R}^n$ be an equilibrium point of H. $x = 0$ is stable if for all $\epsilon > 0$ there exists $\delta > 0$ such that for all $(\tau, q, x) \in \mathcal{H}_{(q_0,x_0)}$ with $\|x_0\| < \delta$, $\|x(t)\| < \epsilon$ for all $t \in \tau$.*

- Continuity definition, captures the notion of start close, stay close.
- $x = 0$ is called unstable if it is not stable.
- NOT a sequence property. The "$\|x_0\| < \delta$, $\|x(t)\| < \epsilon$ for all $t \in \tau$" part by itself is, but not the quantification over ϵ and δ.
- How about convergence?
- For an infinite execution, $\chi = (\tau, q, x)$, let $\tau_\infty = \sum_i (\tau_i' - \tau_i)$. Notice that $\tau_\infty < \infty$ if χ is Zeno and $\tau_\infty = \infty$ otherwise.

Definition 5.3 (Asymptotically Stable Equilibrium) *Let $x = 0 \in \mathbb{R}^n$ be an equilibrium point of H. $x = 0$ is asymptotically stable if it is stable and there exists $\delta > 0$ such that for all $(\tau, q, x) \in \mathcal{H}^{\infty}_{(q_0,x_0)}$ with $\|x_0\| < \delta$, $\lim_{t\to\tau_\infty} x(t) = 0$.*

- Recall that each τ is fully ordered, so the limit is well defined.
- $\lim_{t\to\infty} x(t) = 0$ does not necessarily imply stability.
- $\lim_{t\to\infty} x(t) = 0$ is a liveness property. $\|x_0\| < \delta$ and $\lim_{t\to\infty} x(t) = 0$ is the intersection of a safety property and a liveness property. Asymptotic stability is not a sequence property, since the stability part is not.
- Variants of the definitions: global asymptotic stability, exponential stability, uniform stability, permutations thereof.
- See [46, 72] for details.

5.2 Continuous Systems

- A continuous dynamical system (Lecture 2) can be thought of as a hybrid automaton with $|\mathbf{Q}| = 1$ and $E = \emptyset$.
- For continuous systems typically use energy arguments to prove stability properties.
- Roughly speaking, if the system "dissipates energy" along its executions it will be stable.
- Statement is clear for mechanical systems, energy has a physical interpretation.
- More generally, use energy-like functions, known as Lyapunov functions.

Theorem 5.1 *Let $x = 0$ be an equilibrium point of H. Assume that there exists an open set D with $0 \in D$ and a continuously differentiable function $V : D \to \mathbb{R}$ such that:*

1. *$V(0) = 0$,*
2. *$V(x) > 0$ for all $x \in D \setminus \{0\}$, and*
3. *$\frac{\partial V}{\partial x}(x) f(x) \leq 0$ for all $x \in D$.*

Then $x = 0$ is a stable equilibrium of H.

Proof: For $r > 0$ let $B_r = \{x \in \mathbb{R}^n : \|x\| < r\}$, $S_r = \{x \in \mathbb{R}^n : \|x\| < r\}$ and $\Omega_r = \{x \in \mathbb{R}^n : V(x) \leq r\}$. Given ϵ:

- Choose $r_1 \in (0, \epsilon)$ such that $B_{r_1} \subseteq D$.
- Let $c_1 = \min_{x \in S_{r_1}} V(x)$.
- Choose $c_2 \in (0, c_1)$. Then $\Omega_{c_2} \subseteq B_{r_1} \subseteq B_\epsilon$.

- Choose $\delta > 0$ such that $B_\delta \subseteq \Omega_{c_2}$.

Since V is not increasing along the system executions, executions that start inside B_δ can not leave Ω_{c_2}, therefore B_{r_1} and therefore B_ϵ. ∎

Theorem 5.2 *If in addition, $\frac{\partial V}{\partial x}(x)f(x) < 0$ for all $x \in D \setminus \{0\}$, then $x = 0$ is an asymptotically stable equilibrium point.*

Example: $\dot{x}_1 = x_2$, $\dot{x}_2 = -(g/l)\sin(x_1) - (k/m)x_2$. $V(x) = (g/l)(1 - \cos(x_1)) + (1/2)x_2^2$.

5.3 Hybrid Systems

One would expect that a hybrid system for which all individual continuous systems are stable would be stable, at least if $R(q, q', x) = \{x\}$ for all $(q, q') \in E$ and $x \in G(q, q')$. This is not the case:

Example: Consider the hybrid automaton, H, with:

- $\mathbf{Q} = \{q_1, q_2\}$ and $\mathbf{X} = \mathbb{R}^2$;
- $Init = \mathbf{Q} \times \{x \in \mathbf{X} : \|x > 0\|\}$;
- $f(q_1, x) = A_1 x$ and $f(q_2, x) = A_2 x$, with:

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

- $I(q_1) = \{x \in \mathbf{X} : x_1 x_2 \leq 0\}$ and $I(q_2) = \{x \in \mathbf{X} : x_1 x_2 \geq 0\}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbf{X} : x_1 x_2 \geq 0\}$ and $G(q_2, q_1) = \{x \in \mathbf{X} : x_1 x_2 \leq 0\}$; and
- $R(q_1, q_2, x) = R(q_2, q_1, x) = \{x\}$.

Proposition 5.2 *$x = 0$ is an equilibrium of H.*

Proof: $f(q_1, 0) = f(q_1, 0) = 0$ and $R(q, q', 0) = \{0\}$. ∎

Proposition 5.3 *the continuous system $\dot{x} = A_i x$ for $i = 1, 2$ are asymptotically stable.*

Still $x = 0$ is unstable for H! Notice that if the switching was done the other way around the system would be stable. So we really have to study the switching, we can not expect to decide stability in general, just by studying the individual systems.

Theorem 5.3 *Consider a hybrid automaton H with $x = 0$ and equilibrium point, $|\mathbf{Q}| < \infty$, and $R(q, q', x) = \{x\}$. Consider an open set $D \subseteq \mathbf{X}$ with $0 \in D$ and a function $V : \mathbf{Q} \times D \to \mathbb{R}$ continuously differentiable in x such that for all $q \in \mathbf{Q}$:*

1. $V(q,0) = 0$,
2. $V(q,x) > 0$ *for all* $x \in D \setminus \{0\}$,
3. $\frac{\partial V}{\partial x}(q,x) f(q,x) \leq 0$ *for all* $x \in D$.

If for all $(\tau, q, x) \in \mathcal{H}$ *and all* $\hat{q} \in \mathbf{Q}$ *the sequence* $\{V(q(\tau_i), x(\tau_i)) \ : \ q(\tau_i) = \hat{q}\}$ *is non increasing, then* $x = 0$ *is a stable equilibrium of* H.

- Can think of Theorem 5.4 as allowing one Lyapunov function for each q.
- When discrete state in $\hat{q}$ corresponding Lyapunov function can not be increasing.
- Every time we switch to a new discrete state, the value of the corresponding Lyapunov function can not be larger than what it was last time we switched there.
- More thorough treatment in [18, 85].

Proof:

Consider the case $\mathbf{Q} = \{q_1, q_2\}$, the proof is similar if the system has more discrete states. $\Omega_r^q = \{x \in \mathbb{R}^n \ : \ V(q,x) \leq r\}$. The trick is that since every time we switch to a discrete state we are below where we were the previous time we switched to it the only thing that matters is the first time we switch to every discrete state.

Given $\epsilon > 0$:

- Choose $r_1 \in (0, \epsilon)$ such that $B_{r_1} \subseteq D$.
- Set $c_1(q) = \min_{x \in S_{r_1}} V(q,x)$, $q \in \mathbf{Q}$.
- Choose $c_2(q) \in (0, c_1(q))$. Then $\Omega^q_{c_2(q)} \subseteq B_{r_1}$.
- Choose $r_2(q) > 0$ such that $B_{r_2(q)} \subseteq \Omega^q_{c_2(q)}$.
- Up to now same as individual Lyapunov proofs.
- Set $r = \min_{q \in \mathbf{Q}} r_2(q)$.
- If either individual system starts within r of the equilibrium, it stays within r.
- Set $c_3(q) = \min_{x \in S_r} V(q,x)$, $q \in \mathbf{Q}$.
- Choose $c_4(q) \in (0, c_3(q))$. Then $\Omega^q_{c_4(q)} \subseteq B_r$.
- Choose $r_4(q) > 0$ such that $B_{r_4(q)} \subseteq \Omega^q_{c_4(q)}$.
- Set $\delta = \min_{q \in \mathbf{Q}} r_4(q)$.

Take $(\tau, q, x) \in \mathcal{H}_{(q_0, x_0)}$, with $\|x_0\| < \delta$ and assume without loss of generality that $q_0 = q_1$. By a continuous Lyapunov argument, $x(t) \in \Omega^{q_1}_{c_4(q_1)} \subseteq B_r$ for $t \in [\tau_0, \tau_0']$. Therefore, $x(\tau_1) = x(\tau_0') \in \Omega^{q_2}_{c_2(q_2)}$. By a continuous Lyapunov argument, $x(t) \in \Omega^{q_2}_{c_2(q_2)} \subseteq B_\epsilon$ for $t \in [\tau_1, \tau_1']$. By assumption, $x(\tau_1') = x(\tau_2) \in \Omega^{q_1}_{c_4(q_1)}$. By a continuous Lyapunov argument, $x(t) \in \Omega^{q_1}_{c_4(q_1)} \subseteq B_r$ for $t \in [\tau_2, \tau_2']$. The claim follows by induction. ∎

- For $|\mathbf{Q}| > 2$ we need to repeat the nested construction $|\mathbf{Q}|$ times.
- What do we need the continuous argument for? It only seems to matter for the first switching time!
- Simple extensions, e.g. R is a contraction in x, or is bounded.
- Reduces to Theorem 5.1 for continuous dynamical systems.
- Can use more Lyapunov functions than $|\mathbf{Q}|$.

5.4 Proof of Theorem 3, Lecture 10

The following theorem comes from [18]

Theorem 5.4 *Consider a hybrid automaton H with $x = 0$ an equilibrium point, $|\mathbf{Q}| < \infty$, and $R(q, q', x) = \{x\}$. Consider an open set $D \subseteq \mathbf{X}$ with $0 \in D$ and a function $V : \mathbf{Q} \times D \to \mathbb{R}$ continuously differentiable in x such that for all $q \in \mathbf{Q}$:*

1. *$V(q, 0) = 0$,*
2. *$V(q, x) > 0$ for all $x \in D \setminus \{0\}$,*
3. *$\frac{\partial V}{\partial x}(q, x) f(q, x) \leq 0$ for all $x \in D$.*

If for all $(\tau, q, x) \in \mathcal{H}$ and all $\hat{q} \in \mathbf{Q}$ the sequence $\{V(q(\tau_i), x(\tau_i)) \ : \ q(\tau_i) = \hat{q}\}$ is non increasing, then $x = 0$ is a stable equilibrium of H

- Can think of Theorem 5.4 as allowing one Lyapunov function for each q.
- When discrete state in $\hat{q}$ corresponding Lyapunov function can not be increasing.
- Every time we switch to a new discrete state, the value of the corresponding Lyapunov function can not be larger than what it was last time we switched there.

Proof: Consider the case $\mathbf{Q} = \{q_1, q_2\}$, the proof is similar if the system has more discrete states. Let $\Omega_r(q) = \{x \in \mathbb{R}^n \ : \ V(q, x) \leq r\}$. Since every time we switch to a discrete state we are bound to be below where we were the previous time we switched to it the only thing that matters is the first time we switch to every discrete state. The following construction takes advantage of that fact.

Consider an arbitrary $\epsilon > 0$. We will try to find $\delta > 0$ such that for all executions (τ, q, x), $x(\tau_0) \in B_\delta$ implies $x(t) \in B_\epsilon$ for all $t \in \tau$.

- Choose $r \in (0, \epsilon)$ such that $B_r \subseteq D$.
- Set $a(q) = \min_{x \in S_r} V(q, x)$, for all $q \in \mathbf{Q}$.
- Choose $b(q) \in (0, a(q))$. Then $\Omega_{b(q)}(q) \subseteq B_r$.
- Choose $p(q) > 0$ such that $B_{p(q)} \subseteq \Omega_{b(q)}(q)$.

- Up to now same as individual Lyapunov proofs. If we knew that the system will always stay in one of the two discrete states, say q_1, (i.e. $\tau < [\tau_0, \infty)$ and $q(\tau_0) = q_1$) and we started with $x(\tau_0) \in B_{p(q_1)}$ we would always stay in $\Omega_{b(q)}(q)$, therefore in B_r, therefore in B_ϵ.
- Set $s = \min_{q \in \mathbf{Q}} p(q)$.
- If we knew that the system will always stay in one discrete state whichever that may be, i.e. $\tau < [\tau_0, \infty))$ and we started with $x(\tau_0) \in B_s$, then we should always stay in B_r, therefore B_ϵ.
- Still have not taken care of switching. To do this we need to repeat the process.
- Set $c(q) = \min_{x \in S_s} V(q, x)$, for all $q \in \mathbf{Q}$.
- Choose $d(q) \in (0, c(q))$. Then $\Omega_{d(q)}(q) \subseteq B_s$.
- Choose $w(q) > 0$ such that $B_{w(q)} \subseteq \Omega_{d(q)}(q)$.
- Set $\delta = \min_{q \in \mathbf{Q}} w(q)$.

Take $(\tau, q, x) \in \mathcal{H}_{(q_0, x_0)}$, with $\|x_0\| < \delta$ and assume without loss of generality that $q_0 = q_1$.

- By a continuous Lyapunov argument, $x(t) \in \Omega_{d(q_1)}(q_1) \subseteq B_s \subseteq B_\epsilon$ for $t \in [\tau_0, \tau_0']$.
- If $\tau < [\tau_0, \infty)$ we are done.
- Otherwise, note that by the assumption on R, $x(\tau_1) = x(\tau_0') \in \Omega^{(}_{d(q_1)} q_1) \subseteq B_s \subseteq \Omega^{(}_{b(q_2)} q_2)$.
- By a continuous Lyapunov argument, $x(t) \in \Omega_{b(q_2)}(q_2) \subseteq B_r \subseteq B_\epsilon$ for all $t \in [\tau_1, \tau_1']$.
- If $\tau < [\tau_0, \tau_0'][\tau_1, \infty)$ we are done.
- Otherwise, note that $q(\tau_2) = q(\tau_0) = q_1$.
- By the non-increasing sequence condition $V(q(\tau_2), x(\tau_2)) \leq V(q(\tau_0), x(\tau_0)) \leq d(q)$.
- Therefore, $x(\tau_2) \in \Omega_{d(q_1)}(q_1) \subseteq B_\epsilon$.
- And so on ... The claim follows by induction.

■

Remarks

- For $|\mathbf{Q}| > 2$ we need to repeat the nested construction $|\mathbf{Q}|$ times.
- Continuous argument needed to:
  - Bound the value of the Lyapunov function at the first switching time.
  - Guarantee stability if there are a finite number of transitions.

- As a consequence of the last remark, Theorem 3 of Lecture 10 for continuous dynamical systems follows from Theorem 1 of Lecture 11 as a corollary.
- Some other immediate Corollaries are the following ...

Corollary 5.1 *Consider a hybrid automaton H with $x = 0$ an equilibrium point, $|\mathbf{Q}| < \infty$, and $R(q, q', x) = \{x\}$. Consider an open set $D \subseteq \mathbf{X}$ with $0 \in D$ and assume there exists a function $V : D \to \mathbb{R}$ continuously differentiable in x such that:*

1. *$V(0) = 0$,*
2. *$V(x) > 0$ for all $x \in D \setminus \{0\}$,*
3. *$\frac{\partial V}{\partial x}(x) f(q, x) \leq 0$ for all $q \in \mathbf{Q}$ and all $x \in D$.*

Then $x = 0$ is a stable equilibrium of H

Proof: Define $\hat{V} : \mathbf{Q} \times \mathbf{X} \to \mathbb{R}$ by $\hat{V}(q, x) = V(x)$ for all $q \in \mathbf{Q}$, $x \in \mathbf{X}$ and apply Theorem 1. ∎

Corollary 5.2 *Consider a hybrid automaton H with $x = 0$ an equilibrium point, $|\mathbf{Q}| < \infty$, and assume $R(q, q', x)$ is non-expanding. Consider an open set $D \subseteq \mathbf{X}$ with $0 \in D$ and a function $V : \mathbf{Q} \times D \to \mathbb{R}$ continuously differentiable in x such that for all $q \in \mathbf{Q}$:*

1. *$V(q, 0) = 0$,*
2. *$V(q, x) > 0$ for all $x \in D \setminus \{0\}$,*
3. *$\frac{\partial V}{\partial x}(q, x) f(q, x) \leq 0$ for all $x \in D$.*

If for all $(\tau, q, x) \in \mathcal{H}$ and all $\hat{q} \in \mathbf{Q}$ the sequence $\{V(q(\tau_i), x(\tau_i)) \ : \ q(\tau_i) = \hat{q}\}$ is non increasing, then $x = 0$ is a stable equilibrium of H

R is non-expanding if for all $(q, q') \in E$, all $x, y \in \mathbf{X}$, all $x' \in R(e, x)$ and all $y' \in R(e, y)$:

$$\|x' - y'\| \leq \|x - y\|$$

In particular, setting $y = 0$ (and recalling that 0 being an equilibrium requires $R(e, 0) = \{0\}$) non-expanding implies that for all $e \in E$, for all $x \in \mathbf{X}$, and all $x' \in R(e, x)$

$$\|x'\| \leq \|x\|$$

Notice that unless the non-expanding assumption is made the proof of Theorem 1 breaks down at the time of the first jump. I believe that R being a continuous function will also work, but I did not bother to prove so (it requires some more argument).

5.5 Other Lyapunov-like Theorems

More general reset relations are also covered by the following theorem (which can be found in [85]).

Theorem 5.5 *Consider a hybrid automaton H with $|\mathbf{Q}| < \infty$. Consider an open set $D \subseteq \mathbf{X}$ with $0 \in D$ and a function $V : \mathbf{Q} \times D \to \mathbb{R}$ continuous in x with $V(q,0) = 0$ and $V(q,x) > 0$ for all $x \in D \setminus \{0\}$. Assume that for all $(\tau, q, x) \in \mathcal{H}$ the sequence $\{V(q(\tau_i), x(\tau_i))\}$ is non increasing and that there exists a continuous function $g : \mathbb{R}^+ \to \mathbb{R}^+$ with $g(0) = 0$, such that for all $t \in [\tau_i, \tau_i']$, $V(q(t), x(t)) \leq g((V(q(\tau_i), x(\tau_i))$. Then $x = 0$ is a stable equilibrium of H*

Remarks:

- The conditions of Theorem 2 are weaker than those of Theorem 1 since:
  - R is not constrained.
  - V is not required to be decreasing along continuous evolution as long as it remains bounded by g. If the Lyapunov function happens to be decreasing, $g(x) = x$ will work.
- The conditions of Theorem 2 are stronger than those of Theorem 1 since they require the sequence over all i to be non-increasing (not just the individual subsequences for which $q(\tau_i) = \hat{q}$).

Theorems 1 and 2 can in fact be applied to more general invariant sets.

- A set $S \subseteq \mathbf{Q} \times \mathbf{X}$ is called invariant if for all $(\tau, q, x) \in \mathcal{H}_{(q_0,x_0)}$, $(q_0, x_0) \in S$ implies that $(q(t), x(t)) \in S$ for all $t \in \tau$. Watch out for the overloading of the terminology. The term invariant has already been used twice: in the definition of hybrid automata to describe sets of continuous states for which continuous evolution at a particular discrete state is allowed, and also to describe sets of states that contain all the reachable states.
- An equilibrium point $x = 0$ is a special case of an invariant set, of the form $S = \mathbf{Q} \times \{0\}$.
- For continuous systems, other types of invariant sets include limit cycles, level sets of Lyapunov functions, etc.
- The above theorems extend to more general invariant sets. The proofs are obtained by replacing the Euclidean norm $\|\cdot\|$ by the "distance to the set S" defined as:

  $$d(q,x) = \min_{(\hat{q},\hat{x}) \in S} (d_D(q,\hat{q}) + \|x - \hat{x}\|)$$

  where $d_D(q,\hat{q})$ is the discrete metric defined by $d_D(q,\hat{q}) = 0$ is $q = \hat{q}$ and $d_D(q,\hat{q}) = 1$ otherwise.

This and extensions to asymptotic stability, exponential stability, boundedness and converse theorems can be found in [85].

5.6 Example

Consider the hybrid automaton, H, with:

- $\mathbf{Q} = \{q_1, q_2\}$ and $\mathbf{X} = \mathbb{R}^2$;
- $\mathrm{I}nit = \mathbf{Q} \times \{x \in \mathbf{X} : \|x > 0\|\}$;
- $f(q_1, x) = A_1 x$ and $f(q_2, x) = A_2 x$, with:

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

- $I(q_1) = \{x \in \mathbf{X} : Cx \geq 0\}$ and $I(q_2) = \{x \in \mathbf{X} : Cx \leq 0\}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbf{X} : Cx \leq 0\}$ and $G(q_2, q_1) = \{x \in \mathbf{X} : Cx \geq 0\}$; and
- $R(q_1, q_2, x) = R(q_2, q_1, x) = \{x\}$.

We show that $x = 0$ is a stable equilibrium point of H.

Proposition 5.4 *$x = 0$ is an equilibrium of H.*

Proof: $f(q_1, 0) = f(q_1, 0) = 0$ and $R(q, q', 0) = \{0\}$. ∎

Proposition 5.5 *the continuous system $\dot{x} = A_i x$ for $i = 1, 2$ are asymptotically stable.*

Proposition 5.6 *If $\dot{x} = A_i x$ is stable, then there exists $P_i = P_i^T > 0$ such that $A_i^T P_i + P_i A_i = -I$.*

Proof: See [23]. ∎

Recall that $P_i > 0$ (positive definite) if and only if $x^T P_i x > 0$ for all $x \neq 0$, and I is the identity matrix.

Consider the candidate Lyapunov function:

$$V(q, x) = \begin{cases} x^T P_1 x & \text{if } q = q_1 \\ x^T P_2 x & \text{if } q = q_2 \end{cases}$$

Check that the conditions of the Theorem hold. For all $q \in \mathbf{Q}$:

1. $V(q, 0) = 0$
2. $V(q, x) > 0$ for all $x \neq 0$ (since the P_i are positive definite)

3. $\frac{\partial V}{\partial x}(q,x)f(q,x) \leq 0$ for all x since:

$$\begin{aligned}
\frac{\partial V}{\partial x}(q,x)f(q,x) &= \frac{d}{dt}V(q,x(t)) \\
&= \dot{x}^T P_i x + x^T P_i \dot{x} \\
&= x^T A_i^T P_i x + x^T P_i A_i x \\
&= x^T (A_i^T P_i + P_i A_i) x \\
&= -x^T I x \\
&= -\|x\|^2 i \leq 0
\end{aligned}$$

It remains to test the non-increasing sequence condition. Notice that the level sets of $x^T P_i x$ are ellipses centered at the origin. Therefore each level set intersects the switching line $Cx = 0$ at exactly two points, $\hat{x}$ and $-\hat{x}$. Assume $x(\tau_i) = \hat{x}$ and $q(\tau_i) = q_1$. The fact that $V(q_1, x(t))$ decreases for $t \in [\tau_i, \tau_i']$ (where $q(t) = q_1$) implies that the next time the switching line is reached, $x(\tau_i') = \alpha(-\hat{x})$ for some $\alpha \in (0,1)$. Therefore, $\|x(\tau_{i+1})\| = \|x(\tau_i')\| < \|x(\tau_i)\|$. By a similar argument, $\|x(\tau_{i+2})\| = \|x(\tau_{i+1}')\| = \|x(\tau_{i+1})\|$ and $Cx(\tau_{i+2}) = 0$. Therefore, $V(q(\tau_i), x(\tau_i)) \leq V(q(\tau_{i+2}), x(\tau_{i+2}))$.

Remark: For hybrid systems like this, where the dynamics in each one of the discrete states is linear (or affine), it is possible to obtain piecewise quadratic Lyapunov functions computationally, using convex optimization techniques (in particular, Linear Matrix Inequalities) [45].

Chapter 6

Controller Design for Hybrid Systems

6.1 Formulation of the Controller Synthesis Problem

A control problem involves:

1. A plant
2. A specification
3. A controller

Controller synthesis involves coming up with a methodology for designing controllers that meet the specification. The discussion in this lecture is based on [58] and [76].

Recall that the *plant* is modeled by an open hybrid automaton, $H = (Q, X, V, \mathrm{I}nit, f, I, E, G, R, \phi)$, where

- Q is a finite collection of discrete state variables;
- X is a finite collection of continuous state variables;
- V is a finite collection of input variables. We assume $V = V_D \cup V_C$, where V_D contains discrete and V_C contains continuous variables.
- $\mathrm{I}nit \subseteq \mathbf{Q} \times \mathbf{X}$ is a set of initial states;
- $f : \mathbf{Q} \times \mathbf{X} \times \mathbf{V} \to \mathbb{R}^n$ is an input dependent vector field;
- $I : \mathbf{Q} \to 2^{\mathbf{X} \times \mathbf{V}}$ assigns to each $q \in \mathbf{Q}$ an input dependent invariant set;
- $E \subset \mathbf{Q} \times \mathbf{Q}$ is a collection of discrete transitions;
- $G : E \to 2^{\mathbf{X} \times \mathbf{V}}$ assigns to each $e = (q, q') \in E$ a guard;
- $R : E \times \mathbf{X} \times \mathbf{V} \to 2^{\mathbf{X}}$ assigns to each $e = (q, q') \in E$, $x \in \mathbf{X}$ and $v \in \mathbf{V}$ a reset relation; and,

- $\phi : \mathbf{Q} \times \mathbf{X} \to 2^{\mathbf{V}}$ assigns to each state a set of admissible inputs.

To avoid technical difficulties we introduce the additional assumption that $\phi(q, x) \neq \emptyset$ and f is Lipschitz in x and continuous in v.

Also recall that the control objective is assumed to be encoded by means of a safety property $(Q \cup X, \Box F)$ (always F) with $F \subseteq \mathbf{Q} \times \mathbf{X}$.

Finally recall that we assume the input variables of H are partitioned into *controls*, U and *disturbances*, D:

$$V = U \cup D$$

The disturbances represent uncontrolled inputs such as noise, unmodeled dynamics, the actions of other subsystems (in a distributed system), etc.

6.2 Controllers

A controller can be defined as a map from state executions to sets of allowable inputs:

$$C : \mathcal{X}^* \to 2^{\mathbf{U}}$$

The interpretation is that given a finite execution, the controller restricts the valuations of the control input variables that are allowed at the final state. With this in mind we can define the set of *closed loop causal executions* as:

$$\mathcal{H}_C = \{(\tau, q, x, (u, d)) \in \mathcal{H} | \forall t \in \tau, u(t) \in C((\tau, q, x) \downarrow_t)\}$$

Clearly, $\mathcal{H}_C \subseteq \mathcal{H}$. We say C satisfies property $(Q \cup X, \Box F)$ if:

$$\Box F(\chi) = \text{True for all } \chi \in \mathcal{H}_C$$

To prevent technical problems, assume that for all $(\{[\tau_i, \tau_i']\}_{i=0}^N, q, x, v) \in \mathcal{H}$, $\emptyset \neq C(x) \subseteq \phi(q(\tau_N'), x(\tau_N'))|_U$. This ensures that the controller will not attempt to "cheat" by stalling the execution. This is not enough however. The controller may still be able to cheat by:

1. Blocking the execution at a state (q, x) by applying $u \in \phi(q, x)|_U$ such that for all $d \in \phi(q, x)|_D$ $(x, (u, d)) \notin I(q)$ and for all $e \in E$ either $(x, (u, d)) \notin G(e)$ or $R(e, x, (u, d)) = \emptyset$.

2. Forcing the execution to be Zeno (take infinite number of transitions in a finite amount of time). Recall the water tanks example.

Both of these caveats have to do with modeling over-abstraction: the model of the plant should be such that the controller is not able to cheat in this way. The first loophole can be eliminated by fairly simple assumptions on the plant, similar to the non-blocking assumptions for autonomous hybrid automata. The second loophole is more difficult to deal with. Typically one assumes that all loops among discrete states require a non zero amount of time. This assumption may be somewhat restrictive and is difficult to enforce by manipulating the primitives of the model. Here we will overlook these technical problems.

6.3 Basic Controller Properties

Given a plant hybrid automaton and a property our goal is to find a controller that satisfies the property.

Memoryless Controllers

A controller, C, is called *memoryless* (or sometimes pure feedback) if for all $\chi, \chi' \in \mathcal{H}^*$ ending at the same state, $C(\chi) = C(\chi')$. A memoryless controllers can be characterized by a feedback map:

$$g : \mathbf{Q} \times \mathbf{X} \rightarrow 2^{\mathbf{U}}$$

To prevent technical problems we again restrict our attention to memoryless controllers such that for all $(q, x) \in \mathbf{Q} \times \mathbf{X}$ $\emptyset \neq g(q, x) \subseteq \phi(q, x)$. Given a plant, H, and a memoryless controller, g, we can defined the closed loop open hybrid automaton, $H_g = (Q, X, V, \mathrm{I}nit, f, I, E, G, R, \phi_g)$, where $\phi_g(q, x) = \phi(q, x) \cap g(q, x)$. It is easy to show that:

Proposition 6.1 *If C is memoryless controller with feedback map g, then $\mathcal{H}_g = \mathcal{H}_C$.*

This property allows one to nest controller synthesis problems, provided they can be solved by memoryless controllers. In general, is unclear whether the set of closed loop causal executions is the set of executions of some hybrid automaton.

For properties of the form $(Q \cup X, \Box F)$, it turns out that it it suffices to look for a solution among memoryless controllers.

Proposition 6.2 *A controller that satisfies property $(Q \cup X, \Box F)$ exists if and only if a memoryless controller that satisfies $(Q \cup X, \Box F)$ exists.*

Proof: The if part is obvious. For the only if part, assume that there exists a controller C that solves the synthesis problem $(H, \Box F)$, but there does not exist a feedback controller that solves the synthesis problem. Therefore, there must exist $(q, x) \in F$ and two different finite executions $\chi_1 = (\tau_1, q_1, x_1, (u_1, d_1)) \in \mathcal{H}_C$ and $\chi_2 = (\tau_2, q_2, x_2, (u_2, d_2)) \in \mathcal{H}_C$ ending in (q, x) such that $C(q_1, x_1) \neq C(q_2, x_2)$. Moreover, the "information" about whether (q, x) was reached via χ_1 or whether it was reached via χ_2 must be essential for subsequent control decisions.

More formally, assume (q, x) is reached via χ_2, and let χ' denote a subsequent execution, that is assume that the concatenation $\chi_2\chi'$ belongs to $\mathcal{H}$. Note that, since χ_1 also ends in (q, x), $\chi_1\chi'$ also belongs to $\mathcal{H}$. Let $\chi_2\chi' = (\tau_2', q_2', x_2', (u_2', d_2'))$ and $\chi_1\chi' = (\tau_1', q_1', x_1', (u_1', d_1'))$. Assume that for all $t \in \tau_2' \setminus \tau_2$, a control $u(t) \in C((q_1', x_1') \downarrow_t)$ is applied (instead of a control $u(t) \in C((q_2', x_2') \downarrow_t)$). Then, as the fact that (q, x) was reached via χ_2 is essential, there must exist a subsequent execution χ' such that $\chi_2\chi' \in \mathcal{H}$ (in fact $\chi_2\chi' \in \mathcal{H} \setminus \mathcal{H}_C$) and $\Box F(\chi_2\chi') = \text{False}$. This implies that there exists $t \in \tau_2'$ such that $(q_2'(t), x_2'(t)) \in F^c$. Since C is assumed to solve the synthesis problem and $\chi_2 \in \mathcal{H}_C$, $\Box F(\chi_2) = \text{True}$, therefore $t \in \tau_2' \setminus \tau_2$.

However, since for all $t \in \tau_2' \setminus \tau_2$, $u(t) \in C((q_1', x_1') \downarrow_t)$, and $(\tau_1', q_1', x_1', (u_1', d_1')) \in \mathcal{H}$, we have that $\chi_1\chi' \in \mathcal{H}_C$. But the above discussion indicates that there exists $t \in \tau_1'$ (in fact

$t \in \tau_1' \setminus \tau_1$) such that $(q_1'(t), x_1'(t)) \in F^c$. This contradicts the assumption that C solves the synthesis problem $(H, \Box F)$. ∎

Motivated by Proposition 6.2, we restrict our attention to feedback controllers. For brevity, we refer to the problem of finding a controller for a plant H that satisfies a specification $(Q \cup X, \Box F)$ as the *controller synthesis problem* $(H, \Box F)$.

Controlled Invariant Sets

Typically, for a controller synthesis problem one treats the set of initial conditions, I*nit*, as variable and attempts to establish the largest set of states for which there exists a controller that satisfies the specification. This set of initial conditions turns out to be a "controlled invariant set".

Definition 6.1 (Controlled Invariant) *A set $W \subseteq \mathbf{Q} \times \mathbf{X}$ is called controlled invariant if there exists a controller that solves the controller synthesis problem $(H', \Box W)$ when H' is identical to H except for* I*nit'* *which is equal to W.*

A controlled invariant set W is called *maximal* if it is not a proper subset of another controlled invariant set. We say a controller *renders W invariant* if it solves the controller synthesis problem $(H', \Box W)$ where I$nit' = W$.

Proposition 6.3 *A controller that solves the synthesis problem $(H, \Box F)$ exists if and only if there exists a unique maximal controlled invariant $W \subseteq \mathbf{Q} \times \mathbf{X}$ such that* I$nit \subseteq W \subseteq F$.

Proof: If there exists any control invariant $W \subseteq F$ (in particular, if there exists a unique maximal one) then, by definition, the synthesis problem $(H, \Box F)$ can be solved for $I = W$.

For the only if part, if the synthesis problem can be solved for some I*nit*, there exists a set $\widehat{\text{I}nit}$ and a feedback controller g such that for all d and for all $(q_0, x_0) \in \widehat{\text{I}nit}$ the execution $(\tau, q, x, (u, d))$ with $u(t) \in g(q(t), x(t))$ for all $t \in \tau$ satisfies $(q(t), x(t)) \in F$ for all $t \in \tau$. Consider the set:

$$W = \bigcup_{d} \bigcup_{(q_0,x_0)\in\widehat{\text{I}nit}} \bigcup_{t\in\tau} (q(t), x(t))$$

Then clearly $W \subseteq F$. Moreover, for any $(q_0, x_0) \in W$ consider the execution $(\tau, q, x, (u, d))$ with arbitrary $d \in \mathcal{D}$ and $u(t) \in g(q(t), x(t))$. Then, by definition of W, $(q(t), x(t)) \in W$ for all $t \in \tau$. Therefore, controller g renders the set W invariant.

Having established the existence of controlled invariant subsets of F, consider now two such sets $W_1 \subseteq F$ and $W_2 \subseteq F$. We show that their union is also a controlled invariant subset of F. Clearly $W_1 \cup W_2 \subseteq F$. For $i = 1, 2$, as W_i is controlled invariant, there exists a feedback controller g_i that solves the controller synthesis problem $(H, \Box W_i)$, with I$nit = W_i$. Consider the feedback controller g with:

$$g(q, x) = \begin{cases} g_1(q, x) & \text{if } (q, x) \in W_1 \\ g_2(q, x) & \text{otherwise} \end{cases}$$

Consider an arbitrary $(q_0, x_0) \in W_1 \cup W_2$. Then either $(q_0, x_0) \in W_1$ or $(q_0, x_0) \in (W_1 \cup W_2) \setminus W_1 \subseteq W_2$. In the first case, all executions are guaranteed to satisfy $\Box W_1$ as g_1 renders

W_1 invariant. For the second case, consider an arbitrary execution $\chi = (\tau, q, x, (u, d))$ with $u(t) \in g(q(t), x(t))$ for all $t \in \tau$. Since g_2 solves the controller synthesis problem $(H, \Box W_2)$ with $Init = W_2$, either $\Box(W_2 \setminus W_1)(\chi) = \text{True}$ or $(q, x) \in W_2 \setminus W_1$ until $(q, x) \in W_1$, which brings us back to the first case. Hence, g solves the controller synthesis problem $(H, \Box(W_1 \cup W_2))$ with $Init = W_1 \cup W_2$, and the set $W_1 \cup W_2$ is controlled invariant.

Summarizing, the class of controlled invariant subsets of F is closed under union. Hence, it possesses a unique maximal element. ∎

Least Restrictive Controllers

We would like to derive a memoryless controller that solves the problem while imposing minimal restrictions on the controls it allows. There are at least two reasons why such a controller is desirable:

1. As discussed above, safety properties can sometimes be satisfied using trivial controllers (that cause deadlocks or zeno executions for example). Imposing as few restrictions as possible allows us to find a meaningful controller whenever possible.

2. In many cases multiple, prioritized specifications are given for a particular problem. Imposing fewer restrictions on the controls when designing controllers for higher priority specifications allows us greater flexibility when trying to satisfy lower priority specifications.

Memoryless controllers that solve the synthesis problem $(H, \Box F)$ can be partially ordered by the relation:

$$g_1 \preceq g_2 \Leftrightarrow g_1(x) \subseteq g_2(x) \text{ for all } x \in \mathbf{X}$$

Definition 6.2 *A memoryless controller that solves* $(H, \Box F)$ *is called* least restrictive *if it is maximal among the controllers that solve* $(H, \Box F)$.

There is a conjecture that for every controller synthesis problem $(H, \Box F)$ either there is no solution or there exists a unique least restrictive controller that solves the problem. As of now there is no proof of this fact however.

Some Remarks on "Implementation"

The notion of a controller introduced above may be inadequate when it comes to implementation. For one thing, the set valued map g allows non-deterministic choices of control inputs. Since in practice only one input can be applied to the system at any time, this nondeterminism has to somehow be resolved when it comes time to implement such a controller. The set valued map can in this sense be thought of as a family of single valued controllers; implementation involves choosing one controller from this family.

Normally, one would "implement" a controller by another hybrid automaton, which, when composed with the plant automaton yields the desired behavior. To do this one would need to introduce output variables to the hybrid automaton and define formal semantics for composition, as in Lecture 8. The process is slightly more complicated for the models

considered here because of the presence of the state dependent input constraints, encoded by ϕ.

We assume that the entire state is available to the controller. In general this will not be the case. If a controller is to be implemented by a hybrid automaton, the information the controller has about the plant is obtained through the valuations of the output variables of the plant, which are not necessarily in one to one correspondence with the valuations of the state variables. The controller synthesis problem under partial observation (output feedback) is much more complicated than the full observation (state feedback) problem addressed here (partly because it makes it harder to define composition as discussed above).

6.4 Game Theoretic Approach to Controller Synthesis

To guarantee that a safety specification is met despite the action of the disturbances we cast the design problem as a zero sum dynamic game. The two players in the game are the control u and the disturbance d and they compete over cost a function that encodes the safety specification. We seek the best possible control action and the worst possible disturbance. Note that if the specifications can be met for this pair then they can also be met for any other choice of the disturbance.

Consider a controller synthesis problem $(H, \Box F)$. The game can be cast in the standard min-max setting by introducing a cost function induced by the discrete metric. The discrete metric is simply a map $m : (\mathbf{Q} \times \mathbf{X}) \times (\mathbf{Q} \times \mathbf{X}) \longrightarrow \mathbb{R}$ defined by:

$$m((q_1, x_1), (q_2, x_2)) = \begin{cases} 0 & \text{if } (q_1, x_1) = (q_2, x_2) \\ 1 & \text{if } (q_1, x_1) \neq (q_2, x_2) \end{cases}$$

It is easy to check that m satisfies the axioms of a metric. The metric induces a map on subsets of $\mathbf{Q} \times \mathbf{X}$ by defining:

$$\begin{aligned} M : 2^{\mathbf{Q}\times\mathbf{X}} \times 2^{\mathbf{Q}\times\mathbf{X}} &\longrightarrow \mathbb{R} \\ (W_1, W_2) &\longmapsto \min_{((q_1,x_1),(q_2,x_2))\in W_1\times W_2} m((q_1, x_1), (q_2, x_2)) \end{aligned}$$

In other words, $M(W_1, W_2) = 0$ if $W_1 \cap W_2 \neq \emptyset$ and $M(W_1, W_2) = 1$ if $W_1 \cap W_2 = \emptyset$.

Consider an execution, $\chi = (\tau, q, x, (u, d))$, of the hybrid automaton H starting at an initial state $(q_0, x_0) \in I$. Define the *cost* of this execution by:

$$\begin{aligned} J : \mathcal{H} &\longrightarrow \mathbb{R} \\ \chi &\longmapsto \min_{t\in\tau} M(\{(q(t), x(t))\}, F^c) \end{aligned}$$

Note that J can only take on two values, $J(\chi) \in \{0, 1\}$. Therefore, J implicitly defines a property of the hybrid automaton. In fact:

Proposition 6.4 *$J(\chi) = 1$ if and only if $\Box F(\chi) = True$.*

Intuitively, u tries to maximize the cost function J (prevent the state from leaving F). Because we have no control over the actions of d, we assume that it tries to minimize J (force the state to leave F). As we would like to establish conditions under which $\Box F$ is guaranteed

to be satisfied we bias the game in favor of the disturbance whenever there is ambiguity over how the game will proceed. For example, multiple executions may be possible for the same initial condition, control and disturbance trajectories, due to nondeterminism. Moreover, the order in which the two players play in the game may be important, if one player is assumed to have access to the decision of the other player before choosing his/her action. In both these cases we would like to give the disturbance the "benefit of the doubt".

Consider the max-min solution:

$$J^*(q_0, x_0) = \max_g \min_d \left(\min_{\chi=(\tau,q,x,(u,d))\in\mathcal{H}_g} J(\chi) \right) \tag{6.1}$$

Motivated by Proposition 6.5 we restrict our attention to feedback strategies for u in equation (6.8). Following the standard game theoretic convention, the "player" who appears first in the right hand side of equation (6.8) (the controller g) is also assumed to play first. The player who appears second (the disturbance d) is assumed to have access to the strategy of the first player, when called upon to make his/her decision. The minimum over χ removes all nondeterminism. Therefore, provided a solution to this equation can be found, J^* is a well defined function of the initial state (q_0, x_0). In addition, the minimum over χ implicitly restricts attention to control and disturbance trajectories that satisfy the state based input constraint ϕ. Using J^* we define a set $W^* \subseteq \mathbf{Q} \times \mathbf{X}$ by:

$$W^* = \{(q_0, x_0) \in \mathbf{Q} \times \mathbf{X} | J^*(q_0, x_0) = 1\} \tag{6.2}$$

Proposition 6.5 *W^* is the maximal controlled invariant subset of F.*

Proof: We first show W^* is controlled invariant. Assume for the sake of contradiction that it is not. Then for all g there exists an $(q_0, x_0) \in W^*$, a d, a $\chi = (\tau, q, x, (u, d)) \in \mathcal{H}_g$ and a $t \in \tau$ with $(q(t), x(t)) \notin W^*$. By Proposition 6.4, $J(\chi) = 0$, which, by equation (6.8), implies that $J^*(q_0, x_0) = 0$. This contradicts the assumption that $(q_0, x_0) \in W^*$.

Next we show that W^* is maximal. Assume for the sake of contradiction that it is not, that is there exists another controlled invariant $\hat{W}$ with $W^* \subset \hat{W} \subseteq F$. Then, by definition, there exists a controller g such that $\mathcal{H}_g$ satisfies $\Box F$ with $I = \hat{W}$. In other words, for all $(q_0, x_0) \in \hat{W}$, for all d and for all $\chi = (\tau, q, x, (u, d)) \in \mathcal{H}_g$, $\Box F(\chi) =$ True, or, equivalently, $J(\chi) = 1$. But, from equation (6.8) this would imply that $J^*(q_0, x_0) = 1$. This contradicts the assumption that $W^* \subset \hat{W}$. ∎

If a solution to equation 6.8 can be computed, then there exists a feedback controller (namely one that achieves the maximum) that renders W^* invariant. We would like to find a least restrictive such controller. Our ability to solve the synthesis problem using this technique hinges on finding a solution to equation 6.8. In some cases this can be done by brute force; this typically involves guessing a controller and showing that it achieves the maximum. More systematically, this can be done using optimal control techniques, in particular dynamic programming.

6.5 Example: The Steam Boiler

Recall the steam boiler problem from Lecture 8 (Figure 6.1). The problem was introduced first in [2, 1]. The model presented here is taken from [37]. The controller synthesis for this

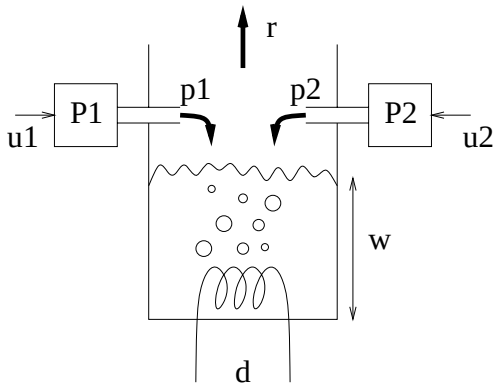


Figure 6.1: The Steam Boiler

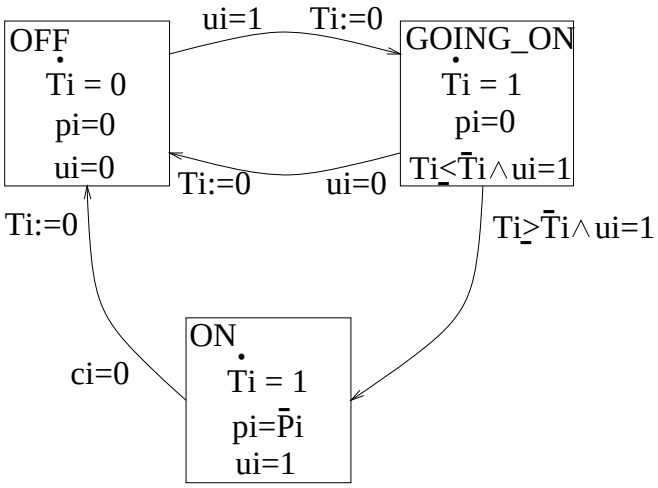


Figure 6.2: The pump hybrid automaton

model can be found in [58]. The continuous dynamics of the boiling process are summarized by the differential equations:

$$\begin{aligned} \dot{w} &= p_1 + p_2 - r \\ \dot{r} &= d \end{aligned}$$

The dynamics of pump i are summarized by the open hybrid automaton of Figure 6.2. Notice that p_i is both an output variable of the pump and an input variable of the boiling process. For a formal definition of the model (with slight differences in the notation) please refer to Lecture 8.

The composite automaton has 4 continuous, real valued state variables:

$$x = (w, r, T_1, T_2) \in \mathbb{R}^4$$

9 discrete states:

$$q \in \{(\mathit{OFF}, \mathit{OFF}), (\mathit{OFF}, \mathit{GOING\_ON}), \ldots, (\mathit{ON}, \mathit{ON})\}$$

2 discrete input variables:

$$(u_1, u_2) \in \{0, 1\} \times \{0, 1\} = \mathbf{U}$$

and one continuous input variable:

$$d \in [-D_1, D_2] = \mathbf{D}$$

s the notation suggests, u_1 and u_2 will play the role of controls and d will play the role of the disturbance. The additional requirement that $r \in [0, R]$ can be encoded by a state dependent input constraint:

$$\phi(q,x) = \begin{cases} \mathbf{U} \times [0, D_2] & \text{if } r \leq 0 \\ \mathbf{U} \times \mathbf{D} & \text{if } r \in (0, R) \\ \mathbf{U} \times [-D_1, o] & \text{if } r \geq R \end{cases}$$

Proposition 6.6 *If* $Init \subseteq \mathbf{Q} \times \mathbb{R} \times [0, R] \times \mathbb{R}^2$*, then for all* $\chi = (\tau, q, x, u_1, u_2, d)$ *and for all* $t \in \tau$*,* $r(t) \in [0, R]$*.*

Our goal is to design a controller that keeps the water level in a given range, $[M_1, M_2]$, with $0 \leq M_1 < M_2]$. This requirement can easily be encoded by a safety property $(Q \cup X, \Box F)$ with

$$F = \mathbf{Q} \times [M_1, M_2] \times \mathbb{R}^3$$

We will try to achieve this goal by treating the situation as a game between (u_1, u_2) and d over the cost function J. Recall that this involves solving the equation:

$$J^*(q_0, x_0) = \max_g \min_d \left(\min_{\chi=(\tau,q,x,(u,d)) \in \mathcal{H}_g} J(\chi) \right)$$

Fortunately, for this example, the equation simplifies considerably.

First, notice that the steam boiler system is deterministic, in the sense that for each initial state and each input sequence consistent with ϕ the automaton accepts a unique execution. In this case, we can represent an execution more compactly by $((q_0, x_0), (u_1, u_2), d)$ with the interpretation that (u_1, u_2) and d represent the entire sequence for these variables. Moreover, if the memoryless controller we pick is single valued, this implies we need not worry about the innermost minimization.

Next notice that J can be encoded by means of two real valued cost functions:

$$J_1(x^0, u_1, u_2, d) = \inf_{t \geq 0} w(t) \quad \text{and} \quad J_1(x^0, u_1, u_2, d) = -\sup_{t \geq 0} w(t) \tag{6.3}$$

Clearly:

$$J = 1 \Leftrightarrow (J_1 \geq M_1) \wedge (J_2 \geq -M_2)$$

The problem of finding a solution to the game over the discrete cost function (known as *qualitative game* or *game of kind*) reduces to finding solutions to two real valued games (known as *quantitative games* or *games of degree*). Even though there is no obvious benefit to doing this, it allows us to use tools from continuous optimal control to address the problem.

Start with the game over J_1. Guess a possible solution:

$$u_i^*(q,x) = 1 \text{ for all } (q,x), \quad \text{and} \quad d^*(q,x) = \begin{cases} D_2 & \text{if } r < R \\ 0 & \text{if } r = R \end{cases} \tag{6.4}$$

Notice that both players resort to a feedback strategy (a trivial one).

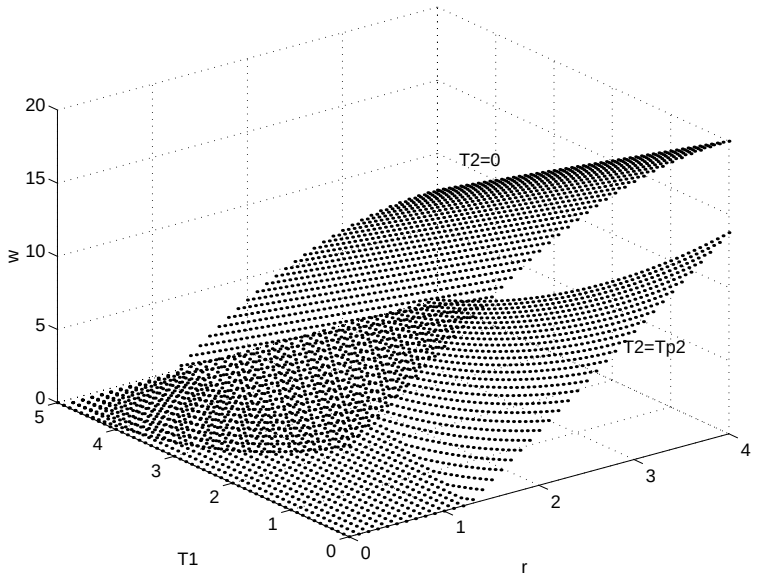


Figure 6.3: Lower limit on w to avoid draining

Lemma 6.1 *(u_1^*, u_2^*, d^*) is globally a saddle solution for the game between (u_1, u_2) and d over J_1.*

Proof: See [58]. ∎

A *saddle solution* is a solution to equation (6.8) for which the order in which the players make their decisions turns out to be unimportant. In other words, a solution for which for all (q, x):

$$J_1^*(q,x) = \max_{(u_1,u_2)} \min_{d} J_1((q,x),(u_1,u_2),d) = \min_{d} \max_{(u_1,u_2)} J_1((q,x),(u_1,u_2),d)$$

Or, in other words, a solution for which for all $(q, x), u_1, u_2$ and d:

$$J_1((q,x),(u_1,u_2),d^*) \leq J_1((q,x),(u_1^*,u_2^*),d^*) \leq J_1((q,x),(u_1^*,u_2^*),d)$$

The last definition is usually somewhat easier to to work with. It is in fact used to prove the lemma.

The saddle cost:

$$J_1^*(q,x) = J_1((q,x),(u_1^*,u_2^*),d^*)$$

Can be computed in closed form. This allows us then to compute the set of states for which there exists a control that for all actions of the disturbance prevents draining. This set turns out to be of the form:

$$W_1^* = \{(q,x) \ : \ J_1^*(q,x) \geq M_1\} = \{(q,x) \ : \ w \geq \hat{w}(r,T_1,T_2)\}$$

Two level sets of this function are shown in Figure 6.3.

The expression for J^* also allows us to compute the least restrictive controller that renders the set W_1^* invariant. It turns out to be unique:

Lemma 6.2 *The feedback controller* g_1^1 *given by:*

$$
\begin{aligned}
&u_1 \in \{0,1\} \text{ and } u_2 \in \{0,1\} \text{ if } [w > \hat{w}(r,0,0)] \vee [w < \hat{w}(r,T_1,T_2)] \\
&u_1 = 1 \text{ and } u_2 \in \{0,1\} \text{ if } \hat{w}(r,0,0) \geq w > \hat{w}(r,T_1,0) \\
&u_1 \in \{0,1\} \text{ and } u_2 = 1 \text{ if } \hat{w}(r,0,0) \geq w > \hat{w}(r,0,T_2) \\
&u_1 = 1 \text{ and } u_2 = 1 \text{ if } w = \hat{w}(r,T_1,T_2)
\end{aligned}
$$

is the unique, least restrictive, non-blocking, feedback controller that renders W_1^{1*} *invariant.*

Proof: See [58]. ∎

Note that the first term applies to states in the interior of the safe set ($w > \hat{w}(r,0,0)$) as well as all the states outside the safe set ($w < \hat{w}(r,T_1,T_2)$). The expression for $\hat{w}$ (see [58]) suggests that $\hat{w}$ is monotone in T_1 and T_2. Therefore, the condition on the last case is enabled if and only if all other conditions fail. The two middle conditions may overlap, however. Therefore there is some nondeterminism in the choice of safe controls (some states may be safe with either one or the other pump on, but not neither).

6.6 Controller Synthesis for Discrete Systems

- Abstractly introduced a gaming technique for determining the maximal controlled invariant subset of a given set and a controller that renders this set invariant.
- Illustrated how the technique can be applied in an example, where the solution to the game can be guessed.
- In the next few lectures we will show how the solution can be constructed more systematically using optimal control tools, more specifically dynamic programming.
- We will start by discussing gaming techniques for purely discrete and purely continuous systems and then show how to extend the results to hybrid systems.

6.6.1 Finite State Machines

Consider a plant automaton $H = (Q,\, X,\, V,\, \mathrm{I}init,\, f,\, I,\, E,\, G,\, R, \phi)$, with:

- $Q = \{q\}$, $\mathbf{Q} < \infty$ (finite discrete states);
- $X = \{x\}$, $\mathbf{X} = \{0\}$ (trivial continuous state);
- $V = \{u, d\}$, $\mathbf{V} < \infty$ (finite inputs);
- $\mathrm{I}init \subseteq \mathbf{Q} \times \{0\}$ (drop the trivial dependence on the continuous state from now on);
- $f(q,x,v) = 0$ (trivial continuous dynamics);
- $I(q) = \emptyset$ (no time evolution);
- $E \subset \mathbf{Q} \times \mathbf{Q}$;

- $G(e) \subseteq \{0\} \times \mathbf{V}$ (drop the trivial dependence on the continuous state from now on);
- $R(e, 0, v) = \{0\}$; and,
- $\phi(q, 0) = \mathbf{V}$.

Notice the similarity between this hybrid automaton and the finite automata considered earlier in the class. Recall that a finite automaton is a collection $(\mathbf{Q}, \Sigma, \Delta, q_0, Q_F)$ consisting of a finite set of states, a finite alphabet, a transition relation, an initial state and a set of final states. Comparing the two definitions we see that $\mathbf{Q}$ plays the same role in both cases, $\mathbf{V}$ plays the role of Σ, Init plays the role of q_0, while $Q_F = \mathbf{Q}$ (restrictions on where we should and should not end up will be encoded in terms of the specification later on). The transition relation is given by:

$$(q, v, q') \in \Delta \Leftrightarrow ((q, q') \in E) \wedge (v \in G(q, q'))$$

Recall that the transition relation can also be encoded by means of a transition map:

$$\delta(q, v) = \{q' \in \mathbf{Q} \ : \ ((q, q') \in E) \wedge (v \in G(q, q'))\}$$

To prevent trivial safe solutions we add a non-blocking assumption:

$$\forall q \in \mathbf{Q}, \ \forall u \in \mathbf{U}, \ \exists d \in \mathbf{D} \ \text{ such that } \delta(q, (u, d)) \neq \emptyset$$

Remarks:

1. The dependence on the trivial continuous state will be dropped to simplify the notation.
2. Strictly speaking the hybrid automaton definition of a finite state system is slightly more general, since many initial states are allowed. In any case, Init will not be very important, since we will be trying to determine the largest set if initial conditions for which a safety specification can be satisfied.
3. All the action takes place at a single point in time (the executions are over time sequences of the form $[0, 0][0, 0], \ldots$). An alternative interpretation is that time has been abstracted away. This can be encoded by setting $I(q) = \{0\}$ for all q; the interpretation is that the transition can take place after an unspecified amount of delay. In either case, the execution can be reduced down to a pair of sequences, one for the states ($q[i]$) and one for the inputs ($v[i]$) such that:
$$q[0] \in \text{I}nit \ \text{ and } \ \forall i \geq 0, q[i+1] \in \delta(q[i], v[i])$$
4. Because time has been abstracted away and $\phi(q) = \mathbf{V}$, the non-blocking assumption eliminates all possibilities of cheating by the controller (zeno executions are meaningless in this setting).
5. Players u and d play simultaneously. Subsumes turn based play [70] and priority play [71] (see [70] and [57]).

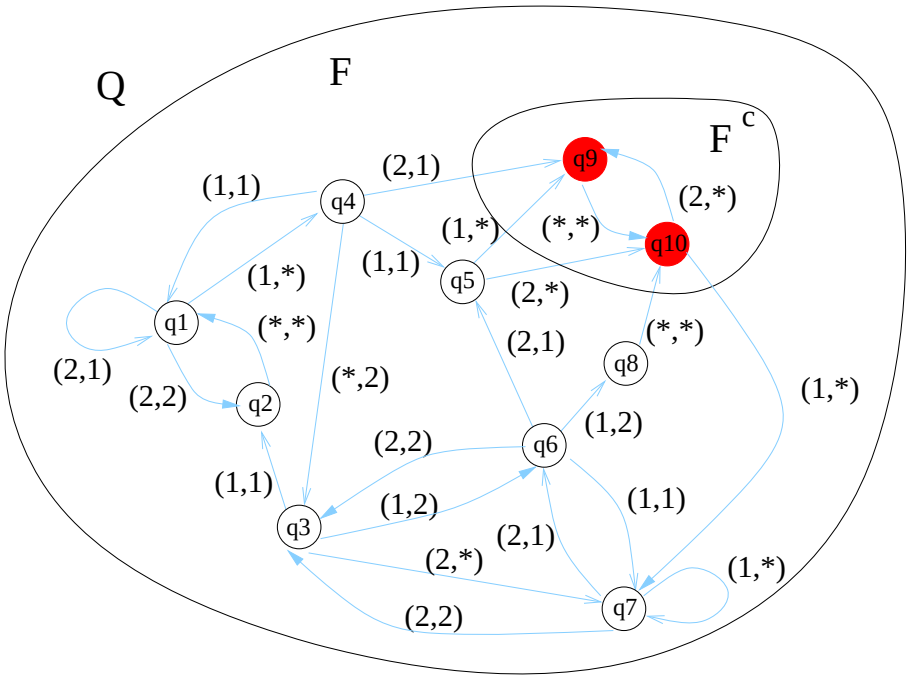


Figure 6.4: Example of discrete controller synthesis

6.6.2 Computation of Controlled Invariant Sets

Consider a set of states $F \subseteq \mathbf{Q}$. Try to establish the largest set of initial states for which there exists a controller that manages to keep all executions inside F. Easiest to demonstrate by means of an example.

Consider a finite automaton with $\mathbf{Q} = \{q_1, \ldots, q_{10}\}$, $\mathbf{U} = \mathbf{D} = \{1, 2\}$, $F = \{q_1, \ldots, q_8\}$, and the transition structure of Figure 6.4 (the inputs are listed in the order (u, d) and * denotes a "wild-card"). Try to establish the largest set of initial conditions, W^* such that there exists a feedback controller that keeps the execution inside F. Clearly:

$$W^* \subseteq F = \{q_1, \ldots, q_8\}$$

Next, look at states that can end up in F in one transition (q_4, q_5, and q_8). Notice that from q_4 if $u = 1$ whatever d chooses to do we remain in F, while from q_5 and q_8 whatever u chooses to do we leave F. Therefore:

$$W^* \subseteq \{q_1, q_2, q_3, q_4, q_6, q_7\}$$

Next look at states that can leave the above set in one transition (q_4 and q_6). Notice that from q_4 if $d = 1$ whatever u chooses we leave the set, while from q_6 if $u = 1$ d can choose $d = 2$ to force us to leave the set while if $u = 2$ d can choose $d = 1$ to force us to leave the set. Therefore:

$$W^* \subseteq \{q_1, q_2, q_3, q_7\}$$

From the remaining states, if u chooses according to the following feedback map:

$$g(q) = \begin{cases} \{1\} & \text{if } q = q_7 \\ \{2\} & \text{if } q \in \{q_1, q_3\} \\ \{1, 2\} & \text{if } q = q_2 \end{cases}$$

we are guaranteed to remain in $\{q_1, q_2, q_3, q_7\}$ for ever. Therefore,

$$W^* = \{q_1, q_2, q_3, q_7\}$$

and g is the least restrictive controller that renders W^* invariant (proof of both facts is by enumeration).

More generally this scheme can be implemented by the following algorithm [60]:

Algorithm 6.0 (Controlled Invariant Set)

Initialization:
$W^0 = F$, $W^1 = \emptyset$, $i = 0$
p **while** $W^i \neq W^{i+1}$ **do**
begin
$W^{i-1} = W^i \cap \{q \in \mathbf{Q} \ : \ \exists u \in \mathbf{U} \ \forall d \in \mathbf{D} \ \delta(x, (u, d)) \subseteq W^i\}$
$i = i - 1$
end

The index decreases as a reminder that the algorithm involves a predecessor operation. This is a real algorithm (can be implemented by enumerating the set of states and inputs). The algorithm terminates after a finite number of steps since:

$$W^{i-1} \subseteq W^i \text{ and } |W^0| = |F| \leq |\mathbf{Q}| < \infty$$

6.6.3 Relation to Gaming

What does any of this have to do with gaming? Introduce a value function:

$$J : \mathbf{Q} \times \mathbb{Z}_- \rightarrow \{0, 1\} \tag{6.5}$$

Consider the difference equation:

$$\begin{array}{rcl} J(q, 0) & = & \begin{cases} 1 & q \in F \\ 0 & q \in F^c \end{cases} \\ J(q, i-1) - J(q, i) & = & \min\{0, \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} [\min_{q' \in \delta(q,(u,d))} J(q', i) - J(q, i)]\} \end{array} \tag{6.6}$$

We will refer to equation (6.18) as a *discrete Hamilton-Jacobi* equation.

Proposition 6.7 (Winning States for u) *A fixed point $J^* : \mathbf{Q} \rightarrow \{0, 1\}$ of (6.18) is reached in a finite number of iterations. The set of states produced by the algorithm is $W^* = \{x \in \mathbf{X} | J^*(x) = 1\}$.*

Proof: We show more generally that $W^i = \{x \in \mathbf{X} | J(x, i) = 1\}$. The proof is by induction (see [58]). ∎

This is beginning to look more game-like. Consider what happens when a fixed point of (6.18) is reached. Ignoring the outermost min for the time being leads to:

$$J^*(q) = \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} \min_{q' \in \delta(q,(u,d))} J^*(q') \tag{6.7}$$

Notice the similarity between this and (excuse the overloading of the notation):

$$J^*(q_0, x_0) = \max_g \min_d \left(\min_{\chi=(\tau,q,x,(u,d))\in\mathcal{H}_g} J(\chi) \right) \tag{6.8}$$

The equations look very similar. The only difference is that instead of having to worry about all feedback controllers, all disturbance trajectories and all possible executions we reduce the problem to a pointwise argument. Clearly equation (6.7) is much simpler to work with that equation (6.8). This is a standard trick in dynamic programming. Equation (6.7) is a special case of what is known in dynamic programming as Bellman's equation, while the difference equation (6.18) is a form of value iteration for computing a fixed point to Bellman's equation.

What about the outer min? This is an inevitable consequence of turning the sequence argument of equation (6.8) to a pointwise argument. It is there to prevent states that have been labeled as unsafe at some point from being relabeled as safe later on. If it were not there, then in the example of Figure 6.4 state q_{10} would be labeled as safe after one step of the algorithm (i.e. $J(q_{10}, -1) = 1$). The extra min operation implements the intersection with W^i in the algorithm of the previous section, ensures the monotonicity of W^i with respect to i and guarantees termination.

The fixed point of equation (6.18) also provides a simple characterization of a least restrictive controller that renders W^* invariant:

$$g(q) = \begin{cases} \left\{ u \in \mathbf{U} \ : \ \min_{d\in\mathbf{D}} \min_{q'\in\delta(q,(u,d))} J^*(q') = 1 \right\} & \text{if } q \in W^* \\ \mathbf{U} & \text{if } q \in (W^*)^c \end{cases}$$

Notice that $g(q) \neq \emptyset$ for all q by construction. Any $u \in \mathbf{U}$ needs to be allowed outside W^* to make the controller least restrictive.

Summarizing:

Proposition 6.8 (Characterization of W^* and g) *W^* is the maximal controlled invariant subset of F and g is the unique, least restrictive feedback controller that renders W^* invariant.*

6.7 Dynamic Programming

Recall that we would like to find functions:

$$(x, u) : \mathbb{R} \to \mathbb{R}^n \times \mathbb{R}^m$$

that minimize:

$$J(x, u) = \phi(x(t_f)) + \int_{t_0}^{t_f} L(x(t), u(t))dt \tag{6.9}$$

subject to:

$$\dot{x}(t) = f(x(t), u(t)), \quad x(t_0) = x_0 \tag{6.10}$$

$$\psi(x(t_f)) = 0 \tag{6.11}$$

where $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $\phi : \mathbb{R}^n \to \mathbb{R}$, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ Lipschitz continuous in x and continuous in u and $\psi : \mathbb{R}^n \to \mathbb{R}$.

In Lecture 24 we discussed how this problem can be related to the solution of a partial differential equation, known as the Hamilton-Jacobi-Bellman equation, by introducing the *value function* (or "cost to go" function), $J^* : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$, given by:

$$J^*(x,t) = \min_{u(\cdot)} \big\{ \phi(x(t_f)) + \int_t^{t_f} L(x(\tau), u(\tau)) d\tau \big\}$$

We argued that if a continuously differentiable solution to the Hamilton-Jacobi-Bellman partial differential equation [22]:

$$\frac{\partial J^*}{\partial t}(x,t) = -\min_{u \in \mathbf{U}} \big\{ \frac{\partial J^*}{\partial x}(x,t) f(x,u) + L(x,u) \big\} \tag{6.12}$$

with boundary condition $J^*(x,0) = \phi(x)$ on $\psi(x) = 0$ can be found, then the optimal controls are given by:

$$u^*(x,t) = \arg\min_{u \in \mathbf{U}} \big\{ \frac{\partial J^*}{\partial x} f(x,u) + L(x,u) \big\}$$

Equation (6.12) can be written more compactly if we introduce the Hamiltonian, $J^* : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$:

$$H(x,p,u) = L(x,u) + p^T f(x,u)$$

and define the optimal Hamiltonian as:

$$H^*(x,p) = \min_{u \in \mathbf{U}} H(x,p,u)$$

Then equation (6.12) becomes:

$$\frac{\partial J^*}{\partial t} = -H^* \left(x, \frac{\partial J^*}{\partial x}(x,t) \right)$$

Notice that we have effectively turned the problem of optimizing a cost over the space of curves (an infinite dimensional vector space), to optimizing a cost pointwise over a finite dimensional vector space. Of course to achieve this we are still required to solve a partial differential equation.

Remarks:

1. The solution is close to the solution obtained through the calculus of variations. Simply replace the co-state p by $\frac{\partial J^*}{\partial x}(x,t)$.

2. On the positive side, dynamic programming (unlike the calculus of variations) inherently leads to feedback solutions.

3. On the negative side, dynamic programming requires one to assume differentiability of the value functions.

4. Differentiability is difficult to guarantee. Even if all the data is "smooth", the solution to the PDE may still develop "corners" (known as *shocks*). The situation is exasperated by the fact that the min operation is continuous but not smooth.

5. Optimal controls are often *bang-bang*. Consider the system:

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \quad \text{(affine in } u\text{)} \\ L &: \mathbb{R}^n \to \mathbb{R} \quad \text{(independent of } u\text{)} \\ u &\in [U_1, U_2] \quad \text{(compact control set)} \end{aligned}$$

Then:

$$H\left(x, \frac{\partial J^*}{\partial x}(x,t), u\right) = L(x) + \frac{\partial J^*}{\partial x}(x,t)f(x) + \frac{\partial J^*}{\partial x}(x,t)g(x)u$$

therefore:

$$u^*(x,t) = \begin{cases} U_1 & \text{if } \frac{\partial J^*}{\partial x}(x,t)g(x) > 0 \\ [U_1, U_2] & \text{if } \frac{\partial J^*}{\partial x}(x,t)g(x) = 0 \\ U_2 & \text{if } \frac{\partial J^*}{\partial x}(x,t)g(x) < 0 \end{cases}$$

Notice that u^* switches between its extreme values whenever $\frac{\partial J^*}{\partial x}(x,t)g(x)$ changes sign. Therefore, even if J^* is continuously differentiable, H^* is continuous, but not continuously differentiable.

6. If $\mathbf{U}$ is not compact, then the optimal controls may be undefined pointwise (consider for example the previous situation with $\mathbf{U} = (-\infty, \infty)$ or $\mathbf{U} = (U_1, U_2)$).

7. The situation may be even worse if $\mathbf{U}$ is not convex. The optimal controls may only be defined in the space of relaxed controls, which are not piecewise continuous as a function of time [86].

8. Finally, both dynamic programming ad the calculus of variations depend on local arguments (small perturbations about the optimal solution). For certain systems these arguments may fail. For example, there may be a curve connecting the initial point to a desired final point, but no neighboring curves have this property. This leads to *abnormal extremals*, which are not captured by the above arguments and have to be studied separately.

6.8 Game Theory

Game theory is related to optimal control, with the difference that there are two player (the control variables are divided into two classes) with possibly conflicting objectives. The simplest case is the *two player, zero sum game*. Consider again a curve:

$$(x, u, d) : \mathbb{R} \to \mathbb{R}^n \times \mathbb{R}^{m_u} \times \mathbb{R}^{m_d}$$

Assume that d is trying to minimize and u is trying to maximize the function:

$$J(x, u, d) = \phi(x(t_f)) + \int_{t_0}^{t_f} L(x(t), u(t), d(t))dt$$

subject to:

$$\dot{x}(t) = f(x(t), u(t), d(t)), \quad x(t_0) = x_0 \tag{6.13}$$

$$\psi(x(t_f)) = 0 \tag{6.14}$$

Definition 6.3 *A pair* (u^*, d^*) *is called a* **saddle equilibrium** *if for all* u, d*:*

$$J(x, u, d^*) \leq J(x, u^*, d^*) \leq J(x, u^*, d)$$

Dynamic programming arguments can also be used to characterize saddle equilibria [13]. As before introduce a Hamiltonian:

$$H(x, p, u, d) = L(x, u, d) + p^T f(x, u, d)$$

Proposition 6.9 *If there exists a continuously differentiable function* $J^* : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ *such that:*

$$\begin{aligned} \frac{\partial J^*}{\partial t}(x,t) &= -\max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} H\left(x, \frac{\partial J^*}{\partial x}(x,t), u, d\right) \\ &= -\min_{d \in \mathbf{D}} \max_{u \in \mathbf{U}} H\left(x, \frac{\partial J^*}{\partial x}(x,t), u, d\right) \\ &= H\left(x, \frac{\partial J^*}{\partial x}(x,t), u^*, d^*\right) \end{aligned}$$

then (u^*, d^*) *is a saddle equilibrium.*

The proof is similar to the one given in the optimal control case, and relies on the fact that by freezing u to u^* we turn the problem to an optimal control problem for d (and vice versa).

Remarks:

1. The above partial differential equation is known as the *Isaacs equation*, and the $\min\max = \max\min$ requirement is known as the *Isaacs condition*.
2. Saddle equilibria do not always exist.
3. A variational formulation is also possible.
4. The same remarks about convexity and compactness of the control and disturbance sets apply.
5. The generalization of the saddle equilibrium concept to multi player games is known as the *Nash equilibrium*. Assume there are N players, each trying to minimize their own cost function:

   $$J_i(x, u_1, \ldots, u_N), \quad i = 1, \ldots, N$$

   Definition 6.4 $(u_1^*, \ldots, u_N^*)$ *is a* **non-cooperative Nash equilibrium** *if for all* i *and for all* u_i*:*

   $$J_i(x, u_1^*, \ldots, u_i, \ldots u_N^*) \geq J_i(x, u_1^*, \ldots, u_i^*, \ldots u_N^*)$$

   The non-cooperative qualification needs to be introduced, since in this setting it may be possible for "gangs" to form: players may be able to improve their returns by collaborating with other players. Note that the saddle equilibrium concept is implicitly non-cooperative, because the game is zero sum.

The solution concept we have in mind for controller synthesis is not as symmetric as the saddle equilibrium, since we give the benefit of the doubt to the disturbance. The interpretation is that the control picks its strategy and lets the disturbance know about it. The disturbance then does its best to damage the controllers plans using this information. Consider a two player game with cost functionals J_1 and J_2 that players 1 and 2 are trying to minimize respectively. Assume that player 1 is the *leader*, i.e. decides on a strategy and lets player 2 know before player 2 has to make a decision. Given a strategy, g_1, for player 1 define the rational responses of player 2 as the set of strategies:

$$R_2(g_1) = \{g \ : \ J_2(g_1, g) \leq J_2(g_1, g') \text{ for all } g_2\}$$

Definition 6.5 *g_1^* is a* **Stackelberg equilibrium strategy** *for the leader if:*

$$\max_{g \in R_2(g_1^*)} J_1(g_1^*, g) = \min_{g_1} \max_{g \in R_2(g_1)} J_1(g_1, g)$$

In general, Stackelberg equilibrium strategies are difficult to compute in feedback form, since the concept is prone to "side payments", "incentives" and "threats". These are all techniques that can be employed by the leader to make it more appealing for player 2 to adopt a certain strategy. Fortunately, the games considered for controller synthesis will be zero sum ($J_1 = -J_2$) which ensures means like that can not be employed and the solution can be computed in feedback form using dynamic programming.

6.9 Controller Synthesis for Continuous Systems

Last time we discussed optimal control and game theoretic problem formulations for continuous dynamical systems. We also highlighted mathematical tools (calculus of variations and dynamic programming) that may be used to solve these problems. This time we will see how these tools (in particular dynamic programming) can be applied to controller synthesis for continuous systems and reachability specifications. Our treatment follows [58] and [76].

Recall that continuous systems are a special class of hybrid systems with:

- $Q = \{q\}$, $\mathbf{Q} = \{q_0\}$ (trivial discrete state);
- $X = \{x\}$, $\mathbf{X} = \mathbb{R}^n$;
- $V = \{u, d\}$, $\mathbf{U} \subseteq \mathbb{R}^{m_u}$ and $\mathbf{D} \subseteq \mathbb{R}^{m_d}$ (no discrete inputs);
- $Init \subseteq \{q_0\} \times \mathbf{X}$ (drop the trivial dependence on the discrete state from now on);
- $f : \mathbf{X} \times \mathbf{U} \times \mathbf{D} \to \mathbb{R}^n$;
- $I(q) = \mathbf{X}$;
- $E = \emptyset$ (no discrete dynamics);
- $\phi(q_0, x) = \mathbf{V}$ (no state dependent input constraints).

Dropping the trivial discrete dynamics, the system can be more compactly characterized by an ordinary differential equation:

$$\begin{array}{rcl} \dot{x}(t) & = & f(x(t), u(t), d(t)) \\ x(0) & = & x_0 \\ u(t) & \in & \mathbf{U} \\ d(t) & \in & \mathbf{D} \end{array} \tag{6.15}$$

To prevent technical problems, we introduce the following assumption:

Assumption 6.1 *f is Lipschitz continuous in x and continuous in u and d. u and d are piecewise continuous as functions of time. $\mathbf{U}$ and $\mathbf{D}$ are convex and compact subsets of $\mathbb{R}^{m_u}$ and $\mathbb{R}^{m_d}$ respectively.*

We use $\mathcal{U}_I$ to denote the set of piecewise continuous functions from an in interval $I \subseteq \mathbb{R}$ to $\mathbf{U}$ and $\mathcal{U}$ for the union of $\mathcal{U}_I$ over all I (and similarly for $\mathcal{D}$). The assumptions on f, u and d are needed to ensure that the system is well posed. Under this assumption the system is (in a sense) deterministic and non-blocking, since, by a standard existence and uniqueness argument for ordinary differential equations, for every $x_0 \in \mathbb{R}^n$, $T \geq 0$, $u \in \mathcal{U}_{[0,T]}$ and $d \in \mathcal{D}_{[0,T]}$, there exists a continuous, piecewise differentiable function $x : [0, T] \to \mathbf{X}$ with $x(0) = x_0$ and $\dot{x}(t) = f(x(t), u(t), d(t))$ for all t where (u, d) is continuous ("almost everywhere"). For this reason we will use:

$$\chi = (x_0, u, d)$$

to denote the unique execution starting at $x_0 \in \mathbb{R}^n$ under control $u \in \mathcal{U}$ and disturbance $d \in \mathcal{D}$.

In addition, the piecewise continuity assumption prevents the controller from "cheating" by forcing the execution to be Zeno. Strictly speaking, the execution of the system will have to be defined over a sequence of intervals, where u and d are continuous over each interval (recall that the definition of an execution of an open automaton from Lecture 8 requires continuity of the input variables over a piece of continuous evolution). Piecewise continuity of u and d implies that there exists an execution such that every compact interval of time overlaps with a finite number of such intervals, or, in other words, there can not be an infinite number of "transitions" (in this case discontinuities in u and d) in a finite amount of time.

The assumptions on $\mathbf{U}$ and $\mathbf{D}$ will be needed to prevent technical problems when posing the optimal control and gaming problems (recall the discussion on bang-bang and relaxed controls)

Consider a set of states $F \subseteq \mathbf{Q}$. Try to establish the maximal control invariant subset of F, i.e. the largest set of initial states for which there exists a controller that manages to keep all executions inside F. Somewhat informally this set can be characterized as:

$$W^* = \{x_0 \in \mathbb{R}^n \ : \ \exists u \in \mathcal{U} \ \forall d \in \mathcal{D}, \ \Box F(x_0, u, d) = \text{True}\}$$

The only additional caveat is that u is implemented by a memoryless controller (following the discussion of Lecture 21). To eliminate technical complications we assume that:

Assumption 6.2 *There exists a continuously differentiable function $l : \mathbb{R}^n \to \mathbb{R}$ such that:*

$$\begin{array}{lll} l(x) > 0 & \text{if} & x \in F^o \\ l(x) = 0 & \text{if} & x \in \partial F \\ l(x) < 0 & \text{if} & x \in F^c \\ l(x) = 0 & \Rightarrow & \frac{\partial l}{\partial x}(x) \neq 0 \end{array}$$

The assumption implies that F is a closed set with non-empty interior, whose boundary is a $n-1$ dimensional manifold.

6.10 Dynamic Programming Solution

To apply the optimal control tools introduced in the previous lecture let $t_f = 0$, consider an arbitrary $t \leq 0$ and introduce the value function:

$$\hat{J}(x,t) = \max_{u \in \mathcal{U}_{[t,0]}} \min_{d \in \mathcal{D}_{[t,0]}} l(x(0))$$

Notice that this optimal control problem involves no Lagrangian ($L \equiv 0$), just a terminal cost. The game obtained in this setting falls in the class of *pursuit-evasion* games [51]. The (u,d) obtained by the above optimal control problem is a Stackelberg equilibrium for the game between control and disturbance, with the control playing the role of the leader. Recall that in general the computation of Stackelberg equilibria in feedback form may be complicated by the possibility of "incentives" and "threats", employed by the leader to coerce the other player into a beneficial course of action. In this case the situation is simplified by the fact that the game is zero sum (any gain achieved by u is equal to a loss suffered by d), so the possibility of incentives and threats is eliminated.

$\hat{J}$ can be computed using the dynamic programming tools discussed in the last two lectures. Introduce a Hamiltonian:

$$\begin{array}{rcl} H : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{m_u} \times \mathbb{R}^{m_d} & \longrightarrow & \mathbb{R} \\ (x,p,u,d) & \longmapsto & p^T f(x,u,d) \end{array}$$

Consider the optimal Hamiltonian:

$$H^*(x,p) = \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} H(x,p,u,d)$$

Notice again that the minimization over u and d is pointwise, as opposed to over functions of time. Then, if $\hat{J}$ is continuously differentiable it satisfies:

$$\begin{array}{rcl} \frac{\partial \hat{J}}{\partial t}(x,t) & = & -H^*\left(x, \frac{\partial \hat{J}}{\partial x}(x,t)\right) \\ \hat{J}(x,0) & = & l(x) \end{array} \tag{6.16}$$

Notice that the evolution of the partial differential equation is "backwards" in time.

Consider the set:

$$\widehat{W}_t = \{x_0 \in \mathbf{X} \ : \ \hat{J}(x_0,t) \geq 0\}$$

This is the set of all states for which starting at $x(t) = x_0$, there exists a controller for u such that for all disturbance trajectories $d \in \mathcal{D}_{[t,0]}$, $l(x(0)) \geq 0$ or, in other words, $x(0) \in F$.

This is not quite what we need yet. We would like the set of all states for which there exists a u such that for all d and for all $t' \in [t, 0]$, $x(t') \in F$. This excludes points in $\widehat{W}_t$ which leave F at some point in $[t, 0]$ but re-enter it before time 0. This requirement can be encoded either after the computation of $\hat{J}$, or by modifying the Hamilton-Jacobi equation:

$$\begin{array}{rcl} \frac{\partial J}{\partial t}(x,t) & = & -H^* \left(x, \frac{\partial J}{\partial x}(x,t)\right) \\ J(x,0) & = & l(x) \end{array} \tag{6.17}$$

Compare this with the discrete Hamilton-Jacobi equation from Lecture 23:

$$\begin{array}{rcl} J(q,0) & = & \left\{ \begin{array}{ll} 1 & q \in F \\ 0 & q \in F^c \end{array} \right. \\ J(q,i-1) - J(q,i) & = & \min\{0, \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} [\min_{q' \in \delta(q,(u,d))} J(q',i) - J(q,i)]\} \end{array} \tag{6.18}$$

$\delta(q,(u,d))$ essentially implements the spatial partial derivative of J along the dynamics of the system. The innermost minimization is not needed in the continuous case, as the continuous system is "deterministic". As before, we seek a stationary solution to this equation. Assume that as $t \to -\infty$, $J(x,t)$ converges to a continuously differentiable function $J^* : \mathbf{X} \to \mathbb{R}$.

Proposition 6.10 *The set $W^* = \{x \in \mathbf{X} \ : \ J^*(x) \geq 0\}$ is the largest controlled invariant set contained in F.*

The solution to the partial differential equation also leads to a least restrictive controller that renders W^* invariant. Consider:

$$g(x) = \left\{ \begin{array}{ll} \left\{ u \in \mathbf{U} \ : \ \min_{d \in \mathbf{D}} \left(\frac{\partial J^*(x)}{\partial x} \right)^T f(x,u,d) \geq 0 \right\} & \text{if } x \in \partial W^* \\ \mathbf{U} & \text{if } x \in (W^*)^\circ \cup (W^*)^c \end{array} \right. \tag{6.19}$$

Proposition 6.11 *g is the unique, least restrictive memoryless controller that renders W^* invariant.*

Notice that no constraint is imposed on u in the interior of W^* (we can delay action until we reach the boundary) and outside W^* (its too late to do anything about it, so might as well give up!).

6.11 Geometric interpretation

For an arbitrary time $t \leq 0$ define:

$$W_t = \{x \in \mathbf{X} \ : \ J(x,t) \geq 0\}$$

Consider and $x \in \partial W_t$ and assume that:

$$\begin{aligned}
& H^*\left(x, \frac{\partial J}{\partial x}(x,t)\right) < 0 \\
\Leftrightarrow \quad & \max_{u\in\mathbf{U}} \min_{d\in\mathbf{D}} H\left(x, \frac{\partial J}{\partial x}(x,t), u, d\right) < 0 \\
\Leftrightarrow \quad & \max_{u\in\mathbf{U}} \min_{d\in\mathbf{D}} \frac{\partial J}{\partial x}(x,t) f(x,u,d) < 0 \\
\Leftrightarrow \quad & \forall u \in \mathbf{U} \;\; \exists d \in \mathbf{D} \text{ such that } \frac{\partial J}{\partial x}(x,t) f(x,u,d) < 0
\end{aligned}$$

But $\frac{\partial J}{\partial x}(x,t)$ is the normal to the boundary of W_t at x, pointing inside W_t. Moreover, $\frac{\partial J}{\partial x}(x,t) f(x,u,d)$ is the inner product between this normal and the vector $f(x,u,d)$. Let θ be the angle between $\frac{\partial J}{\partial x}(x,t)$ and $f(x,u,d)$. Then:

$$\begin{aligned}
\frac{\partial J}{\partial x}(x,t) f(x,u,d) > 0 \quad & \text{if} \quad \theta < \pi/2 \\
\frac{\partial J}{\partial x}(x,t) f(x,u,d) = 0 \quad & \text{if} \quad \theta = \pi/2 \\
\frac{\partial J}{\partial x}(x,t) f(x,u,d) < 0 \quad & \text{if} \quad \theta > \pi/2
\end{aligned}$$

Therefore, the above statement is equivalent to:

> for all $u \in \mathbf{U}$ there exists $d \in \mathbf{D}$ such that the normal to ∂W_t at x pointing towards the interior of W_t makes an angle greater than $\pi/2$ with $f(x,u,d)$,

or, equivalently:

> for all $u \in \mathbf{U}$ there exists $d \in \mathbf{D}$ such that $f(x,u,d)$ points outside W_t.

These are points where whatever u does d can force them to leave the set W_t instantaneously. Notice that the order of the quantifiers in the above expression implies that d may depend on u, in addition to x and t. The part of the boundary of W_t where $H^* < 0$ is known as the "usable part" in the pursuit-evasion game literature.

Returning back to the Hamilton Jacobi equation, we see that for these points:

$$\begin{aligned}
\frac{\partial J}{\partial t}(x,t) &= -\min\left\{0, H^*\left(x, \frac{\partial J}{\partial x}(x,t)\right)\right\} \\
&= -H^*\left(x, \frac{\partial J}{\partial x}(x,t)\right) \\
&> 0
\end{aligned}$$

Therefore, as t decreases, J also decreases. For these points on the boundary of W_t, J becomes negative instantaneously, and they "fall out of" W_t.

What if $H^* \geq 0$? A similar argument shows that in this case:

> there exists $u \in \mathbf{U}$ such that for all $d \in \mathbf{D}$ the normal to ∂W_t at x pointing towards the interior of W_t makes an angle at most $\pi/2$ with $f(x,u,d)$,

or, equivalently:

> there exists $u \in \mathbf{U}$ such that for all $d \in \mathbf{D}$, $f(x,u,d)$ either points inside W_t or is tangent to ∂W_t.

These are points for which there exists a choice or u that for all d forces the state to remain in W_t. Notice that the order of the quantifiers implies that u may only depend x and t, and not d. For these points:

$$\begin{aligned} \frac{\partial J}{\partial t}(x,t) &= -\min\left\{0, H^*\left(x, \frac{\partial J}{\partial x}(x,t)\right)\right\} \\ &= 0 \end{aligned}$$

Therefore, as t decreases, J remains constant. These are points that want to move towards the interior of W_t. The role of the outermost minimum is to ensure that the value of J does not increase for these points, so that W_t does not grow. This is to prevent states that have been labeled as unsafe (can reach F^c) from being relabeled as safe later on.

6.12 Controller Synthesis for Hybrid Systems

Last two weeks we discussed controller synthesis for:

- Discrete systems: design characterized by a fixed point of a difference equation.
- Continuous systems: design characterized by fixed point of a partial differential equation.

In either case the solution is characterized as a fixed point of an equation, which we refer to as the Hamilton-Jacobi equation. Today, we bring the discrete and continuous parts together, and talk about hybrid systems. Our treatment follows [76] and [58]. See also [79].

Recall that the *plant* is modeled by an open hybrid automaton, $H = (Q, X, V, \mathrm{I}nit, f, I, E, G, R, \phi)$, where:

- Q is a finite collection of discrete state variables;
- X is a finite collection of continuous state variables;
- V is a finite collection of input variables. We assume $V = V_D \cup V_C$, where V_D contains discrete and V_C contains continuous variables.
- $\mathrm{I}nit \subseteq \mathbf{Q} \times \mathbf{X}$ is a set of initial states;
- $f : \mathbf{Q} \times \mathbf{X} \times \mathbf{V} \to \mathbb{R}^n$ is an input dependent vector field;
- $I : \mathbf{Q} \to 2^{\mathbf{X} \times \mathbf{V}}$ assigns to each $q \in \mathbf{Q}$ an input dependent invariant set;
- $E \subset \mathbf{Q} \times \mathbf{Q}$ is a collection of discrete transitions;
- $G : E \to 2^{\mathbf{X} \times \mathbf{V}}$ assigns to each $e = (q, q') \in E$ a guard;

- $R : E \times \mathbf{X} \times \mathbf{V} \to 2^{\mathbf{X}}$ assigns to each $e = (q, q') \in E$, $x \in \mathbf{X}$ and $v \in \mathbf{V}$ a reset relation; and,
- $\phi : \mathbf{Q} \times \mathbf{X} \to 2^{\mathbf{V}}$ assigns to each state a set of admissible inputs.

Also recall that the set the input variables are partitioned into *controls* (that can be used to steer the system) and *disturbances* (whose values can not be controlled):

$$V = U \cup D$$

Assumption 6.3 *To avoid technical problems we assume that:*

1. *f is Lipschitz continuous in x and continuous in v.*
2. *for all $(q, x) \in \mathbf{Q} \times \mathbf{X}$, $\phi(q, x) = \mathbf{U} \times \mathbf{D} \neq \emptyset$.*
3. *for all $q \in \mathbf{Q}$ and for all $v \in \mathbf{V}$, $I(q)|_X$ is an open set.*
4. *for all $(q, x) \in \mathbf{Q} \times \mathbf{X}$, and for all $u \in \mathbf{U}$ there exists $d \in \mathbf{D}$ such that:*

$$[(x, u, d) \in I(q)] \vee [((x, u, d) \in G(q, q') \wedge (R(q, q', x, u, d) \neq \emptyset)]$$

Part 1 is standard, and is needed for existence of continuous evolution. Part 2 implies that acceptable control and disturbance inputs always exists, and that the choice of u can not influence the possible choices of d and vice versa. Part 3 implies that we need not worry about what happens on the boundary of the invariant set (otherwise we would have to assume transverse invariants, define the Out set, etc.). Finally, part 4 (together with part 3) implies that the controller can not block the system execution. Notice that this assumption is not symmetric for u and d. The reason is that, since we are dealing with safety specifications it will never be to the benefit of d to stop the execution.

As before we will try to establish the largest controlled invariant subset of a given set $F \subseteq \mathbf{Q} \times \mathbf{X}$, and design a controller that renders this set invariant. To avoid technical problems we assume that:

Assumption 6.4 *F is a closed set.*

We will again take an adverserial approach and treat the design as a game between u and d. Whenever possible we will give the advantage to d, in particular:

1. In the order of play (u will be the "leader" of the game).
2. When resolving non-determinism.

6.13 Definitions of Operators

Notice that the disturbance has two choices. It can:

1. Try to make the system "jump" outside F.

2. Try to "steer" the system outside F along continuous evolution.

The control also has two choices:

1. Try to "jump" to another state in F when the disturbance tries to steer out of F.
2. Try to "steer" the system and keep it in F along continuous evolution.

To charactrize alternative 1 for the disturbance, introduce the *uncontrollable predecessor operator*, $\mathrm{P}re_d : 2^{\mathbf{Q}\times\mathbf{X}} \to 2^{\mathbf{Q}\times\mathbf{X}}$, which, given a set $K \subseteq \mathbf{Q}\times\mathbf{X}$ returns:

$$\begin{aligned}\mathrm{P}re_d(K) = K^c \cup \{(q,x) \in \mathbf{Q}\times\mathbf{X} \ : \quad & \forall u \in \mathbf{U}\ \exists d \in \mathbf{D}, q' \in \mathbf{Q} \text{ such that} \\ & [(x,u,d) \in G(q,q')] \wedge [R(q,q',x,u,d) \cap K^c \neq \emptyset]\}\end{aligned}$$

For a given K, $\mathrm{P}re_d(K)$ returns the set of states that are either in the complement of K, or whatever u does may find themselves in the complement of K by a discrete transition.

To charactrize alternative 1 for the control, introduce the *controllable predecessor operator*, $\mathrm{P}re_u : 2^{\mathbf{Q}\times\mathbf{X}} \to 2^{\mathbf{Q}\times\mathbf{X}}$, which, given a set $K \subseteq \mathbf{Q}\times\mathbf{X}$ returns:

$$\begin{aligned}\mathrm{P}re_u(K) = K \cap \{(q,x) \in \mathbf{Q}\times\mathbf{X} \ : \quad & \exists u \in \mathbf{U} \text{ such that } \forall d \in \mathbf{D}, \\ & [\exists q' \in \mathbf{Q} \text{ such that } (x,u,d) \in G(q,q')] \wedge \\ & [(x,u,d) \notin I(q)] \wedge \\ & [(x,u,d) \in G(q,q') \rightarrow R(q,q',x,u,d) \subseteq K]\}\end{aligned}$$

For a given K, $\mathrm{P}re_u(K)$ returns the set of states that are already in K and there exists a control action that can force a transition that keeps the state in K.

Some simple facts about these two operators:

Proposition 6.12 *For all* $K \subseteq \mathbf{Q}\times\mathbf{X}$, $\mathrm{P}re_u(K) \subseteq K$, $\mathrm{P}re_d(K) \supseteq K^c$ *and* $\mathrm{P}re_u(K) \cap \mathrm{P}re_d(K) = \emptyset$

Remarks:

- The two operators are asymmetric.
- The order of the quantifiers is consistent with u being the leader in the game.
- Since all non-determinism is resolved in favor of d, u has to work harder:
  - it has to ensure a transition back into K exists (first condition in the definition of $\mathrm{P}re_u(K)$),
  - it has to be able to "force" the transition (no mention of $I(q)$ in the definition of $\mathrm{P}re_d(K)$),
  - it has to ensure all possible transitions stay in K (last condition in the definition of $\mathrm{P}re_u(K)$).

Finally, to characterize alternative 2 for bot u and d introduce the *reach-avoid operator*, $\mathrm{P}re_u : 2^{\mathbf{Q}\times\mathbf{X}} \times 2^{\mathbf{Q}\times\mathbf{X}} \to 2^{\mathbf{Q}\times\mathbf{X}}$, which, given two disjoint sets $K \subseteq \mathbf{Q}\times\mathbf{X}$ and $L \subseteq \mathbf{Q}\times\mathbf{X}$ returns:

$$\begin{aligned}\mathrm{R}each(K,L) = \{(q_0,x_0) \in \mathbf{Q}\times\mathbf{X} \ : \quad & \forall u \in \mathcal{U}\ \exists d \in \mathcal{D}, t \geq 0 \text{ such that} \\ & [(q(t),x(t)) \in K)] \wedge [\forall t' \in [0,t]\ ((q(t'),x(t')) \notin L]\end{aligned}$$

where $(q, x) : [0, t] \to \mathbf{Q} \times \mathbf{X}$ is a segment of continuous evolution with inputs u and d, i.e.:

$$\begin{aligned} (q(0), x(0)) &= (q_0, x_0) \text{ and } \forall t' \in [0, t] \\ q(t') &= q_0 \\ (x(t'), u(t'), d(t')) &\in I(q_0) \\ \dot{x}(t') &= f(q_0, x(t'), u(t'), d(t')) \end{aligned}$$

Given two disjoint sets K and L, the operator R*each* returns the set of states for which whatever u does, d can chose an appropriate trajectory to bring the state of the system to K along continuous evolution, without going first through L.

Proposition 6.13 *For all $K, L \subseteq \mathbf{Q} \times \mathbf{X}$ with $K \cap L = \emptyset$, $K \subseteq \mathrm{R}each(K, L) \subseteq L^c$.*

Notice that the definition of R*each* is somewhat informal, since:

- u is implicitly assumed to be a feedback strategy (in fact, without loss of generality a *memoryless* feedback strategy, see Lecture 21).
- u and d are allowed to be piecewise continuous (recall the justification for this given in Lecture 26).

6.14 Basic Algorithm

Using the above definitions, the following algorithm can now be formulated for computing the largest controlled invariant subset of a given set F.

Algorithm 6.1 (Controlled Invariant Set)

Initialization:
$W^0 = F$, $W^1 = \emptyset$, $i = 0$
while $W^i \neq W^{i+1}$ **do**
begin
$W^{i-1} = W^i \setminus \mathrm{R}each(\mathrm{P}re_d(W^i), \mathrm{P}re_u(W^i))$
$i = i - 1$
end

Pictorially, the operations involved in one step of the algorithm is illustrated in Figure 6.5.

Proposition 6.14 *If the algorithm terminates in a finite number of steps, the fixed point W^* is the maximal controlled invariant subset of F.*

Proof: Clearly, $W^* \subseteq \ldots \subseteq W^{i-1} \subseteq W^i \subseteq \ldots \subseteq F$. Assume that the algorithm terminates in a finite number of steps. Then:

$$\begin{aligned} & W^* = W^* \setminus \mathrm{R}each(\mathrm{P}re_d(W^*), \mathrm{P}re_u(W^*)) \\ \Leftrightarrow \quad & W^* \cap \mathrm{R}each(\mathrm{P}re_d(W^*), \mathrm{P}re_u(W^*)) \\ \Leftrightarrow \quad & \mathrm{R}each(\mathrm{P}re_d(W^*), \mathrm{P}re_u(W^*)) \subseteq (W^*)^c \end{aligned}$$

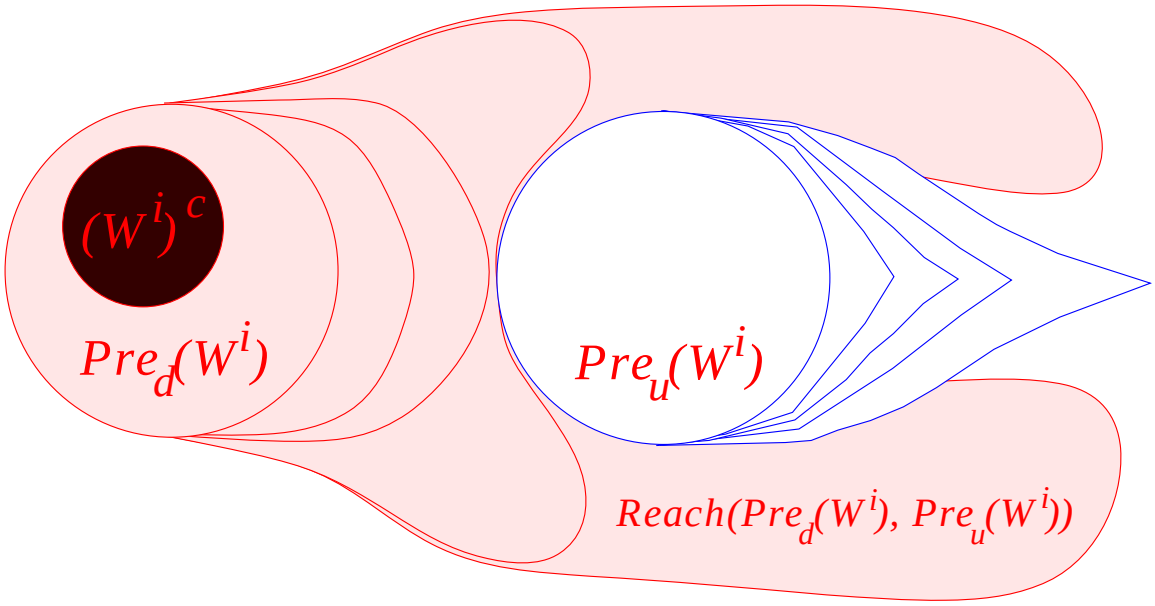


Figure 6.5: One step of the algorithm.

But:

$$\begin{aligned} & (W^*)^c \subseteq \mathrm{P}re_d(W^*) \subseteq \mathrm{R}each(\mathrm{P}re_d(W^*), \mathrm{P}re_u(W^*)) \subseteq (W^*)^c \\ \Leftrightarrow \quad & \mathrm{R}each(\mathrm{P}re_d(W^*), \mathrm{P}re_u(W^*)) = \mathrm{P}re_d(W^*) = (W^*)^c \end{aligned}$$

Consider an arbitrary $(q_0, x_0) \in W^*$. Then:

$$(q_0, x_0) \notin \mathrm{R}each(\mathrm{P}re_d(W^*), \mathrm{P}re_u(W^*))$$

Taking the negation of the expression for R$each$ this becomes:

$\exists u \in \mathcal{U}$ such that $\forall d \in \mathcal{D}, \forall t \geq 0$ $[(q(t), x(t)) \notin \mathrm{P}re_d(W^*))] \vee [\exists t' \in [0, t]$ $((q(t'), x(t')) \in \mathrm{P}re_u(W^*)]$

Replacing $\mathrm{P}re_d(W^*)$ by $(W^*)^c$ and substituting the definition of $\mathrm{P}re_u(W^*)$:

$$\begin{aligned} \exists u \in \mathcal{U} \text{ such that } \forall d \in \mathcal{D},\ \forall t \geq 0 \quad & [(q(t), x(t)) \in W^*] \vee \\ & [\exists t' \in [0, t] \text{ such that } \exists u(t') \in \mathbf{U} \text{ such that } \forall d(t') \in \mathbf{D}, \\ & (\exists q' \in \mathbf{Q} \text{ such that } (x(t'), u(t'), d(t')) \in G(q(t'), q')) \wedge \\ & ((x(t'), u(t'), d(t')) \notin I(q(t'))) \wedge \\ & ((x(t'), u(t'), d(t')) \in G(q(t'), q') \rightarrow R(q(t'), q', x(t'), u(t'), d(t')) \subseteq W^*)] \end{aligned}$$

In other works, if $(q_0, x_0) \in W^*$ either u can keep the system forever in W^* without any discrete transitions taking place, or it can drive the system so that $(q(\tau_1), x(\tau_1)) \in W^*$ (the first discrete transition the state is again in W^*). Controlled invariance of W^* follows by induction.

To show maximality, consider an arbitrary $(q_0, x_0) \in (W^*)^c$. Then, since the algorithm terminated in a finite number of steps, there exists i such that:

$$\begin{aligned} & (q_0, x_0) \in W^i \setminus W^{i-1} \\ \Leftrightarrow \quad & (q_0, x_0) \in \mathrm{R}each(\mathrm{P}re_d(W^i), \mathrm{P}re_u(W^i)) \\ \Leftrightarrow \quad & \forall u \in \mathcal{U}\ \exists d \in \mathcal{D}, t \geq 0 \text{ such that } [(q(t), x(t)) \in \mathrm{P}re_d(W^i)] \wedge [\forall t' \in [0, t],\ ((q(t'), x(t')) \notin \mathrm{P}re_u(W^i)] \end{aligned}$$

Substituting the definition of $\mathrm{P}re_d$ leads to:

$$\begin{array}{ll} \forall u \in \mathcal{U}\ \exists d \in \mathcal{D},\ \exists t \geq 0 \text{ such that} & [\forall t' \in [0,t]\ ((q(t'),x(t')) \notin \mathrm{P}re_u(W^i)] \wedge \\ & [((q(t),x(t)) \in (W^i)^c) \vee \\ & (\forall u(t) \in \mathbf{U}\ \exists d(t) \in \mathbf{D},\ q' \in \mathbf{Q} \text{ such that } [(x(t),u(t),d(t)) \in G(q(t),q')] \wedge \\ & [R(q(t),q',x(t),u(t),d(t)) \cap (W^i)^c \neq \emptyset]) \end{array}$$

Since $W^j \subseteq W^i$, for all $j \leq i$, the above shows that in finite time the state may end up either in F^c or in W^j for some $j > i$ whatever the control does (at best it ends up in W^{i+1}). Therefore, by induction, the state leaves F in finite time. ∎

Chapter 7

Controller Computations for Hybrid Systems

7.1 Controller Synthesis Algorithm Review

Last time an algorithm to compute the largest controlled invariant set contained in a given set F was introduced:

Algorithm 7.0 (Controlled Invariant Set)
Initialization:
$W^0 = F$, $W^1 = \emptyset$, $i = 0$
while $W^i \neq W^{i+1}$ **do**
begin
$W^{i-1} = W^i \setminus \mathrm{R}each(\mathrm{P}re_d(W^i), \mathrm{P}re_u(W^i))$
$i = i - 1$
end

where the index decreases as a reminder that the algorithm involves a predecessor operation. Recall that $\mathrm{P}re_d(W^i)$ is the set of states that are either already in the complement of W^i, or may find themselves in the complement of W^i through a discrete transition, whatever u does through an appropriate choice of d. $\mathrm{P}re_u(W^i)$, on the other hand, is the set of states that are already in W^i and there exists a u that can force a transition to another state in W^i, whatever d does. Finally, $\mathrm{R}each(\mathrm{P}re_d(W^i), \mathrm{P}re_u(W^i))$ is the set of states that whatever u does there exists an appropriate response by d that drives the system to $\mathrm{P}re_d(W^i)$ without going through $\mathrm{P}re_u(W^i)$ along continuous evolution.

7.2 Computation

This is a pseudo-algorithm, in the sense that all the operators used were defined axiomatically. Therefore, even though conceptually the algorithm will produce the largest controlled invariant set, it can not be implemented as it is in practice. To effectively implement it and guarantee termination we need to:

1. be able to store arbitrary sets of states and perform operations on them.

2. compute $\mathrm{P}re_u$ and $\mathrm{P}re_d$ at each step.

3. compute R*each* at each step.

4. ensure that only a finite number of steps are needed.

Finite sets can be stored and manipulated by enumeration. The situation is not as simple for continuous quantities however. Here we will assume that sets of states are stored as level sets of functions. This representation is convenient conceptually, but is not necessarily effective computationally, as it shifts the problem from storing sets to storing functions! For certain classes of sets/functions computation is in fact possible (for example, for polygons, ellipsoids, polynomials, functions encoded by neural networks, etc.). In this lecture we follow [77].

The computation of $\mathrm{P}re_u$ and $\mathrm{P}re_d$ is straightforward conceptually, as it involves inverting the discrete transition relation. The way to implement it computationally depends on the representation one uses to store and manipulate sets, and the sets that encode the discrete transitions (invariants, guards, reset relation). For certain classes (e.g. linear maps, polygonal sets, etc.) the computation may be feasible using quantifier elimination or optimal control tools.

Obtaining a representation of R*each* is not only computationally involved, but also conceptually. The standard optimal control and game theory concepts do not deal with this type of problem, where a certain set must be reached before another. Since q remains constant along continuous evolution, the computation of R*each* can be performed separately for each value of $q \in \mathbf{Q}$. The computation will involve a modification of the Hamilton-Jacobi partial differential equation. Freeze the value of q and consider two sets $K \subseteq \mathbf{X}$ and $L \subseteq \mathbf{X}$. Assume that:

Assumption 7.1 *Assume L is closed, K is open and there exist continuously differentiable functions $l_L : \mathbf{X} \to [0,1]$ and $l_K : \mathbf{X} \to [0,1]$ such that $l_K(x) = 0$ for $x \in L$, $l_K > 0$ for $x \notin L$ and $l_K(x) = 1$ for $x \in \overline{K}$, whereas $l_L(x) = 0$ for $x \in \overline{K}$, $l_L > 0$ for $x \notin \overline{K}$ and $l_L(x) = 1$ for $x \in L$.*

This assumption is satisfied if, for example, the sets have smooth boundaries, non-empty interiors and their closures are disjoint.

Consider the value function $J : \mathbb{R}^n \times \mathbb{R}_- \to \mathbb{R}$ governed by the Hamilton-Jacobi equation:

$$\begin{array}{rcl} -\frac{\partial J(x,t)}{\partial t} & = & \left\{ \begin{array}{ll} \max\left\{0, H^*(x, \frac{\partial J(x,t)}{\partial x}^T)\right\} & \text{if } J(x,t) \geq 1 \\ H^*(x, \frac{\partial J(x,t)}{\partial x}^T) & \text{if } -1 < J(x,t) < 1 \\ \min\left\{0, H^*(x, \frac{\partial J(x,t)}{\partial x}^T)\right\} & \text{if } J(x,t) \leq -1 \end{array} \right. \\ H^*(x,p) & = & \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} p^T f(q,x,u,d) \\ J(x,0) & = & l_L(x) - l_K(x) \end{array} \tag{7.1}$$

Notice that:

Proposition 7.1 *If for some $x \in \mathbf{X}$, $J(x,t) \leq -1$ (or $J(x,t) \geq 1$) for some $t \leq 0$ then $J(x,t') \leq -1$ (or $J(x,t') \geq 1$) for all $t' \leq t$.*

The situation $J(x,t) \leq -1$ corresponds to points for which whatever u does, there exists a response for d such that the state reaches K by time $|t|$, without going through L for any $0 \leq s < |t|$. The situation $J(x,t) \geq 1$ corresponds to points for which there exists a choice for u such that whatever d does the state reaches L by time $|t|$ without going through K for any $0 \leq s < |t|$. Finally, the situation $J(x,t) \in (-1,1)$ correspond to states that are still "undecided" at time t: they may end up in K or in L by some time $|s| > |t|$ or they may never reach either set. Notice that the situation is not quite symmetric because of the order of the quantification. As usual we are looking for a stationary solution to this PDE.

Proposition 7.2 *If a continuously differentiable solution to the above equations exists that converges to a continuously differentiable $J^* : \mathbb{R}^n \to \mathbb{R}$ as $t \to -\infty$, then* $\mathrm{R}each(K,L) = \{x \in \mathbb{R}^n | J^*(x) < -1\}$.

In the limit, the set where $J^*(x) < -1$ consists of states that can ultimately be driven to K without first going through L, the set where $J^*(x) \geq 1$ consists of states that can ultimately be driven to L without first going through K, while the set of states where $J^*(x) \in [-1,1)$ contains states that can not be forced to reach neither K nor L.

An alternative characterization of $\mathrm{R}each(K,L)$ involves two coupled partial differential equations. Assume continuously differentiable function $l_K : \mathbf{X} \to \mathbb{R}$ and $l_L : \mathbf{X} \to \mathbb{R}$ exists such that $K = \{x \in \mathbf{X} | l_K(x) < 0\}$ and $L = \{x \in \mathbf{X_C} | l_L(x) \leq 0\}$. Consider the following system of coupled Hamilton-Jacobi equations:

$$-\frac{\partial J_K(x,t)}{\partial t} = \begin{cases} H_K^*(x, \frac{\partial J_K(x,t)}{\partial x}) & \text{for } \{x \in X \mid J_K(x,t) > 0\} \\ \min\{0, H_K^*(x, \frac{\partial J_K}{\partial x}(x,t))\} & \text{for } \{x \in X \mid J_K(x,t) \leq 0\} \end{cases} \tag{7.2}$$

and

$$-\frac{\partial J_L}{\partial t}(x,t) = \begin{cases} H_L^*(x, \frac{\partial J_L(x,t)}{\partial x}) & \text{for } \{x \in X \mid J_L(x,t) > 0\} \\ \min\{0, H_L^*(x, \frac{\partial J_L(x,t)}{\partial x})\} & \text{for } \{x \in X \mid J_L(x,t) \leq 0\} \end{cases} \tag{7.3}$$

where $J_K(x,t) = l_K(x(0))$ and $J_L(x,t) = l_L(x(0))$, and

$$H_K^*(x, \frac{\partial J_K}{\partial x}) = \begin{cases} 0 & \text{for } \{x \in X \mid J_L(x,t) \leq 0\} \\ \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} \frac{\partial J_K}{\partial x} f(x,u,d) & \text{otherwise} \end{cases} \tag{7.4}$$

$$H_L^*(x, \frac{\partial J_L}{\partial x}) = \begin{cases} 0 & \text{for } \{x \in X \mid J_K(x,t) \leq 0\} \\ \min_{u \in \mathbf{U}} \max_{d \in \mathbf{D}} \frac{\partial J_L}{\partial x} f(x,u,d) & \text{otherwise} \end{cases} \tag{7.5}$$

Equation (7.2) describes the evolution of the set K under the Hamiltonian H_K^*. This is the solution to the "$\max_u \min_d$" game for reachability in purely continuous systems, with the modification that $H_K^* = 0$ in $\{x \in \mathbf{X_C} \mid J_L(x,t) \leq 0\}$. This ensures that the evolution of $J_K(x,t)$ is frozen once this set is reached. Similarly, equation (7.3) describes the evolution of the set L under the Hamiltonian H_L^*. Here a "$\min_u \max_d$" is used, since it is assumed that the control tries to push the system into L, to escape from K. $H_L^* = 0$ in $\{x \in \mathbf{X_C} \mid J_K(x,t) \leq 0\}$ to ensure that the evolution of $J_L(x,t)$ is frozen once this set is reached. One can show that the resulting set $\{x \in \mathbf{X_C} \mid J_K(x,t) < 0\}$ contains neither L nor states for which there is a control which drives the system into L; and the set $\{x \in \mathbf{X_C} \mid J_L(x,t) < 0\}$ contains neither K nor states for which there is a disturbance which drives the system into K.

Proposition 7.3 *Assume that $J_K(x,t)$ ($J_L(x,t)$ respectively) satisfies the Hamilton-Jacobi equation (7.2) ((7.3) respectively), and that it converges as $t \to -\infty$ to a function $J_K^*(x)$ ($J_L^*(x)$ respectively). Then, Reach$(K,L) = \{x \in \mathbf{X_C} \mid J_K^*(x) < 0\}$.*

As for the first representation of R*each*, comments relating the values of J_K^* and J_L^* and the "fate" of the corresponding states can be made.

Remarks:

- The first representation is more compact. It involves solving a PDE in $n+1$ dimensions, as opposed to the $2n+1$ dimensions needed for the second representation.
- The second representation is more general. For example, it does not require the closures of K and L to be disjoint. This often turns out to be needed in applications.
- In either case, we are still faced with the problem of finding a solution to PDEs. Technical problems arise since a solution in the conventional sense does not always exist or may not be unique. For example, weaker solutions concepts (e.g. viscosity solutions) may be needed if the value function turns out not to be smooth.
- Even if we decide on an appropriate solution concept, obtaining the solution computationally may still be a problem. Computation tools that may be applicable include:
  - finite element methods
  - method of characteristics
  - level set methods (these were studied in detail in the doctoral dissertation of Tomlin [79].
  - approximate solution using basis functions
  - viability kernel computations

  Each of these techniques has computational advantages and disadvantages and may be applicable in some problems but not others. They all suffer from the "curse of dimensionality": they typically run out of steam beyond $\mathbb{R}^3$.
- Problems may arise even after a numerical solution is computed:
  - will the numerical approximation be contained in the controlled invariant set?
  - will the numerical approximation be controlled invariant itself?
  - can a controller that renders the numerical approximation controlled invariant be synthesized?
  - or at least, can a controller that renders the largest controlled invariant set invariant be synthesized using the approximation?

7.3 Example: Aircraft Conflict Resolution

Consider from the dissertation of C. Tomlin [79], a collision avoidance maneuver for a pair of aircraft consisting of two modes of operation: a *cruise* mode in which both aircraft follow a straight path, and an *avoid* mode in which both aircraft follow a circular arc path. When

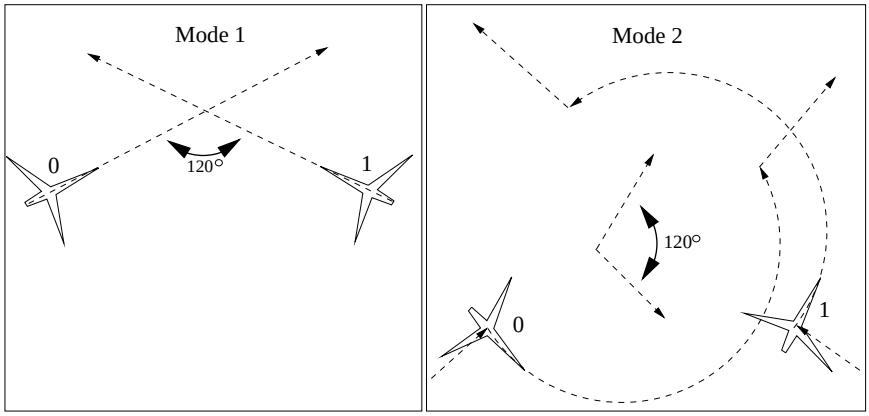


Figure 7.1: The two modes of operation.

the maneuver is initiated, each aircraft turns 90° to its right and follows a half circle. Once the half circle is complete, each aircraft returns to its original heading and continues on its straight path (Figure 9.8).

In each mode, the continuous dynamics may be expressed in terms of the *relative motion* of the two aircraft (equivalent to fixing the origin of the relative frame on aircraft 0 and studying the motion of aircraft 1 with respect to aircraft 0):

$$\begin{aligned} \dot{x}_r &= -v_0 + v_1 \cos\psi_r + \omega_0 y_r \\ \dot{y}_r &= v_1 \sin\psi_r - \omega_0 x_r \\ \dot{\psi}_r &= \omega_1 - \omega_0 \end{aligned} \tag{7.6}$$

in which $(x_r, y_r, \psi_r) \in \mathbb{R}^2 \times [-\pi, \pi]$ is the relative position and orientation of aircraft 1 with respect to aircraft 0, and v_i and ω_i are the linear and angular velocities of each aircraft. In the cruise mode $\omega_i = 0$ for $i = 0, 1$ and in the avoid mode $\omega_i = 1$ for $i = 0, 1$. The control is the linear velocity of aircraft 0, $u = v_0 \in \mathbf{U}$, and the disturbance is the linear velocity of aircraft 1, $d = v_1 \in \mathbf{D}$, where $\mathbf{U}$ and $\mathbf{D}$ denote the range of possible linear velocities of each aircraft. Here we restrict our attention to the case where both $\mathbf{U}$ and $\mathbf{D}$ are singletons (i.e. the aircraft maintain the same speed).

The discrete state takes on three possible values, $\mathbf{Q} = \{q_1, q_2, q_3\}$. q_1 corresponds to cruising before the avoid maneuver, q_2 corresponds to the avoid mode and q_3 corresponds to cruising after the avoid maneuver has been completed. There are two transitions. The first is controllable by a discrete input variable, σ, and corresponds to the initiation of the avoid maneuver. The second transition, corresponding to the completion of the avoid maneuver, is required to take place exactly when the aircraft have completed a half circle, and is therefore treated as uncontrollable. The continuous state space is augmented with a timer $z \in \mathbb{R}$ to force this transition. We set $X = \{x_r, y_r, \psi_r, z\}$. The dynamics of the maneuver can easily be encoded by a hybrid automaton, shown pictorially in Figure 7.2. The maneuver is safe if the aircraft remain at least 5 nautical miles apart throughout, that is:

$$F = \{q_1, q_2, q_3\} \times \{x \mid x_r^2 + y_r^2 \geq 25\}$$

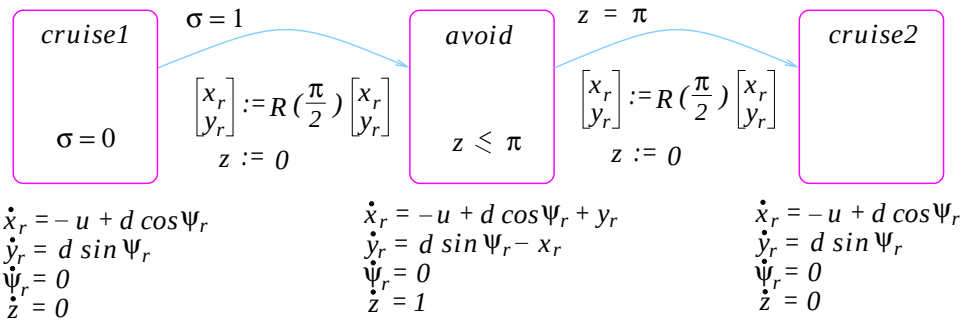


Figure 7.2: Hybrid dynamics of the two aircraft example

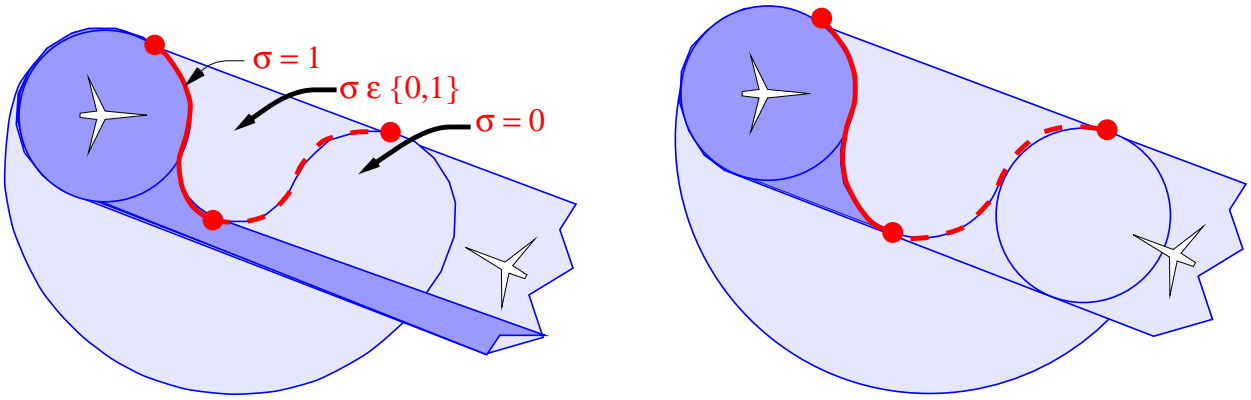


Figure 7.3: Enabling and forcing boundaries for σ_0 and the effect of increasing the radius of the turn.

W^* can be computed following the above algorithm. The computation R*each* is simplified by the fact that the continuous inputs are assumed to be constant, while the computation of Pre_u and Pre_d is simplified by the fact that there are no discrete disturbance inputs. The resulting controller for σ is illustrated in Figure 7.3(a). The transition $\sigma = 0$ until the continuous state reaches the dashed line; at this point the transition is enabled (σ is allowed to be either 0 or 1), and the transition to state q_2 may occur. The transition is forced to occur (by setting $\sigma = 1$) when the state reaches the solid boundary of W^*. Note that there are states which are not rendered safe by the maneuver. Indeed, in q_1, if the initial state is in the dark region, then the aircraft are doomed to collide. Figure 7.3(b) displays the result of increasing the radius of the turn in q_2. Notice that the set W^* increases as the turning radius increases. This implies that the maneuver renders a larger subset of the state space safe. Figure 7.3(b) shows the critical value of the turning radius, for which the maneuver is guaranteed to be safe, provided the conflict is detected early enough.

For more details on the application of hybrid systems to multi-aircraft conflict resolution see [78].

Chapter 9

Algorithms for Distributed Air Traffic Management

We first describe the Air Traffic Control (ATC) system used in the United States today, emphasizing the structure of the airspace, and the methods used by air traffic controllers to direct traffic, and by pilots to follow these directions. We then describe some of the technologies, both proposed and under development, to enable a change towards a more efficient system. [**?**], [**?**], [**?**], [47], and [**?**] provide excellent overviews of the current ATC and some of the new technologies available. We describe a proposed architecture for new Air Traffic Management (ATM) which would move much of the current ATC functionality on board each aircraft. We conclude with three examples representing two crucial problems to be solved in *any* proposed ATM: the problem of conflict resolution between aircraft, and that of consistent and safe flight mode switching in an aircraft's autopilot.

9.1 Overview of the Current System

ATC has its earliest roots in the 1920's, when local airline dispatchers would direct pilots to fly flight plans marked by rudimentary markers on the ground. In 1935, the first inter-airline ATC was organized in the Chicago-Cleveland-Newark corridor, which was taken over in 1937 when the responsibility for ATC was transferred from the airlines to the federal government. The advances in radar and radio technology in the ensuing decades allowed closer surveillance of aircraft, and the growth of the aircraft jet engine industry made it possible for the average aircraft to fly at much faster speeds. The system of aircraft, pilots, and controllers evolved into what today is known as the *National Airspace System*, or NAS, and its management is referred to as *Air Traffic Management*, or ATM.

ATM in the United States is currently organized hierarchically with a single *Air Traffic Control System Command Center (ATCSCC)* supervising the overall traffic flow. This is supported by 22 *Air Route Traffic Control Centers (ARTCCs)* organized by geographical region, which control the airspace up to 60,000 feet. Each Center is sub-divided into about 20 sectors, with at least one air traffic controller responsible for each sector. Coastal ARTCCs have jurisdiction over oceanic airspace: the Oakland Center in California, for example, controls a large part of the airspace above the Pacific Ocean. Within the Center airspace,

the low traffic density region away from airports is known as the *en route airspace* and is under jurisdiction of the ARTCC. The high traffic density regions around urban airports are delegated to *Terminal Radar Approach Control (TRACON)* facilities. The TRACONs generally control this airspace up to 15,000 feet. There are more than 150 TRACONS in the United States: one may serve several airports. For example, the Bay Area TRACON includes the San Francisco, Oakland, and San José airports along with smaller airfields at Moffett Field, San Carlos, and Fremont. The regions of airspace directly around an airport as well as the runway and ground operations at the airport are controlled by the familiar *Air Traffic Control Towers*. There are roughly 17,000 landing facilities in the United States serving nearly 220,000 aircraft. Of these there are about 6,000 commercial aircraft: the number of commercially used airstrips is roughly 400 (these are all equipped with control towers).

ATC currently directs air traffic along predefined jet ways, or "freeways in the sky", which are straight line segments connecting a system of beacons (non-directional beacons (NDBs), very high frequency omni-range receivers (VORs), and distance measuring equipment (DME)). These beacons are used by pilots (and autopilots) as navigational aids, to update and correct the current position information provided by the inertial navigation systems (INS) on board each aircraft. Surveillance is performed by ATC through the use of radar: a primary radar system which processes reflected signals from the aircraft skin, and a secondary radar system, which triggers a transmitter in the aircraft to automatically emit an identification signal. The range of the radars depends on the type of airspace being served: in the en route airspace the long-range Air Route Surveillance Radar (ARSR) is used, while in the TRACON the shorter range Automated Radar Terminal System (ARTS) is used. The accuracy of the radars, and their slow (12 second) update rates, contribute to the FAA standards for aircraft separation, which are 5 nautical miles horizontal separation, 1000 feet (2000 feet above 29,000 feet) vertical separation in the Center airspace, and 3 nautical miles horizontal separation, 1000 feet vertical separation in the TRACON. Each ATC facility is equipped with a computer system which takes the radar signals as input and provides a very limited amount of flight data processing, including a rudimentary conflict alert function. This information is displayed to controllers in two-dimensions on the black and green plan view displays (PVDs). Controllers issue directives to pilots using two-way voice (radio) channels. Figure 9.1 shows a flight map (horizontal profile) of a portion of the San Francisco Bay Area: the circular "dials" indicate VOR beacons (including airports), the boundary of the TRACON is shown as well as a part of the Oakland Center airspace.

Prior to a commercial aircraft's departure, the airline files a flight plan with ATC, which indicates information about the aircraft and its desired trajectory from origin to destination airports in the form of a very coarse sequence of *way points*. ATC modifies the flight plan according to constraints of the NAS and other aircraft, and issues a clearance to the pilot. After take-off, the control of the aircraft is passed through the Tower, TRACON, and possibly several Center facilities until the destination TRACON is reached. Information about the part of the filed flight plan relevant to his sector is passed via the computer system to each TRACON and Center controller, and the information is printed out on "flight strips" (Figure 9.2) which indicate the planned position of the aircraft at several points along the route.

The main goal of ATC is to maintain safe separation between aircraft while guiding them to their destinations. However, the tight control that it has over the motion of every aircraft

Figure 9.1: Bay Area airports, TRACON, and part of Oakland Center.

Figure 9.2: A flight strip from the Oakland Center.

in the system frequently causes bottlenecks to develop. Uncertainties in the positions, velocities, and wind speeds, as well as the inability of a single controller to handle large numbers of aircraft at once lead to overly conservative controller actions and procedures to maintain safety. An example of this is the methods used by air traffic controllers to predict and avoid conflicts between aircraft. If a controller predicts that the separation between two aircraft will become less than the regulatory separation, the controller will issue a directive to one or both of the pilots to alter their paths, speed, or both. Often the resolution is not needed, and usually it is too severe. Also, the so-called "user preferred routes" (shorter or lower fuel consumption routes that take advantage of tailwinds) are disallowed because of the requirement to use prescribed jet ways.

Airspace capacity is the maximum number of operations that can be processed per unit time in a certain volume of the airspace given a continuous demand [47]. In this definition a distinction is made between different modes of operation, such as *level flight at fixed heading*, *climbing*, *descending*, and *changes in heading*. Airspace capacity is a function of aircraft count, activity mix, protocols for conflict detection and resolution, and FAA regulations. It is our contention that this capacity can be increased by better protocols which do not compromise safety.

An area of current activity is the development of decision support tools for air traffic controllers. One such tool is the *Center-TRACON Automation System (CTAS)* [29] which is currently under development at NASA Ames, and under field test at Denver and Dallas-Fort Worth airports. CTAS is software code which runs on computer workstations next to the air traffic controller; it uses radar data, current weather information, aircraft flight plans and simplified dynamic aircraft models to predict the aircraft trajectories, alert the controllers about potential conflicts, and provide advisories to the controller about landing sequences.

We conclude this section with a short introduction to the automated flight management system (FMS) on board commercial jets, such as those of the Boeing B777 and the Airbus A320. In contrast to the "low technology" ATC operation, modern FMSs are highly automated systems which assist the pilot in constructing and flying four-dimensional trajectories, as well as altering these trajectories on line in response to ATC directives. An FMS typically controls the throttle input and the vertical and lateral trajectories of the aircraft to automatically perform such functions as: acquiring a specified altitude and then leveling (ALT ACQ), holding a specified altitude (ALT HLD), acquiring a specified vertical climb or descend rate (V/S), automatic vertical or lateral navigation between specified way points (VNAV, LNAV), or holding a specified throttle value (THR HLD). The combination of these throttle-vertical-lateral modes is referred to as the *flight mode* of the aircraft. A typical autopilot has several hundred flight modes (see [**?**] for a discussion of the Boeing B737 flight modes). It is interesting to note that these flight modes were designed to automate the way pilots fly aircraft manually: by controlling the lateral and vertical states of the aircraft to set points for fixed periods of time, pilots simplify the complex task of flying an aircraft. Figure 9.3 illustrates two screens in the cockpit of such an FMS-equipped jet: a horizontal profile showing the current position of the aircraft as it follows an approach route, marked by way points, into the Los Angeles airport, and an "artificial horizon" which shows the current pitch and roll angles of the aircraft, the airspeed and altitude, and the current flight mode. Prior to take-off, the pilot can enter the approved flight plan into the FMS computer on board the aircraft, and during flight can choose the desired level of automation. For

Figure 9.3: Two screens in a typical glass cockpit: (a) a horizontal profile of way points (into Los Angeles airport); (b) an "artificial horizon" showing the current pitch and roll angles of the aircraft, the airspeed and altitude, and the current flight mode. The first three columns in the flight mode are the throttle-vertical-lateral modes, the fourth is the autopilot mode. ARM means "waiting for the throttle to reach required value", MCP SPD means "speed is controlled to the entry in the mode control panel", HDG SEL means "heading is controlled to the entry in the mode control panel", CMD means "pilot has command over pitch and roll values".

example, if the pilot selects the LNAV or VNAV mode, the FMS determines the altitudes, speeds, pitch, roll, and throttle values to navigate between way points; if the HDG SEL or ALT ACQ modes are chosen, the pilot chooses the desired heading and altitude values.

While the introduction of automation to on-board avionics has resulted in increased performance of commercial autopilots, the need for automation designs which guarantee safe operation of the aircraft has become paramount. Currently, designers and manufacturers of FMSs "verify" the safety of the systems by simulating them for long periods of time with various initial conditions and inputs. This procedure is not adequate, since trajectories to unsafe states may be overlooked. "Automation surprises" have been extensively studied [**?**, **?**, 52] *after* the unsafe situation occurs, and "band-aids" are added to the FMS design to ensure the same problem does not occur again. One of the goals of this dissertation is to present a system design method, in which safety properties are *a priori* verified in the design phase, so that no automation surprises occur.

9.2 Technologies to Enable Change

Several new technologies are under development and certification, and are fueling a change in the structure of ATM. In this section we discuss the *Global Positioning System (GPS)* and a datalink communication protocol called *Automatic Dependent Surveillance (ADS)* and their impact on the future of ATM.

GPS provides 3D position information worldwide using signal information from a constellation of 24 satellites. A single GPS receiver can determine its position to an accuracy of a few meters, using signals from at least 4 out of these 24 satellites; if this information is augmented with differential corrections from another receiver (differential GPS or DGPS), this accuracy can be increased to a few centimeters. Many factors make the use of GPS in the cockpit a desirable alternative to the current ATM navigation methods [**?**]: the accuracy is uniform from aircraft to aircraft whereas with the currently used INS, the accuracy decreases in time due to sensor drift rates; each GPS receiver acts like an atomic-accurate clock, thus making it possible for many aircraft to coordinate among each other over a communication link; a GPS receiver is much cheaper than an INS system, and orders of magnitude cheaper than a VOR beacon. One disadvantage of relying on GPS position information is that the satellite signal may be lost temporarily if the GPS receiver is obscured from the direct path of the signal. Current studies [**?**] suggest an integrated use of both INS and GPS, in which the accurate position information from GPS is used to continually correct the INS position.

ADS is a communication protocol by which aircraft would transmit over digital satellite communication their GPS position information, velocity, as well as information about their intended trajectory, to the ground ATC. ADS-B (for broadcast) is a protocol for broadcasting this information to neighboring aircraft [**?**]. Its major advantage over the current ATM surveillance methods is its ability to provide very accurate information for trajectory prediction, without relying on the radar system. Two immediate benefits of such a communication link are a huge improvement in surveillance over oceanic airspace, which is not covered by radar, and the possibility of reducing the separation standards between aircraft in all airspace.

Despite the short-term benefits that these new technologies provide, the real long-term benefits will depend on how the airspace system and its management evolve around such new technologies. Aviation in the next century will, more than ever before, be based on *systems related issues*: the need to integrate highly automated aircraft, advanced navigation and surveillance technology, sophisticated computation, and user preferences, into a system which meets the demands resulting from skyrocketing growth in air travel, without compromising the standards of such a safety critical system. The aviation community has accepted that today's controller-based system will not meet these requirements, and a new system structure is needed. A concept called *free flight* [**?**] has been proposed in recent years. Free flight is loosely defined to mean that pilots are allowed to choose their own routes, altitude and speed, and would share the tasks of navigation, surveillance, aircraft separation, and weather prediction, with ground-based controllers. User preference would be restricted only in congested or special use (military) airspace.

In the following section, we present an architecture for a "next generation" air traffic management system [**?**], which incorporates user preference and moves some of the current ATC functionality on board the aircraft. Our purpose in presenting this architecture is to provide a framework for the examples presented in this dissertation: the modeling, verification, and controller synthesis techniques which are at the heart of this dissertation are general, and may be applied to any ATM architecture.

Figure 9.4: Proposed framework for on-board planning and control.

9.3 Proposed Architecture

We assume, as in the current ATC practice, that user (airline) preferences are incorporated in the initial flight planning stage, in which the airline and ATC can "negotiate" the sequence of way points that comprises the nominal flight plan for the aircraft. This nominal plan is designed to be time-optimal and conflict-free, within the constraints of the schedules of the other aircraft in the system. Once a commercial aircraft is airborne and outside of the TRACON, it starts to play an active role in its own navigation and surveillance. As shown in Figure 9.4, the flight management system on board each aircraft may be interpreted as a hierarchical system, which takes as input the nominal flight plan from ATC, information about neighboring aircraft, about its own aircraft state, and about wind and weather, and produces a conflict-free full state and input trajectory [**?**]. The *strategic planner* interpolates the nominal trajectory's way points with a set of *control points* which delineate the constant control segments between way points. The *tactical planner* refines the strategic plan by joining the control points with a smooth output trajectory. The *trajectory planner* uses a detailed dynamic model of the aircraft, sensory input about the wind's magnitude and direction, and the tactical plan, to design a full state and input trajectory for the aircraft, and the sequence of *flight modes* necessary to execute the dynamic plan. The *regulation layer* is a simple, fast control scheme, which closes the loop on the dynamics of the aircraft. Tracking errors are passed back to the trajectory planner, to facilitate replanning if necessary.

Often, as with the current ATM, bad weather, high winds, or schedule delays which cause conflicts with other aircraft may force the aircraft to deviate from the nominal route. The strategic planner on board the aircraft has the ability to coordinate with neighboring aircraft to determine a sequence of maneuvers which will result in conflict-free trajectories. We propose a conflict resolution methodology based on a set of *protocols*, easily understood by pilots and easily programmed into an FMS, to allow aircraft to coordinate among each other to avoid conflict. Each strategic planner then commands its own tactical planner to

follow these maneuvers.

9.4 Motivating Examples

We now concentrate on two systems in an ATM architecture: a *provably-safe* algorithm for resolving trajectory conflicts between aircraft, and a *provably-safe* algorithm for a single aircraft to switch between different flight modes. The notion of "safety" in each case is crucial:

Definition 9.1 (Safety) *A system is* safe *if its state trajectories always remain within a* safe subset *of the state space.*

In the conflict resolution problem, the system is safe if the aircraft always maintain *minimum separation* with each other. In the flight mode switching problem, system safety means that the state of the aircraft remains within minimum and maximum bounds imposed on its velocities, angles etc. so that the aircraft doesn't stall, causing it to plunge out of the sky. The latter is referred to as *aerodynamic envelope protection*. We present these systems through examples, which are introduced in this section and developed throughout the dissertation.

9.4.1 Conflict Resolution for Aircraft

Consider a system of aircraft, each navigating using a combination of GPS and INS, and each providing surveillance information through an ADS link with ATC, and an ADS-B link with neighboring aircraft. Each aircraft is surrounded by two virtual *cylinders*, the *protected zone* and *alert zone* shown in Figure 9.5 as a top view. The radius and height of the protected zone depends on the FAA separation standards (2.5 nautical miles by 1000 feet in Center, 1.5 nautical miles by 1000 feet in TRACON). The size and shape of the alert zone depends on various factors including airspeed, altitude, accuracy of sensing equipment, traffic situation, aircraft performance and average human and system response times: it is shown as an ellipsoid in Figure 9.5. A *conflict* or loss of separation between aircraft occurs when their protected zones overlap. The system of aircraft is defined to be *safe* if the aircraft trajectories are such that their protected zones never overlap.

We propose a conflict resolution algorithm which may be executed either on board each aircraft, as suggested by the architecture of the previous section, or in an ATC TRACON or ARTCC facility on the ground. The algorithm has access to the state and intent information of the other aircraft involved in the conflict, through the GPS/INS system linked to the ADS/ADS-B communication link, to information about the aerodynamics and performance characteristics of the other aircraft, and to information about the constraints imposed by the global traffic flow (see Figure 9.6). When aircraft enter the alert zone of another aircraft, an alert is issued to ATC as well as to the FMS of each involved aircraft, and depending on the relative configurations (positions, velocities) of the aircraft, a maneuver is generated which resolves the conflict. ¿From a database of flight modes, such as segments of constant heading, of constant bank angle, of constant airspeed, the conflict resolution algorithm synthesizes the *parameters of the maneuver*, such as the proper sequencing of these modes, the numerical values associated to each segment (heading angle, bank angle, airspeed), and

Figure 9.5: Aircraft Zones.

the conditions for switching between flight modes. The result is a maneuver, proven to be safe within the limits of the models used, which is a familiar sequence of commands easily executable by the FMSs. The resulting maneuvers may be viewed as *protocols*, or "rules of the road".

Conflict prediction and resolution have been sources of interest for the air traffic, control, and computational geometry communities in recent years. Spatial and temporal approaches, such as [**?**, **?**], calculate the four dimensional coordinates of a possible conflict. Probabilistic approaches, such as [84, 68] assume stochastic uncertainty in the measured information and determine the probability of collision. The user interface of CTAS allows controllers to manually alter aircraft trajectories to resolve conflicts in en route airspace. TCAS [**?**] provides resolution advisories (flight level changes) to pilots involved in two-aircraft conflicts, however these advisories are not formally verified. Conflict prediction, resolution, and verification are the most important modules that are in need of augmentation in the current implementations of CTAS and TCAS. A feature of our algorithm is that it is *provably safe* to within the limits of our models. We account for uncertainty or incompleteness in any of the information: as the bounds on the uncertainties increase, so does the conservatism of the resulting maneuver.

Conflict Resolution for Two Aircraft in $SE(2)$

We present as motivating example a model for the kinematic motions of two aircraft at a fixed altitude, as shown in Figure 9.7(a). The position and heading of each aircraft is described by an element of the Lie group G of rigid motions in $\mathbb{R}^2$, called $SE(2)$ for the *Special Euclidean* group in $\mathbb{R}^2$. Let $g_i \in G$ denote the configuration of aircraft i:

$$g_i = \begin{bmatrix} \cos\psi_i & -\sin\psi_i & x_i \\ \sin\psi_i & \cos\psi_i & y_i \\ 0 & 0 & 1 \end{bmatrix} \tag{9.1}$$

Figure 9.6: Conflict Resolution Algorithm.

Figure 9.7: (a) Two aircraft in a conflict scenario; (b) The relative configuration, showing the relative protected zone.

where (x_i, y_i) denotes the position of aircraft i and ψ_i is its heading. The motion of the aircraft may be modeled as a left-invariant vector field on G:

$$\dot{g}_i = g_i X_i \tag{9.2}$$

where $X_i \in \mathcal{G}$, the Lie algebra associated with the Lie group G. The Lie algebra in this case is $\mathcal{G} = se(2)$, with $X_i \in se(2)$ represented as

$$X_i = \begin{bmatrix} 0 & -\omega_i & v_i \\ \omega_i & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{9.3}$$

where ω_i is the aircraft's angular velocity, and v_i is its airspeed.

A coordinate change is performed to place the identity element of the Lie group G on aircraft 1, as shown in Figure 9.7(b). Let $g_r \in G$ denote the relative configuration of aircraft 2 with respect to aircraft 1. Then

$$g_2 = g_1 g_r \Rightarrow g_r = g_1^{-1} g_2 \tag{9.4}$$

In local coordinates, the coordinate transformation is expressed as

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = R(-\psi_1) \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix} = \begin{bmatrix} \cos(-\psi_1) & -\sin(-\psi_1) \\ \sin(-\psi_1) & \cos(-\psi_1) \end{bmatrix} \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix} \tag{9.5}$$

$$\psi_r = \psi_2 - \psi_1 \tag{9.6}$$

and g_r is given by

$$g_r = \begin{bmatrix} \cos\psi_r & -\sin\psi_r & x_r \\ \sin\psi_r & \cos\psi_r & y_r \\ 0 & 0 & 1 \end{bmatrix} \tag{9.7}$$

in which $(x_r, y_r, \psi_r) \in \mathbb{R}^2 \times [-\pi, \pi)$ represent the relative position and orientation of aircraft 2 with respect to aircraft 1. Differentiating g_r, we obtain

$$\dot{g}_r = g_r X_2 - X_1 g_r \tag{9.8}$$

which may be written in (x_r, y_r, ψ_r) coordinates as

$$\begin{aligned} \dot{x}_r &= -v_1 + v_2 \cos\psi_r + \omega_1 y_r \\ \dot{y}_r &= v_2 \sin\psi_r - \omega_1 x_r \\ \dot{\psi}_r &= \omega_2 - \omega_1 \end{aligned} \tag{9.9}$$

The protected zone of aircraft 2 may be translated to the origin as shown in Figure 9.7(b).

In order to maintain safe separation, the relative position (x_r, y_r) must remain outside of the protected zone, defined as

$$\{(x_r, y_r, \psi_r) : x_r^2 + y_r^2 < 5^2\} \tag{9.10}$$

for the lateral 5 nautical mile separation in Center airspace.

The flight modes for this system of two aircraft are based on the linear and angular velocities of the aircraft. We consider two possibilities: $\omega_i = 0$, meaning that aircraft i follows a straight line, and $\omega_i \neq 0$, but is a constant, meaning that aircraft i follows an arc of a circle. Thus the *database of maneuvers* for the example in this section are straight line segments of varying length and associated varying airspeed, and arcs of circles of varying length and radii. These maneuvers approximate closely the behavior of pilots flying aircraft: straight line segments (constant heading) and arcs of circles (constant bank angle) are easy to fly both manually and on autopilot.

Three-Mode Example

Consider a scenario in which there are three modes of operation: a *cruise* mode in which both aircraft follow a straight path; an *avoid* mode in which both aircraft follow a circular arc path; and a second *cruise* mode in which the aircraft return to the straight path. The protocol of the maneuver is that as soon as the aircraft are within a certain distance of each other, each aircraft turns 90° to its right and follows a half circle. Once the half circle is complete, each aircraft returns to its original heading and continues on its straight path (Figure 9.8). In each mode, the continuous dynamics may be expressed in terms of the *relative motion* of the two aircraft (9.9). In the cruise mode, $\omega_i = 0$ for $i = 1, 2$ and in the avoid mode, $\omega_i = 1$ for $i = 1, 2$. We assume that both aircraft switch modes simultaneously, so that the relative orientation ψ_r is constant. This assumption simply allows us to display the state space in two dimensions, making the results easier to present.

Problem statement: Generate the relative distance between aircraft at which the aircraft may switch safely from mode 1 to mode 2, and the minimum turning radius R in mode 2, to ensure that the 5 nautical mile separation is maintained.

Seven-Mode Example

The previous example is somewhat academic (aircraft cannot change heading instantaneously), yet (as so often happens with academic examples) its simplicity makes it a good vehicle to illustrate the controller synthesis methods of this dissertation. To show that our methods are not confined to academia and may indeed be applied to real-world situations, we present a "seven-mode example" which much better approximates current ATC practice. The example is illustrated in Figure 9.9. When two aircraft come within a certain distance of each other, each aircraft starts to turn to its right, following a trajectory which is a sequence of arcs of circles of fixed radii, and straight lines. As in the previous example, we assume that both aircraft switch modes simultaneously. We also assume that the angles of the avoid maneuver are fixed, so that the straight path of mode 3 is at a $-45°$ angle to the straight path of mode 1, and that of mode 5 is at a 45° to that of mode 1. Also, the length of each arc is fixed at a prespecified value, and the lengths of the segments in modes 3 and 5 are equal to each other, but unspecified.

Problem statement: Given some uncertainty in the actions of the aircraft, generate the relative distance between aircraft at which the aircraft may switch safely from mode 1 to mode 2, and the minimum lengths of the segments in modes 3 and 5, to ensure that the 5 nautical mile separation is maintained.

9.4.2 Flight Mode Switching and Envelope Protection

We would like to design a *safe* automatic flight mode switching algorithm for an FMS which interacts with both the dynamical system consisting of the aircraft and autopilot as well as with Air Traffic Control (ATC), and guides the aircraft safely through a sequence of waypoints in the presence of disturbances. As overviewed in the previous section, the FMS accepts a high level trajectory plan from ATC and constructs a sequence of elementary

Figure 9.8: Two aircraft in three modes of operation: in modes 1 and 3 the aircraft follow a straight course and in mode 2 the aircraft follow a half circle. The initial relative heading (120°) is preserved throughout.

Figure 9.9: Two aircraft in seven modes of operation: in modes 1, 3, 5, and 7 the aircraft follow a straight course and in modes 2, 4, and 6 the aircraft follow arcs of circles. Again, the initial relative heading (120°) is preserved throughout.

Figure 9.10: A planar aircraft in flight with attached axes about its center of mass.

flight modes to effect a conflict-free version of this plan. The trajectory planner in the FMS is responsible for the safe sequencing of these modes. In this case, the aircraft is safe if its state trajectory remains within the *aerodynamic flight envelope*, which is a subset of the state space delineated by allowable limits on the airspeed, vertical velocity, flight path angle, and altitude. Our algorithm must ensure that the FMS will attempt to select and activate only those flight modes for which the state trajectory is guaranteed to stay within the envelope. This is known as *envelope protection*.

Mode Switching for the Longitudinal Axis Dynamics of a CTOL Aircraft

The example is inspired by the work of [40], in which the flight modes for the airspeed and flight path angle dynamics of an aircraft are derived.

We consider a nonlinear model of the longitudinal axis dynamics of a conventional take-off and landing (CTOL) aircraft in normal aerodynamic flight in still air [75, **?**], shown in Figure 9.10. The horizontal and vertical axes are respectively the $(x_{inertial}, h_{inertial})$ (denoted x, h) axes and the *pitch angle* θ is the angle made by the aircraft body axis, x_{body} with the x axis. The *flight path angle* γ and the *angle of attack* α are defined as: $\gamma = \tan^{-1}(\frac{\dot{h}}{\dot{x}})$, $\alpha = \theta - \gamma$. Expressions for the lift (L) and drag (D) forces are given by

$$\begin{array}{rl} L & = a_L(\dot{x}^2 + \dot{h}^2)(1 + c\alpha) \\ D & = a_D(\dot{x}^2 + \dot{h}^2)(1 + b(1 + c\alpha)^2) \end{array} \tag{9.11}$$

where a_L, a_D are dimensionless *lift* and *drag coefficients*, and b and c are positive constants. We assume that the autopilot has direct control over both the forward thrust T (throttle) and the aircraft pitch θ (through the elevators), thus there are two continuous control inputs $(u_1, u_2) = (T, \theta)$. Physical considerations impose constraints on the inputs:

$$u \in [T_{min}, T_{max}] \times [\theta_{min}, \theta_{max}] \tag{9.12}$$

The longitudinal dynamics may be modeled by the Newton-Euler equations:

$$M \begin{bmatrix} \ddot{x} \\ \ddot{h} \end{bmatrix} = R(\theta) \left[R^T(\alpha) \begin{bmatrix} -D \\ L \end{bmatrix} + \begin{bmatrix} T \\ 0 \end{bmatrix} \right] + \begin{bmatrix} 0 \\ -Mg \end{bmatrix} \tag{9.13}$$

$$J\ddot{\theta} = u_2 \tag{9.14}$$

where $R(\alpha)$ and $R(\theta)$ are standard rotation matrices, M is the mass of the aircraft, and g is gravitational acceleration. The state of the system is $\mathbf{x} = (x, \dot{x}, h, \dot{h})^T$.

The airspeed of the aircraft is defined as $V = \sqrt{\dot{x}^2 + \dot{h}^2}$. The simplified FMS studied in this dissertation uses control inputs T and θ to control combinations of the airspeed V, flight path angle γ, and altitude h. The linear and angular accelerations $(\dot{V}, V\dot{\gamma})$ may be derived directly from (9.13):

$$\dot{V} = -\frac{D}{M} - g\sin\gamma + \frac{T}{M}\cos\alpha \tag{9.15}$$

$$V\dot{\gamma} = \frac{L}{M} - g\cos\gamma + \frac{T}{M}\sin\alpha \tag{9.16}$$

Note that these dynamics are expressed solely in terms of (V, γ) and inputs (T, θ), where $\alpha = \theta - \gamma$; thus equations (9.15), (9.16) are a convenient way to represent the dynamics for modes in which h is not a controlled variable.

Safety regulations for the aircraft dictate that V, γ, and h must remain within specified limits:

$$\begin{array}{c} V_{min} \le V \le V_{max} \\ \gamma_{min} \le \gamma \le \gamma_{max} \\ h_{min} \le h \le h_{max} \end{array} \tag{9.17}$$

where $V_{min}, V_{max}, \gamma_{min}, \gamma_{max}, h_{min}, h_{max}$ are functions of such factors as airspace regulations, type of aircraft, and weather. For aircraft flying in en-route airspace, we assume that these limits are constants, and thus the aerodynamic flight envelope F is as illustrated in Figure 9.11, in (V, γ)-space and $(h, V, \dot{h})$-space, where $\dot{h} = V\sin\gamma$. The state trajectory must remain within F at all times during en-route flight. We also impose a *secondary criterion*, that the state trajectory must satisfy constraints on the linear and angular acceleration:

$$|\dot{V}| \le 0.1g, \quad |V\dot{\gamma}| \le 0.1g \tag{9.18}$$

imposed for passenger comfort.

The system may be discretized into five flight modes, depending on the state variables being controlled:

- **Mode 1:** (Speed, Flight Path), in which the thrust T is between its specified operating limits ($T_{min} < T < T_{max}$), the control inputs are T and θ, and the controlled outputs are the speed and the flight path angle of the aircraft $y = (V, \gamma)^T$;
- **Mode 2:** (Speed), in which the thrust saturates ($T = T_{min} \vee T = T_{max}$) and thus it is no longer available as a control input; the only input is θ, and the only controlled output is V;
- **Mode 3:** (Flight Path), in which the thrust saturates ($T = T_{min} \vee T = T_{max}$); the input is again θ, and the controlled output is γ;

Figure 9.11: (a) Simplified Aerodynamic Flight Envelope in (V, γ)-space: axes are airspeed V, flight path angle γ; (b) Simplified Aerodynamic Flight Envelope in $(h, V, \dot{h})$-space: axes are altitude h, airspeed V, vertical speed $\dot{h}$.

- **Mode 4:** (Speed, Altitude), in which the thrust T is between its specified operating limits ($T_{min} < T < T_{max}$), the control inputs are T and θ, and the controlled outputs are the speed and the vertical position of the aircraft $y = (V, h)^T$;

- **Mode 5:** (Altitude), in which the thrust saturates $T = T_{min} \vee T = T_{max}$; the input is θ, and the controlled output is h.

In our calculations we use the following parameter values, which correspond to a DC-8 at cruising speed: $M = 85000$kg, $b = 0.01$, $c = 6$, $a_L = 30$, $a_D = 2$, $T_{min} = 40000$ N, $T_{max} = 80000$ N, $\theta_{min} = -22.5°$, $\theta_{max} = 22.5°$, $V_{min} = 180$ m/s, $V_{max} = 240$ m/s, $\gamma_{min} = -22.5°$ and $\gamma_{max} = 22.5°$. The bounds on the pitch angle θ and the flight path angle γ are chosen to be symmetric about zero for ease of computation. In actual flight systems, the positive bound on these angles is greater than the negative bound. Also, the angles chosen for this example are greater than what are considered acceptable for passenger flight ($\pm 10°$). Since we are interested in en route flight, the limits on the altitudes are: $h_{min} = 15,000$ feet, $h_{max} = 51,000$ feet.

Problem statement: Generate the mode switching logic for this flight management system, as well as the continuous control inputs (T, θ) to use in each flight mode, so that envelope protection is guaranteed.

Bibliography

[1] J.-R. Abrial. The steam-boiler control specification problem. In J.-R. Abrial, E. Börger, and H. Langmaack, editors, *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, number 1165 in LNCS. Springer-Verlag, Berlin, 1996.

[2] J.-R. Abrial, E. Börger, and H. Langmaack. The steam-boiler case study project, an introduction. In J.-R. Abrial, E. Börger, and H. Langmaack, editors, *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, number 1165 in LNCS. Springer-Verlag, Berlin, 1996.

[3] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho. Hybrid automaton: An algorithmic approach to the specification and verification of hybrid systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, number 736 in LNCS, pages 209–229. Springer-Verlag, Berlin, 1993.

[4] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[5] R. Alur and D. L. Dill. Automata for modeling real-time systems. In *Proceedings of ICALP '90*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer-Verlag, Berlin, 1990.

[6] R Alur and T A Henzinger. Modularity for timed and hybrid systems. In A. Mazurkiewicz and J. Winkowski, editors, *CONCUR 97: Concurrency Theory*, Lecture Notes in Computer Science 1243, pages 74–88. Springer-Verlag, 1997.

[7] R. Alur and T.A. Henzinger. Reactive modules. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 207–218. IEEE Computer Society Press, 1996.

[8] R. Alur and T.A. Henzinger. Modularity for timed and hybrid systems. In *Proceedings of the Eighth International Conference on Concurrency Theory (CONCUR 1997)*, number 1243 in LNCS, pages 74–88, Berlin, 1997. Springer-Verlag.

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

[10] M. Anderson, D. Bruck, S. E. Mattsson, and T. Schonthal. Omsim- an integrated interactive environment for object-oriented modeling and simulation. In *IEEE/IFAC joint symposium on computer aided control system design*, pages 285–290, 1994.

[11] M. Andersson. *Object-Oriented Modeling and Simulation of Hybrid Systems*. PhD thesis, Lund Institute of Technology, Lund, Sweden, December 1994.

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

[13] T. Başar and G. J. Olsder. *Dynamic Non-cooperative Game Theory*. Academic Press, second edition, 1995.

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

[15] B. Bérard, P. Gastin, and A. Petit. On the power of non observable actions in timed automata. In *Actes du STACS '96*, Lecture Notes in Computer Science 1046, pages 257–268. Springer-Verlag, Berlin, 1996.

[16] V.S. Borkar. *Probability theory: an advanced course*. Springer-Verlag, New York, 1995.

[17] M.S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD thesis, Massachusetts Institute of Technology, 1995.

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

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

[20] R.W. Brockett. Hybrid models for motion control systems. In H.L. Trentelman and J.C. Willems, editors, *Perspectives in Control*, Boston, MA, 1993. Birkhäuser.

[21] M. Broucke. Regularity of solutions and homotopy equivalence for hybrid systems. In *IEEE Conference on Decision and Control*, Tampa, FL, 1998.

[22] A.E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Hemisphere Publishing Corporation, 1975.

[23] Frank M. Callier and Charles A. Desoer. *Linear System Theory*. Springer-Verlag, 1991.

[24] T. Dang and O. Maler. Reachability analysis via face lifting. In S. Sastry and T.A. Henzinger, editors, *Hybrid Systems: Computation and Control*, number 1386 in LNCS, pages 96–109. Springer-Verlag, Berlin, 1998.

[25] C. Daws, A. Olivero, S. Trypakis, and S. Yovine. The tool KRONOS. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III*, number 1066 in LNCS, pages 208–219. Springer-Verlag, Berlin, 1996.

[26] A. Deshpande, A. Gollu, and L. Semenzato. The SHIFT programming language and run-time system for dynamic networks of hybrid automata. Technical Report UCB-ITS-PRR-97-7, Institute of Transportation Studies, University of California, Berkeley, 1997.

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

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

[29] H. Erzberger, T.J. Davis, and S. Green. Design of center-tracon automation system. In *Proceedings of the AGARD Guidance and Control Syposium on Machine Intelligence in Air Traffic Management*, pages 11.1–11.12, 1993.

[30] A. F. Filippov. *Differential equations with discontinuous right-hand sides*. Kluwer Academic Publishers, 1988.

[31] Rainer Gawlick, Roberto Segala, Jorgen Sogaard-Andersen, and Nancy Lynch. Liveness in timed and untimed systems. In *Automata, Languages and Programming*, number 820 in LNCS, pages 166–177. Springer Verlag, 1994.

[32] M.R. Greenstreet and I. Mitchell. Integrating projections. In S. Sastry and T.A. Henzinger, editors, *Hybrid Systems: Computation and Control*, number 1386 in LNCS, pages 159–174. Springer-Verlag, Berlin, 1998.

[33] M. Heemels. *Linear Complementarity Systems: a Study in Hybrid Dynamics*. PhD thesis, Technische Universiteit Eindhoven, 1999.

[34] M. Heemels, H. Schumacher, and S. Weiland. Well-posedness of linear complementarity systems. In *Proc. 38th IEEE Conference on Decision and Control*, Phoenix, AZ, 1999.

[35] W. P. M. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001.

[36] T. A. Henzinger, P. H. Ho, and H. Wong Toi. A user guide to HYTECH. In E. Brinksma, W. Cleaveland, K. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, number 1019 in LNCS, pages 41–71, Berlin, 1995. Springer-Verlag.

[37] Tomas A. Henzinger and Howard Wong-Toi. Using HYTECH to synthesize control parameters for a steam boiler. In J.-R. Abrial, E. Börger, and H. Langmaack, editors, *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, number 1165 in LNCS, pages 265–282. Springer-Verlag, Berlin, 1996.

[38] M. Heymann, F. Lin, and G. Meyer. Control synthesis for a class of hybrid systems subject to configuration-based safety constraints. In *Hybrid and Real Time Systems*, number 1201 in LNCS, pages 376–391. Springer-Verlag, Berlin, 1997.

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

[40] Charles S. Hynes and Lance Sherry. Synthesis from design requirements of a hybrid system for transport aircraft longitudinal control. preprint, NASA Ames Research Center, 1996.

[41] J. Imura and A. J. van der Schaft. Characterization of well-posedness of piecewise linear systems. *IEEE Transactions on Automatic Control*, 45(9):1600–1619, September 2000.

[42] K.H. Johansson, M. Egerstedt, J. Lygeros, and S.S. Sastry. On the regularization of Zeno hybrid automata. *Systems and Control Letters*, 38(3):141–150, 1999.

[43] K.H. Johansson, J. Lygeros, S.S. Sastry, and M. Egerstedt. Simulation of Zeno hybrid automata. In *IEEE Conference on Decision and Control*, pages 3538–3543, Phoenix, Arizona, December 7–10, 1999.

[44] M. Johansson. *Piecewise linear control systems*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Sweden, March 1999.

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

[46] H.B. Khalil. *Nonlinear Systems*. Prentice Hall, third edition, 2001.

[47] P.K. Kokotovic and R.A. Freeman. *Robust nonlinear control design*. Birkhauser, Basel, 1996.

[48] M. Lemmon. On the existence of solutions to controlled hybrid automata. In Nancy Lynch and Bruce H. Krogh, editors, *Hybrid Systems: Computation and Control*, number 1790 in LNCS, pages 229–242. Springer-Verlag, Berlin, 2000.

[49] M. Lemmon, J. A. Stiver, and P. J. Antsaklis. Event identification and intelligent hybrid control. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, number 736 in LNCS, pages 268–296. Springer-Verlag, Berlin, 1993.

[50] B. Lennartsson, M. Tittus, B. Egardt, and S. Pettersson. Hybrid systems in process control. *Control Systems Magazine*, 16(5):45–56, 1996.

[51] J. Lewin. *Differential Games*. Springer-Verlag, 1994.

[52] H.R. Lewis and C. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, second edition, 1997.

[53] J. Lygeros. *Hierarchical Hybrid Control of Large Scale Systems*. PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1996.

[54] J. Lygeros, D.N. Godbole, and S.S. Sastry. Verified hybrid controllers for automated vehicles. *IEEE Transactions on Automatic Control*, 43(4):522–539, April 1998.

[55] J. Lygeros, K.H. Johansson, S.S. Sastry, and M. Egerstedt. On the existence of executions of hybrid automata. In *IEEE Conference on Decision and Control*, pages 2249–2254, Phoenix, Arizona, December 7–10, 1999.

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

[57] J. Lygeros, C.J. Tomlin, and S.S. Sastry. Multi-objective hybrid controller synthesis. Technical Report UCB/ERL M96/59, Electronic Research Laboratory, University of California Berkeley, 1997.

[58] J. Lygeros, C.J. Tomlin, and S.S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, March 1999.

[59] N. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg. Hybrid I/O automata. In *Hybrid Systems III*, number 1066 in LNCS, pages 496–510. Springer-Verlag, Berlin, 1996.

[60] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Theoretical Aspects of Computer Science*, number 900 in LNCS, pages 229–242, Berlin, 1995. Springer-Verlag.

[61] Zohar Manna and the STeP group. STeP: The Stanford Temporal Prover. Technical Report STAN-CS-TR-94-1518, Computer Science Department, Stanford University, July 1994.

[62] S. E. Mattsson. On object-oriented modeling of relays and sliding mode behaviour. In *Proc. 13th IFAC World Congress*, volume F, pages 259–264, San Francisco, CA, 1996.

[63] S. E. Mattsson, M. Andersson, and K. J. Åström. Object-oriented modelling and simulation. In D. A. Linkens, editor, *CAD for Control Systems*, chapter 2, pages 31–69. Marcel Dekker Inc., New York, 1993.

[64] S. E. Mattsson, M. Otter, and H. Elmqvist. Modelica hybrid modeling and efficient simulation. In *IEEE Conference on Decision and Control*, Phoenix, AZ, 1999.

[65] R. May. *Stability and Complexity of Model Ecosystems*. Princeton University Press, Princeton, NJ, 1973.

[66] A.S. Morse. Control using logic based switching. In A. Isidori, editor, *Trends in Control*, pages 69–114. Springer Verlag, 1995.

[67] A. Nerode and W. Kohn. Models for hybrid systems: Automata, topologies, controllability, observability. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, number 736 in LNCS, pages 317–356. Springer-Verlag, Berlin, 1993.

[68] R.A. Paielli and H. Erzberger. Conflict probability estimation for free flight. *Journal of Guidance, Control and Dynamics*, 20(3):588–596, 1997.

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

[70] A. Puri. *Theory of Hybrid Systems and Discrete Event Systems*. PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1995.

[71] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, Vol.77(1):81–98, 1989.

[72] S.S. Sastry. *Nonlinear Systems: Analysis, Stability and Control*. Springer-Verlag, New York, 1999.

[73] S. Simic, K.H. Johansson, S.S. Sastry, and J. Lygeros. Towards a geometric theory of hybrid systems. In Nancy Lynch and Bruce H. Krogh, editors, *Hybrid Systems: Computation and Control*, number 1790 in LNCS, pages 421–436. Springer-Verlag, Berlin, 2000.

[74] L. Tavernini. Differential automata and their simulators. *Nonlinear Analysis, Theory, Methods and Applications*, 11(6):665–683, 1987.

[75] C.J. Tomlin, J. Lygeros, L. Benvenuti, and S.S. Sastry. Output tracking for a non-minimum phase dynamic CTOL aircraft model. In *IEEE Conference on Decision and Control*, pages 1867–1872, 1995.

[76] C.J. Tomlin, J. Lygeros, and S.S. Sastry. Synthesizing controllers for nonlinear hybrid systems. In S. Sastry and T.A. Henzinger, editors, *Hybrid Systems: Computation and Control*, number 1386 in LNCS, pages 360–373. Springer-Verlag, Berlin, 1998.

[77] C.J. Tomlin, J. Lygeros, and S.S. Sastry. Computing controllers for nonlinear hybrid systems. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, number 1569 in LNCS, pages 238–255. Springer-Verlag, Berlin, 1999.

[78] C.J. Tomlin, G.J. Pappas, and S.S. Sastry. Conflict resolution for air traffic management: A case study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, 1998.

[79] Claire Tomlin. *Hybrid Systems with Application to Air Traffic Management*. PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1998.

[80] V. I. Utkin. *Sliding Modes in Control and Optimization*. Springer-Verlag, Berlin, 1992.

[81] A.J. van der Schaft and H. Schumacher. Complementarity modeling of hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):483–490, 1998.

[82] P. Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, AC-38(2):195–207, 1993.

[83] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *IEEE Conference on Decision and Control*, pages 4607–4613, San Diego, California, December 10–12 1997.

[84] L. Yang and J.K. Kuchar. Prototype conflict alerting logic for free flight. *Journal of Guidance Control and Dynamics*, 20(4):768–773, 1997.

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

[86] L. C. Young. *Optimal Control Theory*. Chelsea, second edition, 1980.

[87] J. Zhang, K.H. Johansson, J. Lygeros, and S.S. Sastry. Zeno hybrid systems. *International Journal of Robust and Nonlinear Control*, 11:435–451, 2001.