Foundations and Trends® in Systems and Control
Vol. 1, No. 1 (2014) 1–172

DOI: 10.1561/2600000001

now
the essence of knowledge

Hybrid Dynamical Systems: An Introduction to Control and Verification

Hai Lin
University of Notre Dame
hlin1@nd.edu

Panos J Antsaklis
University of Notre Dame
antsaklis.1@nd.edu

Contents

Abstract

Hybrid dynamical systems are a class of complex systems that involve interacting discrete-event and continuous-variable dynamics. They are important in applications in embedded systems, cyber-physical systems, robotics, manufacturing systems, traffic management, biomolecular networks, and have recently been at the center of intense research activity in the control theory, computer-aided verification, and artificial intelligence communities. This paper provides a tutorial introduction to this multidisciplinary research area. A number of fundamental topics, such as modeling, abstraction, verification, supervisory control, stability analysis, stabilization, and optimal control of hybrid systems are introduced and discussed. Additionally, more advanced topics are briefly discussed at the end of each chapter with references given for further reading.

H. Lin and P.J. Antsaklis. *Hybrid Dynamical Systems: An Introduction to Control and Verification*. Foundations and Trends® in Systems and Control, vol. 1, no. 1, pp. 1–172, 2014.

DOI: 10.1561/2600000001.

1

Introduction

Hybrid dynamical systems contain heterogeneous dynamics that interact with each other and determine their behaviors over time. By heterogeneity, we mean systems containing two different kinds of dynamics: Time-driven continuous variable dynamics, usually described by differential or difference equations; and event-driven discrete variable dynamics, the evolutions of which depend on if-then-else type of rules, usually described by automata or Petri nets. These two kinds of dynamics interact with each other and generate complex dynamical behaviors, such as switching once the value of a continuous variable passes through a threshold, or state jumping upon certain discrete event occurring to mention but a few.

As an example, consider a typical room temperature control system in winter. Assume that the set point of the thermostat is 70 degrees Fahrenheit. The furnace will turn on if the room temperature is below the set point, and turn off otherwise. The room temperature control system is actually a typical hybrid system as the furnace, along with the heat flow characteristics of the room, form the continuous variable dynamics, whereas the on-off thermostat can be modeled as a discrete event system with two states "ON" and "OFF". In addition, the tran-

sition between these two discrete states is triggered by the temperature in the room, while the evolution of the temperature depends on whether the furnace is on or off, i.e., discrete state of the thermostat. Hence, the temperature control system contains interacting discrete and continuous dynamics, and can be modeled and studied as a hybrid system.

Hybrid systems actually arise in a great variety of applications, such as manufacturing systems [Pepyne and Cassandras, 2000], air traffic management [Tomlin et al., 1998], automotive engine control [Balluchi et al., 2000], chemical processes [Engell et al., 2000], to mention but a few. Hybrid systems also arise from the hierarchical organization of complex systems, and from the interaction of discrete planning algorithms and continuous control algorithms in autonomous, intelligent systems. Hybrid systems have a central role in networked embedded control systems that interact with the physical world, and as such play a key role in the understanding of the evolution of systems that contain an information and networking core and interact tightly with the physical world and human operators; such systems are also referred to as Cyber-Physical Systems (CPS) [Lee, 2008, Baheti and Gill, 2011]. Studies in hybrid systems could provide a unified modeling framework for CPS, and systematic methods for performance analysis, verification, and design.

Besides their enormous practical importance, hybrid systems also represent a fascinating and highly challenging area of study that encompasses a variety of theoretical research questions. Actually, the introduction of switching and state jumps in hybrid systems nontrivially extend the dynamical behaviors that can be modeled by hybrid system models compared with traditional modeling frameworks, such as ordinary differential equations and automata. Hence, hybrid system models are of interest in themselves, and have been successfully used to model a large variety of complex systems, such as gene-regulatory networks [De Jong, 2002], communication networks [Hespanha, 2004b] and robotic systems [Egerstedt, 2000]. However, the price associated with the increased modeling power is the difficulty in analyzing properties of the evolution or solution of a hy-

brid system model, such as the existence and uniqueness of a solution, and the continuity of trajectories with respect to initial conditions. These difficulties have motivated significant and intense research activities targeting formal analysis and synthesis of hybrid systems. On the other hand, the introduction of switching logic into controllers may help to achieve performance that exceeds any fixed classical linear or nonlinear smooth controller; for example, there are some nonlinear systems that cannot be stabilized by any smooth feedback control law, but can be asymptotically stabilized by a hybrid controller [Hespanha et al., 1999]. Moreover, to meet challenging high-performance design requirements that reflect multiple objectives such as response speed, accuracy, optimality, robustness, and disturbance attenuation, a multi-modal (hybrid) control architecture may be the proper choice. When the requirements are represented by time and event-based behaviors or when the plant to be controlled has tight interactions of continuous variable and discrete event dynamics, one needs to employ hybrid control methods [Antsaklis and Nerode, 1998, Antsaklis, 2000].

The history of hybrid system research can be traced back at least to the 1960s to the study of engineering systems that contained relays and/or hysteresis, see e.g., [Witsenhausen, 1966]. However, hybrid systems began to seriously attract the attentions of researchers in the early 1990s, mainly because of the widespread development and implementation of digital micro controllers and embedded devices. The last two decades have seen considerable research activities in modeling, analysis and synthesis of hybrid systems. The investigation of hybrid systems is a fascinating discipline bridging control engineering, mathematics and computer science.

Computer scientists tend to look at hybrid systems primarily as discrete (computer) programs interacting with the physical environment. They extend their computational models, such as finite state machine, automata and petri nets, from discrete systems to hybrid systems by embedding the continuous variable dynamics into these discrete models, see e.g., [Alur et al., 1993, 1995]. Typically, these approaches are able to deal with complex discrete dynamics and empha-

size analysis results (verification) and simulation methodologies. Such approaches typically ask whether certain properties, such as safety, liveness and fairness that are formulated in temporal logic formulas, hold true or not for a given hybrid system model. This is called the verification of hybrid systems, and one of the main verification methods is symbolic model checking, which is based on the computation of reachable sets for hybrid systems. Consequently, a good deal of research effort has focused on developing sophisticated techniques drawn from optimal control, game theory, and computational geometry to calculate or approximate the reachable sets for various classes of hybrid systems, see e.g., [Chutinan and Krogh, 2003, Tomlin et al., 2003].

On the other hand, researchers from the areas of dynamical systems and control theory have approached hybrid systems as a collection of differential/difference equations with discontinuous or multi-valued right-hand sides. Representative modeling frameworks in this category include piecewise affine/linear systems [Sontag, 1981, Johansson, 2003a] and switched systems [Liberzon, 2003, Lin and Antsaklis, 2009]. They extend the models and methodologies for traditional continuous variable systems, such as ordinary differential/difference equations, by including discrete variables so as to describe the jumping or switching phenomena. Typically, these approaches are able to deal with complex continuous variable dynamics and focus mainly on stability, controllability, robustness and synthesis issues.

The methods for hybrid systems are distributed across a wide spectrum, ranging from methods known in the discrete (cyber-) domain at one end, to traditional approaches for the continuous physical systems at the other. Rooted at opposite ends, both computer scientists and control theorists have made significant contributions to the field of hybrid systems by extending traditional methods from the purely discrete or continuous domain to deal with hybrid systems. However, in general, there has been little work on integrating methods from these two domains. This is possibly because the formal methods pursued in computer science traditionally lie in the realm of discrete

mathematics, while control theory approaches lie mainly in the realm of continuous mathematics. A noticeable trend in the recent hybrid system literature emphasizes the synthesis of hybrid controllers for continuous or hybrid dynamical systems to satisfy complicated temporal logic specifications. This is known as symbolic control or hybrid supervisory control, which can be seen as a crosstalk between these two schools of thoughts.

This tutorial paper seeks to balance the emphasis on methods from both computer science and control theory, and hopefully gives the readers a relatively complete picture of the whole field of hybrid dynamical systems. For such a purpose, the rest of paper is organized as follows. First, several modeling frameworks for hybrid systems, namely hybrid automata, switched systems and piecewise affine systems, are introduced in Chapter 2. Chapter 3 briefly reviews the results on stability, stabilization and optimal control of hybrid systems. Then, Chapter 4 investigates the verification problems for hybrid systems with a particular focus on model checking approaches. Finally, Chapter 5 reviews the developments of hybrid supervisory control, also known as symbolic control, which can be seen as an effort to combine the results and approaches from both systems and control theory and computer sciences.

2

Modeling of Hybrid Systems

In this chapter, we will introduce several modeling frameworks for hybrid systems. First, we start with a very general modeling framework, that of hybrid automata. Hybrid automata models can be seen as extensions of finite automata by embedding continuous dynamics in each of the discrete modes. In this chapter, we will first give a very brief review of discrete event systems and the formal automata theory. Then, hybrid automata will be introduced with focus on the characterization of their trajectories and well-posedness. Hybrid automata were proposed in the computer science literature to model hybrid systems arising from computer programs interacting with the physical world. In contrast, control engineers usually consider hybrid systems arising from physical dynamical systems controlled by digital circuits. The introduction of digital circuits causes difficulties because of the switching of dynamics and resetting of states. In response, control engineers have extended traditional models, e.g., state space models, by explicitly introducing switchings and state jumps to model hybrid systems. In particular, two popular modeling frameworks for hybrid systems proposed in control, namely switched systems and piecewise affine systems, are reviewed. It is also shown that both switched sys-

tems and piecewise affine systems can be written in the form of hybrid automata. In addition, piecewise affine systems can be seen as a special case of switched systems with linear dynamics in each mode and mutual exclusive partitions of the state space.

2.1 Finite Automata

Finite automata are popular models for discrete event systems. A discrete event system(DES) is a dynamical system which has discrete valued states and the evolution among states is triggered by the occurrence of discrete events [Cassandras and Lafortune, 2008, Ramadge and Wonham, 1989]. For example, the operation of a vending machine can be seen as a discrete event system. Usually the machine remains in the "ready" mode or state, which can be seen as the initial state of the system. When a customer inserts enough coins, the vending machine will change into "waiting" mode and wait for the customer's choice for the drink. According to the customer's choice, the machine will respond with either "Coke" or "Pepsi" respectively and deliver the product. After this round of service, the machine will return back to the "ready" state and wait for the next customer.

Thus the system has four states - "ready", "waiting", "Output Coke" and "Output Pepsi", and four events - "coins received", "Coke being chosen", "Pepsi being chosen" and "Drink being taken." Let's denote the set of states as a set $X = \{q_0, q_1, q_2, q_3\}$, which is a discrete finite symbol set and corresponds to the four discrete states of the vending machine respectively. For simplicity, we denote the event set as $\Sigma = \{coin,\ Coke,\ Pepsi, taken\}$.

Then, the behavior of the system can be described by a collection of sequences of triples consisting of states, events and time instants of the form $(q_0, \sigma_1, t_1)(q_1, \sigma_2, t_2)\cdots$, where $q_0 \in X$ is the initial state (here it is q_0), and for each $i \geq 1$, $q_i \in X$ stands for the ith state of the system, $\sigma_i \in \Sigma$ denotes the ith event, and $t_i \in \mathbb{R}^+$ is the time instant when the ith event σ_i occurs. A typical state trajectory of the vending machine example is plotted in Figure 2.1.

Usually in DES, we ignore the timing information and focus on

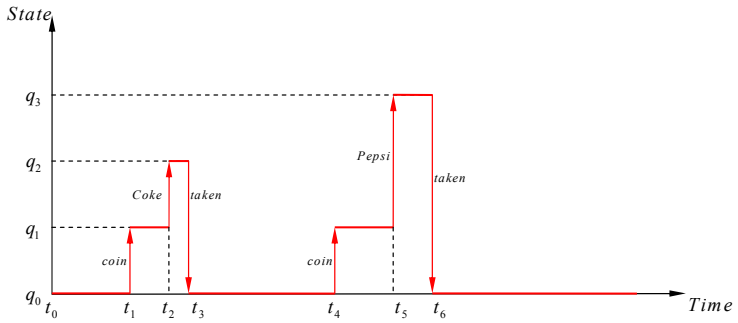


Figure 2.1: A typical state trajectory of the vending machine example, and the transition of states is triggered by discrete events.

the ordering of state and event pairs. The reasons are twofold. First, when exactly an event happens is not as significant as what the consequences of the event will be. In our case, the consequences are reflected by state transitions in the DES. Secondly, the relationships between state transitions and events are complex and cannot be described using differential or difference equations. In addition, no one can predict when exactly such an event is going to occur. Hence, a logical model is sufficient to study the qualitative properties of the DES. Finite automata are popular logical models that have been used in practice successfully. Our treatment of finite automata mainly follows a classical textbook [Hopcroft et al., 2006].

Definition 2.1. A finite automaton A is a tuple $(Q, Q_0, \Sigma, \delta, F)$ where

- Q is a finite set of states;
- $Q_0 \subseteq Q$ is the set of initial states;
- Σ is a finite set of symbols representing inputs;
- $\delta : Q \times \Sigma \to 2^Q$ represents the state transition relation;
- $F \subseteq Q$ is a set of accepting states.

Here, 2^Q stands for the collection of all subsets of Q, i.e., the power set of Q. Note that the map $\delta(q, \sigma)$ specifies the set of states that the

current state q can move into when it reads the symbol σ. It is worth pointing out that the set $\delta(q, \sigma)$ maybe empty for all $\sigma \in \Sigma$, which means that the automaton A gets stuck at the state q and has nowhere to go. When this happens, the automaton A is called *blocking*. Namely, $\exists q$, such that for $\forall \sigma \in \Sigma, \delta(q, \sigma) = \emptyset$. Otherwise, A is *non-blocking*.

On the other hand, the set $\delta(q, \sigma)$ may contain more than one states, which means that there may exist many possible transitions for each state and symbol. Hence, the automaton A is called *non-deterministic* for this case. A non-deterministic finite automaton can have many different runs on a given input word. In contrast, an automaton A is *deterministic* if $|Q_0| = 1$ and $|\delta(q, \sigma)| \leq 1$ for all $q \in Q$, and $\sigma \in \Sigma$. Here, $|\cdot|$ denotes the cardinality of a set. A deterministic finite automaton can have at most one run on a given input word.

The dynamical behavior of a finite automaton is conveniently described by the strings of its state evolution. Formally, we have the following definitions and notations.

Definition 2.2. A set of symbols, $\Sigma = \{a_1, a_2, \cdots\}$, is called an alphabet. A *word* s is a sequence of symbols chosen from the alphabet Σ.

Next we define some notations on words that will be used later.

- The length of a word s is denoted as $|s|$. Let $s = a_1 a_2 \cdots a_m$, then $|s| = m$, the i-th symbol of s, denoted as $s(i) = a_i$ for $i \leq |s|$. Prefix of s with length i, $s[i] = a_1 a_2 \cdots a_i$.

- Σ^k stands for the set of all strings of length k with symbols from Σ. In other words, $\Sigma^k = \{s, |s| = k\}$.

- Σ^* stands for the set of all words of finite length with symbols from Σ, that is
$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots,$$
where $\Sigma^0 = \{\varepsilon\}$ (empty string).

- Σ^ω stands for the set of all words of infinite length with symbols from Σ.

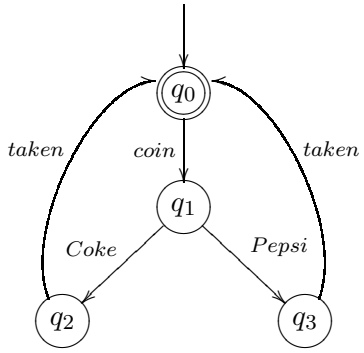
Definition 2.3. A *run* r of a finite automaton $A = (Q, Q_0, \Sigma, \delta, F)$ on a finite word $s = a_0 a_1 \cdots a_{n-1}$ is a sequence $q_0, q_1, \cdots, q_n$ of $n+1$ states in Q such that $q_0 \in Q_0$, and $q_{i+1} \in \delta(q_i, a_i)$ for $0 \leq i < n$.

To illustrate the definitions, we revisit the vending machine example.

Example 2.1. Formally, the vending machine can be modeled as a finite automaton $A = (Q, Q_0, \Sigma, \delta, F)$, where

- $Q = \{q_0, q_1, q_2, q_3\}$;
- $Q_0 = \{q_0\}$;
- $\Sigma = \{coin, Coke, Pepsi, taken\}$;
- $\delta(q_0, coin) = \{q_1\}$, $\delta(q_1, Coke) = \{q_2\}$, $\delta(q_1, Pepsi) = \{q_3\}$, $\delta(q_2, taken) = \{q_0\}$, $\delta(q_3, taken) = \{q_0\}$;
- $F = \{q_0\}$.

Intuitively, the dynamics of the system can be modeled by the following digraph.



Here, the nodes stand for the discrete states in Q respectively, while the double-circled nodes represent the marked states in F, and the initial state q_0 is pointed by a small arrow without source nodes. The edges between nodes correspond to the state transition relation δ with the triggering events labeled. A run of the system S could be: $q_0 \longrightarrow^{coin} q_1 \longrightarrow^{Coke} q_2 \longrightarrow^{taken} q_0 \cdots$. It means that a customer inserts a coin to the vending machine, and the machine asks the customer to choose between "Coke" and "Pepsi". After the customer

chooses "Coke", the machine responds and returns to the ready state q_0. It can be easily verified that the automaton for the vending machine example is deterministic and non-blocking. □

If a word s has a valid run in the sense of Definition 2.3, then s is called a generated word of the automaton A. The collection of all such generated words is called the *language generated by* A, denoted as $\mathcal{L}(A)$. A run $r = q_0, q_1, \cdots, q_n$ is *accepting* if $q_n \in F$, and the word s is accepted by A if A has an accepting run on s. The collection of all finite words $s \in \Sigma^*$ accepted by A is called the (marked) *language accepted by* A, denoted as $\mathcal{L}_M(A)$. Given a finite automaton A, the pair $(\mathcal{L}(A), \mathcal{L}_M(A))$ is called the *language model* of A. Two automata are called *language equivalent* if they have the same language model.

2.1.1 Properties of Finite Automata

An important property of finite automata is their closure under Boolean operations, i.e., the union or intersection of two finite automata is also finite automaton; so is the negation of a finite automaton. To be more precise, we formally state the property in the following propositions:

Theorem 2.1. Let A_1, A_2 be two finite automata. Then

- there exists a finite automaton A such that $\mathcal{L}(A) = \mathcal{L}(A_1) \cup \mathcal{L}(A_2)$ and $\mathcal{L}_M(A) = \mathcal{L}_M(A_1) \cup \mathcal{L}_M(A_2)$;
- there also exists a finite automaton A' such that $\mathcal{L}(A') = \mathcal{L}(A_1) \bigcap \mathcal{L}(A_2)$, and $\mathcal{L}_M(A') = \mathcal{L}_M(A_1) \bigcap \mathcal{L}_M(A_2)$.
- Furthermore, there exists a finite automaton $\bar{A_1}$ such that $\mathcal{L}_M(\bar{A_1}) = \mathcal{L}(A_1) - \mathcal{L}_M(A_1)$ and $\mathcal{L}(\bar{A_1}) = \mathcal{L}(A_1)$.

The proof can be found in any textbooks on automata theory, e.g., [Hopcroft et al., 2006]. Another important result about finite automata is language equivalence between deterministic finite automata and non-deterministic finite automata as stated in the following theorem.

Theorem 2.2. Let A be a non-deterministic finite automaton. Then there exists a deterministic finite automaton A_d such that $\mathcal{L}(A_d) = \mathcal{L}(A)$ and $\mathcal{L}_M(A_d) = \mathcal{L}_M(A)$.

In other words, for any non-deterministic finite automaton A, we can always find a deterministic finite automaton A_d that is language equivalent to A. Such a deterministic finite automaton A_d is not necessarily unique, and can be constructed from A by a so-called *subset construction*, see e.g., [Hopcroft et al., 2006]. So, there is no loss of generality for us to focus on deterministic finite automata from the language sense. Another important fact about finite automata is that the class of languages accepted by finite automata are called *regular languages*. It is usually convenient to represent the language accepted by a finite automaton by *regular expressions*. For example, the vending machine automaton accepts a language that can be expressed as $(coin \cdot (Coke + Pepsi) \cdot taken)^*$, where the operator "$\cdot$" stands for concatenation between events (usually ignored), "$+$" means choice between events, and "*" indicates any finite number of repeating of an events or string of events. The above expression characterizes the language accepted by the vending machine automaton, which can be presented as finite number of repeats of the pattern - the event $coin$ followed by either the event $Coke$ or the event $Pepsi$, and then the event $taken$. Note that nothing happening, denoted as ϵ, is also acceptable as it corresponds to zero repeating of such a pattern. Its generated language is the *prefix closure* of the generated language , i.e., the set of all prefixes of the strings in $(coin \cdot (Coke + Pepsi) \cdot taken)^*$, which is denoted as $pr((coin \cdot (Coke + Pepsi) \cdot taken)^*)$.

On the other hand, given a regular language K, we can build a finite automaton (not unique, but there is always a deterministic finite automaton) that accepts K or generates K (if K is also prefix closed, i.e., $K = pr(K)$). Hence, regular language is also closed under union, intersection and complementation [Hopcroft et al., 2006].

2.2 Hybrid Automata

Finite automata have been successfully used in modeling and studying typical discrete event systems, such as communication protocols and computer programs, where the logic correctness (e.g., deadlock free) is the main concern [Ramadge and Wonham, 1989,

Cassandras and Lafortune, 2008]. However, finite automata cannot model for example cyber-physical systems, where both discrete event dynamics and continuous physical dynamics coexist and interact with each other [Lee, 2008, Baheti and Gill, 2011]. Hybrid automata provide formal models for cyber-physical systems, and can be seen as an extension of finite automata by adding continuous dynamics into each of its discrete states (also called modes). Each mode is associated with constraints within which the continuous dynamics can evolve. Edges between modes are annotated with guards that specifies the conditions when the mode transition can be triggered; each edge is also associated with a reset map indicating how the continuous variables are being updated after the discrete transition. Following [Alur et al., 1995, Henzinger, 1995, Lygeros et al., 2003], hybrid automata are defined below (with restrictions for simplicity).

Definition 2.4. A hybrid automaton H is a collection $H = \{Q, X, f, Init, Inv, E, G, R\}$, where

- $Q = \{q_1, q_2, \cdots\}$ is a finite set of discrete states;
- $X \subseteq \mathbb{R}^n$ represents the state space where the continuous state variables take values;
- $f : Q \times X \to \mathbb{R}^n$ assigns to each discrete state $q \in Q$ an analytic vector field $f(q, \cdot)$;
- $Init \subseteq Q \times X$ is the set of initial states;
- $Inv : Q \to 2^X$ assigns to each discrete state $q \in Q$ a set $Inv(q) \subseteq X$ called the invariant set;
- $E \subseteq Q \times Q$ is the set of discrete transitions;
- $G : E \to 2^X$ assigns to each discrete transition $(q, q') \in E$ a guard set $G(q, q') \subset X$;
- $R : E \times X \to 2^X$ is a reset map.

We refer to (q, x), where $q \in Q$ and $x \in X$, as the state of H. As illustration, let's model the temperature control system in the introduction as a hybrid automaton.

Example 2.2. Consider the temperature control system example. The system contains two discrete modes corresponding to the on-off operation of the thermostat. The temperature follows different continuous dynamics depending on whether the furnace is on or off. It is assumed that the furnace is initially off and the temperature is below 60 degree. As we start the temperature control process, the furnace is turned on and remains on heating the room as long as the temperature is below 70 degree. Once the room temperature reaches 70 degree, the furnace will be commanded to turn off. Due to latency and other practical reasons, the furnace actually turns off just before the temperature hits 71 degree. As the furnace is off, the room temperature starts dropping due to heat losses. Once the temperature drops below 70 degree, the furnace will be commanded to turn on; for practical reasons it actually turns on just before the temperature drops to 69 degree. Formally, the temperature control system can be modeled as a hybrid automaton with elements in the model being identified as follows.

- $Q = \{\text{ON}, \text{OFF}\}$;
- $X = \mathbb{R}$ denotes the range of the room temperature;
- $f(\text{ON}, x) = -x + 100$ and $f(\text{OFF}, x) = -x$;
- $Init = \{OFF\} \times \{x \leq 60\}$;
- $Inv(\text{ON}) = \{x \in \mathbb{R} : x \leq 71\}$,
  and $Inv(\text{OFF}) = \{x \in \mathbb{R} : x \geq 69\}$;
- $E = \{(\text{ON}, \text{OFF}), (\text{OFF}, \text{ON})\}$;
- $G(\text{ON}, \text{OFF}) = \{x \in \mathbb{R} : x \leq 70\}$,
  and $G(\text{OFF}, \text{ON}) = \{x \in \mathbb{R} : x \geq 70\}$;
- $R((\text{ON}, \text{OFF}), x) = \{x\}$, and $R((\text{OFF}, \text{ON}), x) = \{x\}$.

□

A hybrid automaton can be visualized as a directed graph. To graphically represent a hybrid automaton, we first draw a directed graph (V, E) with one to one mapping between the vertices V and the discrete state Q, while E is the same as in the definition of a hybrid

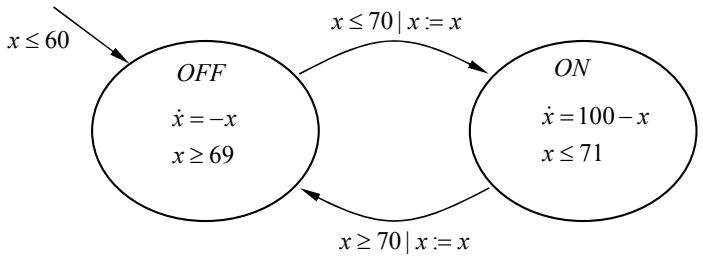


Figure 2.2: Graphical representation of the temperature control hybrid automaton model.

automaton. Secondly, for each vertex of the graph, we specify the discrete mode $q \in Q$, the differential equation implied by the vector field $\dot{x} = f(q, x)$, and the invariant condition $Inv(q)$ in each vertex of the graph. On each edge $(q, q') \in E$, the guard condition $G(q, q')$ and the reset map $x' := R((q, q'), x)$ are specified. Finally, the initial state may be marked by an arrow (an edge without a source vertex) pointing at the vertex $q \in Q_0$ with the set $Init(q)$ specified on it.

To illustrate the process, we plot the graphical representation of the hybrid automaton model for the temperature control example in Figure 2.2, which illustrates the switching of continuous dynamics due to discrete transitions. Another important dynamical phenomenon is that the continuous state may suddenly change its value (state jump) upon a discrete transition as well. To illustrate such a phenomenon, consider the following example from [Johansson et al., 1999].

Example 2.3. Consider a bouncing ball as shown in Figure 2.3. The vertical position of the ball is denoted by x_1 and the velocity by x_2. As long as the ball is above the ground ($x_1 > 0$), the continuous dynamics can be presented as, $\dot{x}_1 = x_2; \dot{x}_2 = -g$, where g is the gravity constant. When the ball hits the ground ($x_1 = 0$), a discrete jump takes place. The speed x_2 is reset according to $x_2 := -cx_2$, where $c \in (0, 1)$ is a coefficient of restitution.

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

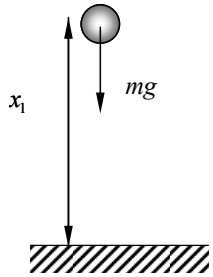


Figure 2.3: A bouncing ball example.

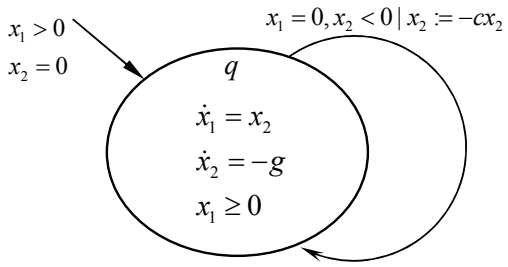


Figure 2.4: Hybrid automata model for the bouncing ball example.

- $Q = \{q\}$;
- $X = \mathbb{R}^2$;
- $f(q, x) = \begin{bmatrix} x_2 \\ -g \end{bmatrix}$;
- $Init = \{q\} \times \{x \in \mathbb{R}^2 | \ x_1 > 0, x_2 = 0\}$;
- $Inv(q) = \{x \in \mathbb{R}^2 | \ x_1 \geq 0\}$;
- $E = \{(q, q)\}$;
- $G(q, q) = \{x \in \mathbb{R}^2 | x_1 = 0, x_2 < 0\}$;
- $R((q, q), x) = \{x_1 = x_1, x_2 = -cx_2\}$.

Then its graphical representation of the hybrid automaton model

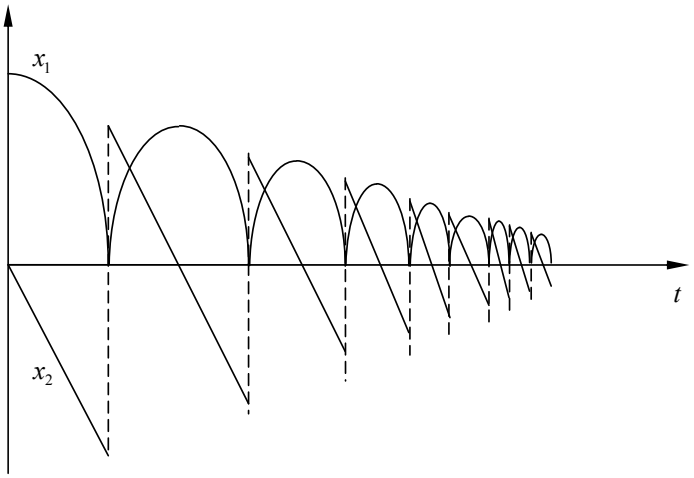


Figure 2.5: Typical trajectories for the position x_1 and velocity x_2 of the bouncing ball example.

is plotted in Figure 2.4. It only contains a single discrete mode, while discrete (self-)transitions occur when the ball hits the ground ($x_1 = 0$ and $x_2 < 0$, i.e., the guard set condition $G(q, q)$ holds). Accompanying with the transition, the velocity x_2 gets reset, namely the velocity x_2 changes its direction suddenly after bouncing ($x_2 := -cx_2$, with $0 < c < 1$). □

For illustration, a typical trajectory for the position and velocity of the bouncing ball example is plotted in Figure 2.5. Usually, we also need to consider the evolution of the discrete states of a hybrid automaton. In this example, there is only one discrete state, so the time evolution of discrete state is q for the whole duration.

2.2.1 Hybrid State Trajectories

Given a hybrid automaton, a typical state trajectory can be described as follows: Starting from a point in an initial region, say $(q_0, x_0) \in Init$, first, the q_0-th mode dynamics are followed, i.e., $\dot{x} = f(q_0, x)$, with initial condition $x(0) = x_0$ while $x(t) \in Inv(q_0)$. Assume that the con-

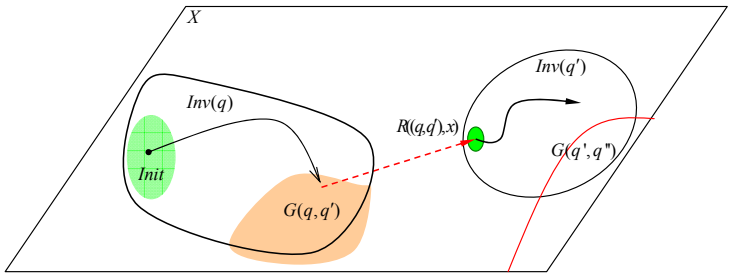


Figure 2.6: A typical trajectory of a hybrid automaton.

tinuous state $x(t)$ evolves into the guard set $G(q_0, q_1)$ and a discrete transition from q_0 to q_1 occurs at time t_1, where $x(t_1) \in G(q_0, q_1)$. Accompanying the transition, the continuous state x is also being reset by following the reset map $R((q_0, q_1), x(t_1))$, which is contained in the invariant set of mode q_1, $Inv(q_1)$. Then, the continuous state x evolves again following the dynamics of mode q_1 as illustrated in Figure 2.6.

Next, we formally define the hybrid time trajectory and hybrid executions.

Definition 2.5. [Lygeros et al., 2003] A hybrid time trajectory is a finite or infinite sequence of intervals $\mathcal{T} = \{I_i\}_{i=1}^{N}$, where

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

A hybrid time trajectory is a sequence of intervals of the real line, the end points of which overlap. The interpretation is that the end points of the intervals are the times at which discrete transitions take place. Hybrid time trajectories can be extended to infinity if $\mathcal{T}$ is an infinite sequence or if it is a finite sequence ending with an interval of the form $[\tau_N, \infty)$. We use $t \in \mathcal{T}$ as shorthand notation that there exists i such that $t \in I_i$ with $I_i \in \mathcal{T}$. Each hybrid time trajectory $\mathcal{T}$ is linear ordered by the relation $\prec$, which is defined as follows. For $t_1, t_2 \in \mathcal{T}$, there exist i and j such that $t_1 \in [\tau_i, \tau_i']$ and $t_2 \in [\tau_j, \tau_j']$. We say $t_1 \prec t_2$ if $t_1 < t_2$ or $i < j$.

The following operations defined in [Lygeros et al., 2003] on the hybrid time trajectory $\mathcal{T}$ are useful for the definition and classification of executions of a hybrid automaton H.

- Prefix: $\mathcal{T} = \{I_i\}_{i=1}^{N}$ is said to be a prefix of $\mathcal{T}' = \{I'_i\}_{i=1}^{M}$, denoted as $\mathcal{T} \sqsubseteq \mathcal{T}'$ if $\mathcal{T} = \mathcal{T}'$ or $\mathcal{T}$ is finite and $N \leq M$, $I_i = I'_i$ for $i = 1, \cdots, N$.
- Index set: Define the index set of $\mathcal{T} = \{I_i\}_{i=1}^{N}$ as $< \mathcal{T} >= \{1, 2, \cdots, N\}$ if N is finite, or $< \mathcal{T} >= \mathbb{N}$ if N is infinite.
- Length: Define the length of $\mathcal{T} = \{I_i\}_{i=1}^{N}$ as $|\mathcal{T}| = \sum_{i \in < \mathcal{T} >} (\tau'_i - \tau_i)$.

Next we introduce a concept for hybrid automata similar to a solution of a continuous dynamical system. We use q and x to also denote the time evolution of the discrete and continuous state, respectively, with a slight abuse of notation. An *execution* of a hybrid automaton H is a collection $\mathcal{X} = (\mathcal{T}, q, x)$, where

- $\mathcal{T}$ is a hybrid time trajectory;
- $q :< \mathcal{T} >\rightarrow Q$ is a map; and
- $x = \{x^i, i \in < \mathcal{T} >\}$ is a collection of differentiable maps $x^i : I_i \rightarrow X$ such that

  1. $(q(0), x(0)) \in Init$;
  2. for all $t \in [\tau_i, \tau'_i]$, $\dot{x}^i(t) = f(q(i), x^i(t))$ and $x^i(t) \in Inv(q(i))$;
  3. for all $i \in < \mathcal{T} > / N$, $e = (q(i), q(i+1)) \in E$, and $x^i(\tau'_i) \in G(e)$, $x^{i+1}(\tau_{i+1}) \in R(e, x^i(\tau'_i))$.

In Figure 2.7, a typical execution of a hybrid automaton is illustrated, where the horizontal t-axis represents a hybrid time trajectory. It is noticeable that hybrid executions consist of concatenation of several pieces of continuous flow x^i. There are discrete transitions and

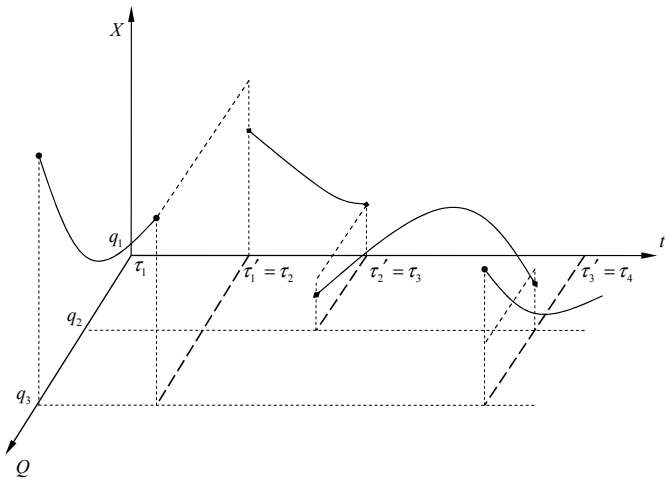


Figure 2.7: A typical trajectories of a hybrid automaton.

possible state jumps between any two successive pieces of continuous flow. Note that multiple discrete transitions may take place at the same τ_i. Continuous flows keep the discrete part q of the hybrid state constant; the continuous part evolves over time according to the differential equation $\dot{x} = f(q, x)$, as long as $x \in Inv(q)$. If during the continuous flow it happens that the continuous part hits a guard set, i.e., $x \in G(e)$ for some $e = (q, q') \in E$, then the edge e becomes enabled. The state may then instantaneously jump from (q, x) to any (q', x') with $x' \in R(e, x)$. Then the process repeats, and the continuous parts of the state evolve according to the differential equation $\dot{x} = f(q', x)$.

Example 2.4. [Johansson et al., 1999] Consider a two tank system as shown in Figure 2.8. Water is added to these two tanks through a hose (with constant flow rate w), and the goal is to keep the water levels of the tanks above certain level, r_1 and r_2. However, both tanks are leaking; also the hose is dedicated to only one tank, either Tank 1 or Tank 2, at any time instant.

It is assumed that the leaking rates for both tanks are constant, and are v_1 and v_2 respectively. Let x_i denote the level of water in Tank i, for $i = 1,\ 2$. Also assume that the initial water levels are above r_1 and

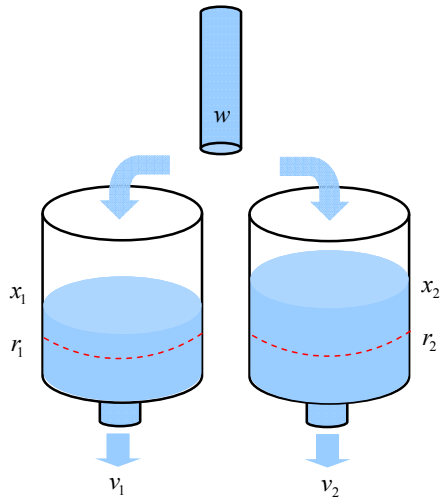


Figure 2.8: A water tank example.

r_2 respectively.

To make sure that the water level of Tank i is above r_i for $i = 1,\ 2$, we employ the following control law: switching 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:

- $Q = \{q_1, q_2\}$;
- $X = \mathbb{R}^2$;
- $f(q_1, x) = \begin{bmatrix} w - v_1 \\ -v_2 \end{bmatrix}$ and $f(q_2, x) = \begin{bmatrix} -v_1 \\ w - v_2 \end{bmatrix}$;
- $Init = \{q_1, q_2\} \times \{x \in \mathbb{R}^2 |\ x_1 \geq r_1,\ x_2 \geq r_2\}$;
- $Inv(q_1) = \{x \in \mathbb{R}^2 |\ x_2 \geq r_2\}$, and $Inv(q_2) = \{x \in \mathbb{R}^2 |\ x_1 \geq r_1\}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbb{R} | x_2 \leq r_2\}$, and $G(q_2, q_1) = \{x \in \mathbb{R} | x_1 \leq r_1\}$;
- $R((q_1, q_2), x) = \{x\}$, and $R((q_2, q_1), x) = \{x\}$.

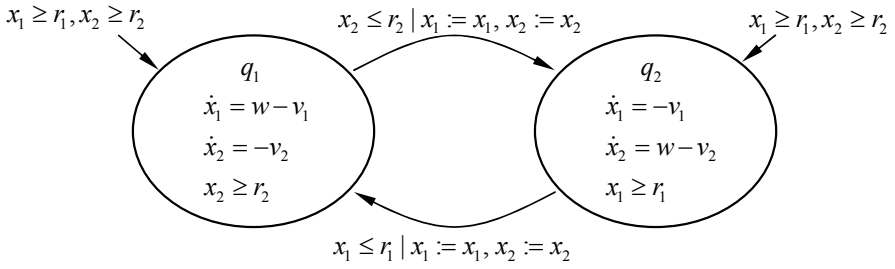


Figure 2.9: The hybrid automata model for the water tank example.

The graphical presentation of the hybrid automaton model is plotted in Figure 2.9. It can be shown that the water tank hybrid automaton accepts a unique infinite execution for each initial state. As an illustration, a typical state trajectory is plotted in Figure 2.10 for initial condition $x_1(0) \geq r_1$ and $x_2(0) \geq r_2$, under the assumption that $\max\{v_1, v_2\} < w < v_1 + v_2$.

It can be seen that the switching between the two tanks becomes faster and faster, and the state trajectories converge to the point $(r_1, r_2)^T$. This phenomenon is independent of the initial conditions provided that $x_1(0) \geq r_1$ and $x_2(0) \geq r_2$. In other words, the region $Init = \{q_1, q_2\} \times \{x \in \mathbb{R}^2 | \ x_1 \geq r_1, \ x_2 \geq r_2\}$ is invariant, which means that all trajectories start from this region will stay there.

However, the condition, $\max\{v_1, v_2\} < w < v_1 + v_2$, implies that the rate at which water is added to the system is less than the rate at which water is removed. Physical intuition suggests that in this case at least one of the water tanks will have to eventually become empty. Why does the analysis of the hybrid automaton fail to predict that? This is because there are infinite number of switches within a finite time interval, called Zeno phenomenon in the literature, see e.g., [Johansson et al., 1999]. However, every switch of the hose between tanks actually takes some time in reality, so the Zeno phenomenon predicted by the hybrid automata model in Figure 2.9 will not happen in the real world. □

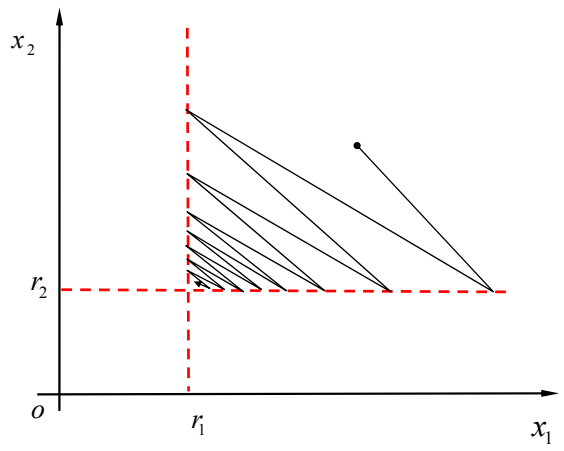


Figure 2.10: Illustration of a typical trajectory of the water tank example.

Motivated by the complexity of dynamical behaviors, in the following we classify the executions of hybrid automata. We first introduce the prefix definition.

Definition 2.6. [Lygeros et al., 2003] $\mathcal{X} = (\mathcal{T}, q, x)$ is said to be a prefix of $\hat{\mathcal{X}} = (\mathcal{T}', \hat{q}, \hat{x})$, denoted as $\mathcal{X} \sqsubseteq \hat{\mathcal{X}}$ if $\mathcal{T} \sqsubseteq \mathcal{T}'$, and for all $i \in < \mathcal{T} >$, $\forall t \in I_i$, $(q(i), x^i(i)) = (\hat{q}(i), \hat{x}^i(i))$.

When $\mathcal{X} \sqsubseteq \hat{\mathcal{X}}$ and in addition $\mathcal{X} \neq \hat{\mathcal{X}}$, then $\mathcal{X}$ is called a strict prefix of $\hat{\mathcal{X}}$ and denoted as $\mathcal{X} \sqsubset \hat{\mathcal{X}}$.

Then, following [Lygeros et al., 2003], we are ready to introduce the definitions:

- An execution is *maximal*, if it is not a strict prefix for any other execution;
- An execution is *finite*, if $\mathcal{T}$ is finite with a closed finial interval;
- An execution is *infinite*, if $\mathcal{T}$ is either an infinite sequence or $\tau'_N = \infty$;

- An execution is *Zeno*, if it is infinite but has finite length $|\mathcal{T}| < \infty$, or, equivalently, if it takes an infinite number of discrete transitions in a finite amount of time.

It is easy to see that, under our definitions, the transition times τ_i of a Zeno execution converge to some finite accumulation point from the left. In other words, the definition of an execution precludes the situation where the transition times have a right accumulation point. Another typical Zeno execution is generated by the previous bouncing ball example [Lygeros et al., 2001], which is illustrated as follows.

Example 2.5. Following the bouncing ball example, the first bouncing occurs at

$$\tau_0' = \tau_1 = \tau_0 + \frac{x_2(\tau_0) + \sqrt{x_2^2(\tau_0) + 2gx_1(\tau_0)}}{g},$$

and the second bouncing time is at

$$\tau_2 = \tau_1' = \tau_0 + \tau_1 + \frac{2x_2(\tau_1)}{g},$$

and so on.

More generally, the N-th bouncing time can be calculated as

$$\tau_N = \tau_{N-1}' = \tau_0 + \tau_1 + \frac{2x_2(\tau_1)}{g}\sum_{k=1}^{N} c^{k-1},$$

where $x_2(\tau_1) = -cx_2(\tau_0') = c\sqrt{x_2^2(\tau_0) + 2gx_1(\tau_0)}$. Since for $c \in (0, 1)$

$$\lim_{N\to\infty}\sum_{k=1}^{N} c^{k-1} = \frac{1}{1-c},$$

we have that

$$|\tau| = \tau_0 + \frac{x_2(\tau_0)}{g} + \frac{(1+c)\sqrt{x_2^2(\tau_0) + 2gx_1(\tau_0)}}{g(1-c)}.$$

is finite, so this is a Zeno execution. □

2.2.2 Determinism and Nonblocking Properties

Before the formal definition of non-blocking and deterministic hybrid automaton, the following notations from [Lygeros et al., 2003] are introduced.

- Let $\mathcal{E}_H(q_0, x_0)$ represent all executions of a hybrid automaton H with initial condition $(q_0, x_0) \in Init$;
- Let $\mathcal{E}_H^M(q_0, x_0) \subseteq \mathcal{E}_H(q_0, x_0)$ denote all maximal executions of H with initial condition $(q_0, x_0) \in Init$;
- Let $\mathcal{E}_H^*(q_0, x_0) \subseteq \mathcal{E}_H(q_0, x_0)$ denote all finite executions of H with initial condition $(q_0, x_0) \in Init$;
- Let $\mathcal{E}_H^\infty(q_0, x_0) \subseteq \mathcal{E}_H(q_0, x_0)$ denote all infinite executions of H with initial condition $(q_0, x_0) \in Init$.

Using these notations, we can defined the following properties for hybrid automata.

Definition 2.7. [Lygeros et al., 2003] A hybrid automaton H is *non-blocking*, if $\mathcal{E}_H^\infty(q_0, x_0) \neq \emptyset$ for all initial conditions $(q_0, x_0) \in Init$.

A hybrid automaton H is *deterministic*, if $\mathcal{E}_H^M(q_0, x_0)$ contains at most one solution for all initial conditions $(q_0, x_0) \in Init$.

To check the non-blocking and deterministic properties of a hybrid automaton, we need to define the set of states reachable by a hybrid automaton H as

$$Reach_H = \left\{(\hat{q}, \hat{x}) : \exists \mathcal{X} \in \mathcal{E}_H^*(q_0, x_0) \; s.t. \; (q(N), x^N(\tau_N)) = (\hat{q}, \hat{x})\right\},$$

and the set of states from which continuous evolution is impossible as

$$Out_H = \{(q, x) \in Q \times X : \forall \epsilon > 0, \exists t \in [0, \epsilon), \Phi(t, q, x) \notin Inv(q)\},$$

where $\Phi(t, q, x)$ is a solution trajectory following the dynamics of $\dot{x} = f(q, x)$.

In general, the exact computation of $Reach_H$ and Out_H may be very complicated, but they may be calculated for some simple hybrid automata. To illustrate the calculation of $Reach_H$ and Out_H, let's revisit the bouncing ball example [Lygeros et al., 2001].

Example 2.6. First, let's calculate $Reach_H$ and Out_H for the hybrid automaton model for a bouncing ball. Note that $Init \subseteq Reach_H$, so we have $\{q_1\} \times \{x \in \mathbb{R}^2 \,|x_1 \geq 0,\ x_2 = 0\} \subseteq Reach_H$. On the other hand, we note that the position of ball cannot be negative, so $Reach_H \subseteq \{q_1\} \times \{x \in \mathbb{R}^2 \,|x_1 \geq 0\}$. By checking the state trajectories, all states in the set $\{q_1\} \times \{x \in \mathbb{R}^2 \,|x_1 \geq 0\}$ are actually reachable (with proper initial conditions). So $Reach_H = \{q_1\} \times \{x \in \mathbb{R}^2 \,|x_1 \geq 0\}$.

Next, we calculate the set Out_H. First note that the continuous states outside of $Inv(q)$ cannot be reached under mode q, i.e., $Out_H \subseteq \bigcup_{q\in Q}(\{q\} \times Inv(q)^C)$, so $\{q_1\} \times \{x \in \mathbb{R}^2 \,|x_1 < 0\} \subseteq Out_H$. On the other hand, $Out_H \subseteq Reach_H^C$, so $Out_H \subseteq \{q_1\} \times \{x \in \mathbb{R}^2 \,|x_1 > 0\}$.

Hence, we only need to check whether the states on the boundary, i.e., $\{q_1\} \times \{x \in \mathbb{R}^2 \,|x_1 = 0\}$, have continuous extensions under $\dot{x} = f(q_1, x)$ or not. For such a purpose, we further divide the boundary into two pieces, i.e., $B_1 = \{q_1\} \times \{x \in \mathbb{R}^2 \,|x_1 = 0,\ x_2 \geq 0\}$ and $B_2 = \{q_1\} \times \{x \in \mathbb{R}^2 \,|x_1 = 0,\ x_2 < 0\}$. It is easy to see that for states in B_1, there exists a positive $\epsilon > 0$ such that for all $0 < t < \epsilon$, the continuous state flow $\Phi(t, q_1, x)$ following the dynamics $\dot{x}_1 = x_2$, $\dot{x}_2 = -g$, is contained in $Inv(q_1) = \{x_1 \geq 0\}$. However, for states in B_2, for all $\epsilon > 0$, there exists $t \in [0, \epsilon)$ such that $\Phi(t, q, x) \notin Inv(q_1)$. Hence, $B_1 \subseteq Out_H$ by definition.

In conclusion, we have $Out_H = \{q_1\} \times (\{x \in \mathbb{R}^2 \,|x_1 < 0\} \cup \{x \in \mathbb{R}^2 \,|x_1 = 0,\ x_2 < 0\})$. □

As another example, let's calculate the $Reach_H$ and Out_H for the temperature control example.

Example 2.7. For $Reach_H$, firstly, all initial states are reachable, so

$$Init = \{OFF\} \times \{x \leq 60\} \subseteq Reach_H.$$

Then from the state trajectory, we get $Reach_H = (\{OFF\} \times \{x \leq 60\}) \cup (\{ON\} \times \{x \leq 69\}) \cup (\{ON, OFF\} \times \{69 \leq x \leq 71\})$.

For Out_H, note that $\bigcup_{q\in Q}(q \times Inv^c(q)) \subseteq Out_H$ so $(\{OFF\} \times \{x < 69\}) \cup (\{ON\} \times \{x > 71\}) \subseteq Out_H$. On the other hand, since $\{OFF\} \times \{x > 69\}$ and $\{ON\} \times \{x < 71\}$ are feasible, so $Out_H \subseteq (\{OFF\} \times \{x \leq 69\}) \cup (\{ON\} \times \{x \geq 71\})$. After checking the boundary $\{OFF\} \times \{x =$

$69\}$ and $\{ON\} \times \{x = 71\}$, we get $Out_H = (\{OFF\} \times \{x \leq 69\}) \cup (\{ON\} \times \{x \geq 71\})$. □

Using the notations of $Reach_H$ and Out_H for a hybrid automaton H, we can give a sufficient condition to determine whether the hybrid automaton H is non-blocking.

Lemma 2.3. [Lygeros et al., 2003] A hybrid automaton H is non-blocking, if for all states $(q, x) \in Reach_H \cap Out_H$, $\exists (q, q') \in E$ such that $x \in G(q, q')$.

The lemma is very intuitive as it basically requests that if a continuous evolution is impossible to continue further then a discrete transition must be available. The conditions of the non-blocking result are tight, but not necessary unless the automaton is deterministic. If the conditions are violated, then there exists an execution that blocks. However, unless the automaton is deterministic, starting from the same initial state a non-blocking execution may also exist.

Example 2.8. Let's check the conditions in Lemma 2.3 to determine whether the hybrid automaton model for bouncing ball is non-blocking. To do that, we first calculate the intersection of $Reach_H$ and Out_H, which is

$$Reach_H \cap Out_H = \{q_1\} \times \{x \in \mathbb{R}^2 \, | x_1 = 0, \; x_2 < 0\}.$$

Since $Reach_H \cap Out_H \subseteq \{q_1\} \times G(q_1, q_1)$ so the conditions in Lemma 2.3 hold. Hence, the hybrid automaton model for a bouncing ball is non-blocking. □

Next, we check the hybrid automaton model for the temperature control example.

Example 2.9. For the hybrid automaton model of the thermostat example, the intersection of $Reach_H$ and Out_H is given as $Reach_H \cap Out_H = (\{OFF\} \times \{x \leq 60\}) \cup (\{OFF\} \times \{x = 69\}) \cup (\{ON\} \times \{x = 71\})$. Note that $\{OFF\} \times (\{x \leq 60\} \cup \{x = 69\}) \subset \{OFF\} \times G(OFF, ON)$, where $G(OFF, ON) = \{x \leq 70\}$. Also, $\{ON\} \times \{x = 71\} \subseteq \{ON\} \times G(ON, OFF)$, where $G(ON, OFF) = \{x \geq 70\}$. Hence, the conditions in Lemma 2.3 hold, so we conclude that the hybrid automaton model of the thermostat example H is non-blocking. □

Intuitively, a hybrid automaton may be non-deterministic if either there is a choice between continuous evolution and a discrete transition, or if a discrete transition can lead to multiple destinations (under assumption, continuous evolution is unique). The following result provides a formal statement of this fact.

Lemma 2.4. [Lygeros et al., 2003] A hybrid automaton H is deterministic, if and only if for state $(q, x) \in Reach_H$,

1. if $x \in G(q, q')$ for some $(q, q') \in E$, then $(q, x) \in Out_H$;
2. if $(q, q') \in E$, $(q, q'') \in E$ and $q' \neq q''$, then $x \notin G(q, q') \cap G(q, q'')$;
3. if $(q, q') \in E$ and $x \in G(q, q')$ then $R((q, q'), x)$ contains at most one element.

To be more specific, the first condition indicates that once a discrete transition becomes valid then it must be fired; The second condition excludes non-deterministic choices in discrete transitions, while the third condition on the reset map requests that the states are uniquely updated after a discrete transition. The bouncing ball example is used to illustrate Lemma 2.4.

Example 2.10. To decide whether the hybrid automaton model for bouncing ball is deterministic, we need to check the conditions in Lemma 2.4.

The first condition, "if $x \in G(q, q')$ for some $(q, q') \in E$, then $(q, x) \in Out_H$", holds true since there is only one transition (q_1, q_1), and $(\{q_1\} \times G(q_1, q_1)) \subseteq Out_H$ as $G(q_1, q_1) = \{x \in \mathbb{R}^2 \,|x_1 = 0,\ x_2 < 0\}$ while $Out_H = \{q_1\} \times (\{x \in \mathbb{R}^2 \,|x_1 < 0\} \cup \{x \in \mathbb{R}^2 \,|x_1 = 0,\ x_2 < 0\})$.

The second condition trivially holds since there is only one transition (q_1, q_1) in H.

The third condition also holds true since the possible state set under the reset map $R((q_1, q_1), x) = \begin{bmatrix} x_1 \\ -cx_2 \end{bmatrix}$ is a singleton.

In view of these, we can verify that the hybrid automaton model for bouncing ball is deterministic. □

It will be show that the hybrid automaton for the temperature control example is non-deterministic.

Example 2.11. Consider the state $\{OFF\} \times \{x = 69.5\} \in Reach_H$, we can verify that the first condition is not satisfied. So the hybrid automaton model in the thermostat example H is non-deterministic. The basic reason for nondeterminacy comes from the fact that the furnace can be turned off at any point if the temperature is above 70 but below 71. In other words, any transition that occurs in such a range generates a valid hybrid trajectory. So, there are infinitely many trajectories from any initial state. So, the hybrid automaton H is not deterministic. □

Combine the above two lemmas, a hybrid automaton H is well-posed (its solution exists and is unique for a given initial condition), if and only if the conditions in Lemma 2.3 and Lemma 2.4 hold. Interested readers may refer to [Lygeros et al., 2003] for proofs. Put together, we can conclude that the hybrid automaton model for bouncing ball is well-posed.

Hybrid automata are rich in expressiveness and are therefore quite suitable for modeling and simulating hybrid dynamical systems. However, due to heterogeneous discrete and continuous parts, they are not directly suitable for solving analysis and synthesis problems. In the following sections, we will introduce different kinds of hybrid models from the literatures that are more suitable for computations.

2.3 Switched Systems

A switched system is a dynamical system that consists of a finite number of continuous-variable subsystems and a logical rule that orchestrates switching between these subsystems. Mathematically, these subsystems are usually described by a collection of indexed differential or difference equations.

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

with initial condition $\{q(t_0), x(t_0)\} \in Init$, where $x(t) \in \mathbb{R}^n$ is the continuous state vector, and $q(t) \in Q = \{q_1, q_2, \cdots, q_N\}$ stands for the collection of discrete modes.

The logical rule that orchestrates switching between these subsystems generates switching signals, which are usually described as

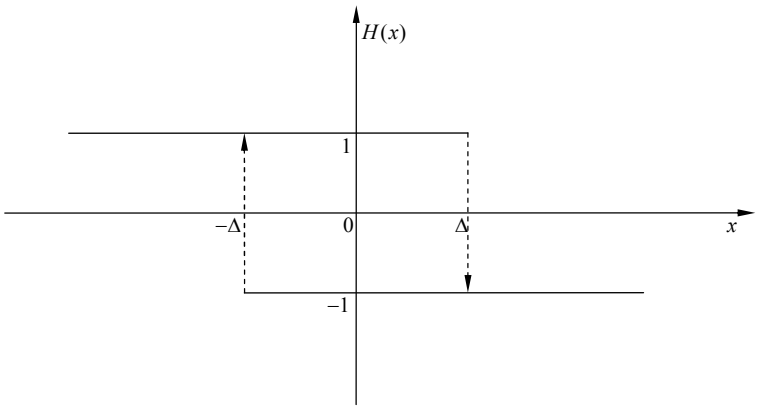


Figure 2.11: Hysteresis function.

classes of piecewise constant maps, $\sigma : \mathbb{R} \to Q$. By piecewise constant, we mean that the switching signal $\sigma(t)$ has finite number of discontinuities on any finite interval of $\mathbb{R}$.

Example 2.12. A dynamical system with hysteresis exhibits lag effects as its parameters and evolution depend not only on its current environment but also on its past history. Hysteresis occurs in many industrial, economic and bio-molecular systems. A simple dynamical systems with hysteresis can be represented by a differential equation, $\dot{x} = H(x)$, with a discontinuous $H(x)$ as shown in Figure 2.11.

Dynamical systems with hysteresis can be modeled by switched systems with two discrete modes $Q = \{q_1, q_2\}$, and

$$f(x, q_1) = 1, f(x, q_2) = -1,$$

$$\delta(x, q_1) = \begin{cases} q_1 & x \leq \Delta \\ q_2 & x \geq \Delta \end{cases}$$

$$\delta(x, q_2) = \begin{cases} q_1 & x \leq -\Delta \\ q_2 & x \geq -\Delta \end{cases}$$

with the initial condition $Init = \{q_1, q_2\} \times \mathbb{R}$. □

Properties of switched systems have been studied for the past sixty years to consider engineering systems that contain relays and/or hysteresis. Recently, there has been increasing interest in the stability analysis and switching control design of switched systems, see e.g., [Liberzon, 2003]. The primary motivation for studying such switched systems comes partly from the fact that switched systems and switched multi-controller systems have numerous applications in the control of mechanical systems, process control, automotive industry, power systems, aircraft and traffic control, and many other fields. In addition, there exists a class of nonlinear systems which can be stabilized by switching control schemes, but cannot be stabilized by any continuous static state feedback control law [Brockett, 1983].

Next, we explore the relationship between the hybrid automaton and switched system modeling. A switched system can be modeled as a hybrid automaton (See Section 2). Specifically,

- $Q = \{q_1, q_2, \cdots, q_N\}$ is the same;
- $X = \mathbb{R}^n$;
- f is the same;
- $Init \subseteq Q \times X$ is the same;
- Inv : for all $q \in Q$, $Inv(q) = \{X \in \mathbb{R}^n | \, q = \delta(x, q)\}$, i.e., all modes of dynamics are feasible on the whole state space;
- $E : (q, q') \in E$ when $q \neq q'$ and there exists $x \in X$ such that $q' = \delta(x, q)$;
- G : for $(q, q') \in E$, $G(q, q') = \{x \in \mathbb{R}^n | \, q' = \delta(x, q)\}$;
- R is the identity map, i.e., no state jumps.

For illustration, let's revisit the hysteresis example.

Example 2.13. A system with hysteresis can be modeled as a hybrid automaton. In particular, we plot its graphical representation of the hybrid automata model in Figure 2.12. □

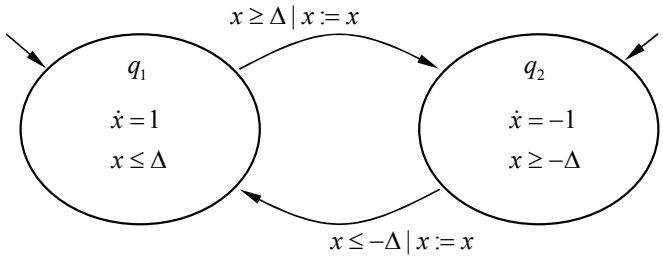


Figure 2.12: A hybrid automata model for a dynamical system with hysteresis.

As shown above, any switched system can be modeled as a hybrid automaton. On the other hand, any hybrid automaton without state jumps (i.e., the reset mapping R is identity for any discrete transitions) can be written as a switched system with the same Q, X, f, $Init$ and

$$\delta(x, q) = \begin{cases} q & x \in Inv(q) \\ q' & x \in G(q, q') \end{cases}$$

as illustrated in the following example.

Example 2.14. The temperature control example can be written as a switched system with

$$f(x, OFF) = -x, f(x, ON) = -x + 100$$

$$\delta(x, OFF) = \begin{cases} OFF & x \geq 69 \\ ON & x \leq 70 \end{cases}$$

$$\delta(x, ON) = \begin{cases} OFF & x \geq 70 \\ ON & x \leq 71 \end{cases}$$

□

2.4 Piecewise Affine Systems

Piecewise affine systems represent switching dynamics among a collection of linear differential or difference equations with state space being partitioned by a finite number of linear hyperplanes, see e.g.,

[Sontag, 1981, Johansson, 2003b]. Mathematically, a piecewise affine system can be represented by

$$\dot{x}(t) = A_q x(t) + b_q, \text{ for } x \in \Omega_q, \tag{2.1}$$

where $x(t) \in \mathbb{R}^n$ is the continuous-time state, $q \in Q$ is a finite index and $\Omega_q \subseteq \mathbb{R}^n$. The sets Ω_q are assumed to provide a subdivision of the state space $\mathbb{R}^n$, that is $\bigcup_{q\in Q} \Omega_q = \mathbb{R}^n$ and Ω_q for $q \in Q$ are polyhedral and disjoint sets (only share common boundaries). Piecewise affine systems arise very often as mathematical models in practical applications. For example, piecewise affine systems can be used to model systems with discontinuous dynamics that arise because of saturation constraints, friction in mechanical systems and so on.

Example 2.15. [Rantzer and Johansson, 2000] Consider a saturated linear system:

$$\dot{x} = Ax + b \cdot sat(v),\ v = k^T x,$$

where the saturation function $sat(\cdot)$ is defined as

$$sat(v) = \begin{cases} -1, & v \le -1 \\ v, & -1 < v \le 1 \\ 1, & v > 1 \end{cases},$$

Graphically, the system can be represented by the block diagram in Figure 2.13.

To depict the system as a piecewise affine system, we first divide the state space into three disjoint regions:

- negative saturation, $\Omega_1 = \{x \in \mathbb{R}^n | k^T x \le -1\}$;
- linear operation, $\Omega_2 = \{x \in \mathbb{R}^n | -1 \le k^T x \le 1\}$; and
- positive saturation $\Omega_3 = \{x \in \mathbb{R}^n | k^T x \ge 1\}$.

It is clear that the three regions Ω_i are all polyhedral (i.e., can be described by a set of linear inequalities), for $i = 1, 2, 3$. Then, the saturated system can be modeled as the following piecewise affine system

$$\dot{x} = \begin{cases} Ax - b, & x \in \Omega_1 \\ (A + bk^T)x, & x \in \Omega_2 \\ Ax + b, & x \in \Omega_3 \end{cases}$$

□

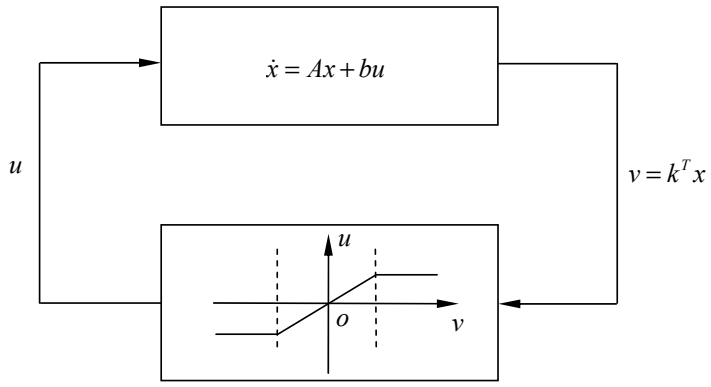


Figure 2.13: A feedback control system with saturation.

$\bar{\Omega}_q$ stands for the closure of the set Ω_q

Piecewise affine systems can be seen as a special class of hybrid automata, as any piecewise affine system can be written as a hybrid automaton where

- Q is the finite index set;
- $X = \mathbb{R}^n$;
- f : for all $q \in Q$, $f(q, x) = A_q x(t) + b_q$;
- $Inv(q) : \forall q \in Q$, $Inv(q) = \Omega_q$;
- $Init = \bigcup_{q \in Q}(\{q\} \times Inv(q))$;
- $E : (q, q') \in E$ when $q \neq q'$ if two partitions Ω_q and $\Omega_{q'}$ are adjacent, i.e., $\bar{\Omega}_q \cap \bar{\Omega}_{q'} \neq \emptyset$, where $\bar{\Omega}_q$ stands for the closure of the set Ω_q;
- G : for any pair $(q, q') \in E$, $G(q, q') = \bar{\Omega}_{q'}$;
- R : for any pair $(q, q') \in E$, its reset map is the identity map, i.e., $R(\cdot, x) = \{x\}$.

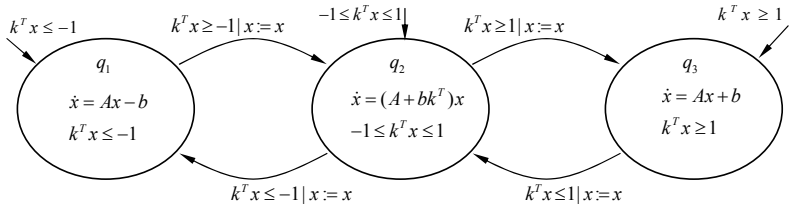


Figure 2.14: A hybrid automaton model for the feedback control system with saturation.

For illustration, let's revisit the saturated control system example.

Example 2.16. The saturated system example can be written in the form of hybrid automata as shown in Figure 2.14. □

Furthermore, a piecewise affine system can be written as a switched system with

$$\dot{x}(t) = f(x, q) = A_q x(t) + b_q$$

$$\delta(x, q) = \begin{cases} q & x \in \Omega_q \\ q' & x \in \Omega_{q'} \end{cases}$$

As an example, we revisit the saturated control system example.

Example 2.17. To write the saturated system example as a switched system, $\delta(x, q_2)$ is given by

$$\delta(x, q_2) = \begin{cases} q_1 & x \in \Omega_1 \\ q_2 & x \in \Omega_2 \\ q_3 & x \in \Omega_3 \end{cases}$$

$\delta(x, q_1)$ and $\delta(x, q_3)$ can be obtained in the same way. □

A typical trajectory of a piecewise affine systems is illustrated in Figure 2.15. We can observe that the trajectory consists of patched pieces concatenated together. A concatenating point stands for the occurrence of a switching, which is caused by the evolution of the continuous state $x(t)$ intersecting with the boundary of Ω_q, e.g., the temperature being above some value.

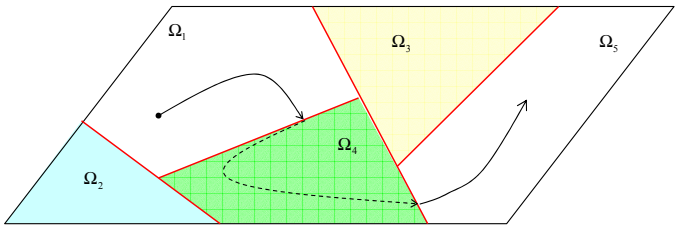


Figure 2.15: A typical trajectory of piecewise affine systems.

We are interested in the "well-posedness" of a piecewise affine system. In particular, a piecewise affine system is said to be well-posed if for any initial state $x(t_0) = x_0$, there exists an $\varepsilon > 0$ such that there is a unique solution $x(t)$ satisfying $\dot{x}(t) = A_q x(t) + b_q$ with $x(t_0) = x_0$ and $x(t) \in \Omega_q$ for all $t \in [t_0, t_0 + \varepsilon)$. In other words, well-posedness refers to local existence and uniqueness of solutions for the piecewise affine system starting from a given initial condition. If local uniqueness holds for all initial conditions and existence holds globally, then uniqueness must also hold globally since there is no point at which solutions can split.

The following example from [Imura and van der Schaft, 2000] shows that well-posedness is nontrivial for a piecewise affine system. Consider a planar piecewise affine system

$$\dot{x} = \begin{cases} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ -1 \end{bmatrix}, & \text{if } y = \begin{bmatrix} 0 & 1 \end{bmatrix} x \leq 0 \\ \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix}, & \text{if } y = \begin{bmatrix} 0 & 1 \end{bmatrix} x \geq 0 \end{cases}$$

where $x \in \mathbb{R}^2$ is the state.

Note that from any initial state $x(0) = (c, 0)^T$ with $|c| \leq 1$, there are two possible solutions. First, let's check

$$\begin{cases} x_1(t) = -1 + (c+1)\cos t \\ x_2(t) = -(c+1)\sin t \end{cases}$$

which satisfies the equation $\dot{x}_1 = x_2$, $\dot{x}_2 = -x_1 - 1$, and $\exists \varepsilon > 0$ such

that $x_2(t) \leq 0$ for $0 \leq t < \varepsilon$. So it is a solution of the above piecewise affine system.

On the other hand, starting from the same initial condition, the following function

$$\begin{cases} x_1(t) = 1 + (c-1)\cos t \\ x_2(t) = -(c-1)\sin t \end{cases}$$

satisfies the equation $\dot{x}_1 = x_2$, $\dot{x}_2 = 1 - x_1$, and $\exists \varepsilon > 0$ such that $x_2(t) \geq 0$ for $0 \leq t < \varepsilon$. So it is also a solution of the above piecewise affine system.

In summary, the above piecewise affine system is not well-posed as a dynamical system. Also, there are examples showing that the solutions of piecewise affine systems are not continuous with respect to the initial conditions [Imura and van der Schaft, 2000]. Namely, two trajectories starting from two arbitrarily close points could evolve far away from each other. Since piecewise affine systems are a special case of switched systems and hybrid automata, so the well-posedness problem for general hybrid systems is really non-trivial and remains open. To gain more insights, researchers have focused on some special classes of piecewise affine system, such as bimodal systems and conic systems, and obtained some interesting results, see e.g., [Imura and van der Schaft, 2000] and the references therein.

2.5 Notes and Further Reading

The results presented on finite automata and the formal language theory mainly follow [Hopcroft et al., 2006], where interested readers may find the proofs of all claims and more examples. Readers may refer to the book [Cassandras and Lafortune, 2008] for a comprehensive discussion on discrete event systems.

Hybrid automata models were first proposed for verification in the 1990s, see e.g., [Alur et al., 1993, 1995]; interested readers may refer to the survey [Henzinger, 2000] and its references therein. Our discussions on hybrid automata modeling and its determinism and non-blocking properties are based on [Lygeros et al., 2003]. There

are excellent lecture notes available on hybrid automata, see e.g., [Lygeros et al., 2001].

It is possible to further generalize the hybrid automata model by introducing inputs and outputs, see e.g., hybrid I/O automata [Lynch et al., 2003]. In particular, continuous control inputs U can be introduced in the continuous dynamics for each mode q and discrete events Σ can be added to trigger discrete transitions $(q_i, q_j) \in E$. It is also possible to include inputs to control the reset value of the reset map and consider set valued mappings and differential or difference inclusions, see e.g., [Aubin et al., 2002, Goebel et al., 2012]. The extension of hybrid automata to handle probabilistic uncertainties gives the stochastic hybrid automata model [Hu et al., 2000, Pola et al., 2003, Cassandras and Lygeros, 2010], where the continuous variable dynamics inside each invariant set of the discrete modes are described as stochastic differential equations and the discrete transitions also occur randomly.

A complicated hybrid automaton model can be constructed by the composition of several hybrid modules. Interested readers may refer to [Alur and Henzinger, 1997, Lynch et al., 2003] for modularity and composition of hybrid automata. There exist a number of software packages in support of hybrid automata modeling and simulation, such as SHIFT [Deshpande et al., 1997], PTOLEMY [Liu et al., 1999], CHARON [Alur et al., 2000b]. There also exist commercial modeling tools such as STATEFLOW (see `www.mathworks.com`) and MODELICA (see `www.modelica.org`) that have been successfully used in industry.

Switched systems and piecewise linear systems are widely used to model dynamical systems that exhibit mode switching due to either external inputs or environmental change, e.g., the slippage of a legged robot, the gear switch of a car. Piecewise affine systems are also called piecewise linear systems, and have been widely used in the study of circuits, see e.g., [Leenaerts and Van Bokhoven, 1998], since they can approximate nonlinear dynamics with arbitrary accuracy [Lin and Unbehauen, 1992]. There also exist efficient computational techniques for the identification of piecewise affine

models from input-output data, such as clustering based methods [Ferrari-Trecate et al., 2003], mixed-integer programming [Roll et al., 2004], and Bayesian methods [Juloski et al., 2005].

Besides switched systems and piecewise affine systems, there are other modeling frameworks arising from the control and mathematics literature attempting to capture the hybrid nature of a dynamical system, such as autonomous or controlled switching and state jumps. Early efforts include impulsive differential inclusions [Aubin et al., 2002], complementarity systems [Schaft and Schumacher, 2000], mixed logic dynamical systems [Bemporad and Morari, 1999]. For the discrete time case, it is shown that piecewise affine systems, mixed logic dynamical systems, linear complementarity systems, and min-max-plus-scaling systems are actually equivalent [Heemels et al., 2001], in the sense that one can be converted to another. A software toolbox HYSDEL has been developed to automate the translation process between these modeling frameworks, see e.g., [Torrisi and Bemporad, 2004].

3

Stability, Stabilization and Optimal Control

This chapter reviews the basic results for stability, stabilization and optimal control of hybrid systems. Most of the results in the literature were developed for either switched systems, piecewise affine systems or their special cases. An attempt is made here to state the main results in the language of hybrid automata so to be consistent with other sections. On the other hand, since we do not explicitly consider state jumps in this chapter, we use the term switched systems and hybrid systems interchangeably here, and use switching signals to denote the sequence of discrete mode transitions.

The rest of this chapter is organized as follows. First, we focus on the stability analysis of hybrid systems under given discrete switching logics in Section 3.1. In particular, some results on the stability analysis for hybrid systems under arbitrary switching are introduced first, then the stability under slow switching (like dwell time and average dwell time) is studied. The general case of hybrid system stability under restricted switching is investigated through multiple Lyapunov functions. Then, we turn to the synthesis of stabilizing switching logic for a given collection of continuous variable dynamical systems in Section 3.2, where several stabilization conditions and design meth-

ods are described. A closely related problem to switching stabilization is the optimal control problem for hybrid systems. Optimal control for hybrid systems has a rich literature. Early efforts may be found in studies of control systems involving relays or dynamical systems with hysteresis [Witsenhausen, 1966]. Significant efforts have been devoted to the extensions of the maximum principle and dynamical programming techniques to hybrid systems, see e.g., [Branicky et al., 1998, Sussmann, 1999, Hedlund and Rantzer, 2002]. Here we choose to focus on some recent developments in computational approaches to the optimal control synthesis for switched systems, in particular, the embedding optimization and two-stage optimization approaches which are reviewed in Section 3.3. Optimal control in piecewise affine system models is also discussed in Section 3.3, where the piecewise affine system model is transformed into a mixed logic dynamic system and the optimal controller synthesis can be solved as a mixed integer optimization problem.

3.1 Stability of Hybrid Systems

Stability is a basic requirement in control practice. Generally speaking, a dynamical system is stable when there exists at least one operating point (typically an equilibrium where the state stays if there is no external stimulus) such that all the state trajectories starting from somewhere near it will stay close to the operating point all the time, and may eventually converge to that point. This is known as Lyapunov stability, and some control problems, like trajectory tracking, can be transformed into stability problems, see e.g., [Khalil and Grizzle, 2002, Sontag, 1998].

3.1.1 Motivating Examples

Interestingly, the stability issues of hybrid systems include several remarkable phenomena. Consider the following motivating examples, which are adopted from [Branicky, 1998, DeCarlo et al., 2000].

Example 3.1. Consider a hybrid system described by a hybrid automaton (see Section 2.2)

- $Q = \{q_1, q_2\}$;
- $X = \mathbb{R}^2$;
- $f(q_1, x) = A_1 x$ and $f(q_2, x) = A_2 x$, where

$$A_1 = \begin{bmatrix} -1 & -100 \\ 10 & -1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} -1 & 10 \\ -100 & -1 \end{bmatrix}; \tag{3.1}$$

- $Init = Q \times X$;
- $Inv(q_1) = X$, and $Inv(q_2) = X$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbb{R}^2 : x_2 = -0.2x_1\}$,
  and $G(q_2, q_1) = \{x \in \mathbb{R}^2 : x_2 = 5x_1\}$;
- $R((q_1, q_2), x) = \{x\}$, and $R((q_2, q_1), x) = \{x\}$.

Here the hybrid system is switching between the two linear time invariant (LTI) subsystems

$$\dot{x} = A_1 x, \;\; \dot{x} = A_2 x.$$

Notice that both A_1 and A_2 are exponentially stable since all eigenvalues of A_1 and A_2 have negative real parts. In addition, they have a common equilibrium point, which is the origin. However, this simple hybrid system is unstable, since a divergent trajectory can be generated for an initial state even very close to the equilibrium point. See Figure 3.1 for illustrations (with initial condition $x_0 = \begin{bmatrix} -0.01 \\ 0.02 \end{bmatrix}$). □

This motivating example shows that even when all the subsystems are exponentially stable, the hybrid systems may be unstable. On the other hand, switching between unstable subsystems may generate convergent trajectories as the following example shows.

Example 3.2. Consider again a hybrid system switching between two LTI subsystems that can be described by a hybrid automaton as below

- $Q = \{q_1, q_2\}$;
- $X = \mathbb{R}^2$;

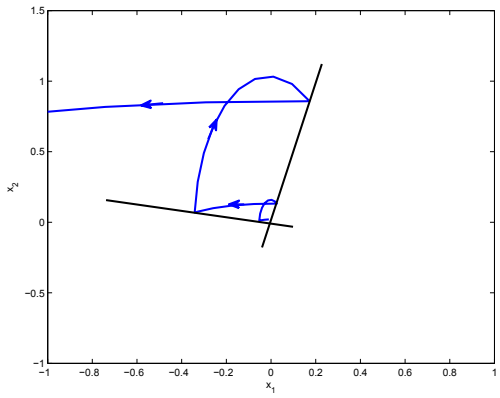


Figure 3.1: Divergent trajectories can be generated by switching between two stable subsystems. The guard sets (also known as switching surfaces) are plotted for illustration.

- $f(q_1, x) = A_1 x$ and $f(q_2, x) = A_2 x$, where
$$A_1 = \begin{bmatrix} 1 & -100 \\ 10 & 1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1 & 10 \\ -100 & 1 \end{bmatrix}; \tag{3.2}$$
- $Init = Q \times X$;
- $Inv(q_1) = X$, and $Inv(q_2) = X$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbb{R}^2 : x_2 = -x_1\}$,
and $G(q_2, q_1) = \{x \in \mathbb{R}^2 : x_2 = 0\}$;
- $R((q_1, q_2), x) = \{x\}$, and $R((q_2, q_1), x) = \{x\}$.

Notice that both A_1 and A_2 are unstable since both have positive real part eigenvalues. However, the overall hybrid system is asymptotically stable. This is illustrated in Figure 3.2 as the state trajectory (with initial condition $x_0 = \begin{bmatrix} 0.4 \\ -0.2 \end{bmatrix}$) approaches the origin as time goes to infinity. □

As these examples suggest, the stability of hybrid systems depends not only on the dynamics of each subsystem but also the properties of

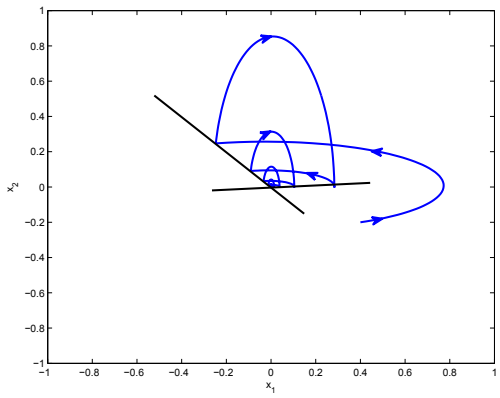


Figure 3.2: Switching between unstable subsystems may generate stable (convergent) closed-loop behaviors.

the logic part, here the switching signals. Therefore, the stability study of hybrid systems can be roughly divided into two kinds of problems. One is the stability analysis of hybrid systems under given switching signals (maybe arbitrary, slow switching etc.); the other is the synthesis of stabilizing switching signals for a given collection of dynamical systems. This section focuses on the stability analysis for switched (hybrid) systems under given switching signals; while the switching stabilization problem will be studied in the next section.

3.1.2 Arbitrary Switching

First, we consider stability analysis problems when there is no restrictions on the discrete event dynamics in the hybrid systems. This may be due to no available *a prior* knowledge about the discrete event logic, partitions of the state space, constraints etc. Under this circumstance, one usually tends to be conservative and assume that all discrete transitions are possible; this is usually called arbitrary switching in the literature. As it is shown in the first motivating example, a hybrid system may become unstable even when all subsystems are exponentially stable. Therefore, to identify conditions for which hybrid systems are

stable under arbitrary switching signals is nontrivial and interesting.

For this problem, it is necessary to require that all the subsystems are asymptotically stable, since one may always stay at a certain unstable subsystem all the time, which is a valid 'switching signal'. However, in general, the above subsystems' stability assumption is not sufficient to assure stability for the hybrid systems under arbitrary switching signals. On the other hand, if there exists a common Lyapunov function for all the subsystems, i.e., a continuously differentiable, radially unbounded, positive definite function $V : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$, for which the derivative $\dot{V}(x,t)$ is negative definite along all subsystems' trajectories, then the stability of the hybrid system is guaranteed under arbitrary switchings. This provides us with a possible way to solve this problem, and many efforts have been focused on common quadratic Lyapunov functions.

First, we consider a collection of continuous-time LTI systems

$$\dot{x}(t) = A_i x(t), \;\; t \in \mathbb{R}^+, \;\; i \in \mathcal{I} \tag{3.3}$$

where $\mathcal{I}$ stands for a finite index set. For all $i \in \mathcal{I}$, the state matrices $A_i \in \mathbb{R}^{n\times n}$. Note that the origin $x_e = 0$ is a common equilibrium for the systems described in (3.3). The hybrid system of interest is built by allowing arbitrary switching among these LTI systems (3.3). Since no restrictions are imposed on the switching signals, we will not consider particular discrete-event dynamics in the hybrid system. Under this situation, we call (3.3) an arbitrary switching (linear) system or a switched linear system under arbitrary switching.

A Common Quadratic Lyapunov Function (CQLF) for (3.3) has the form

$$V(x) = x^T P x$$

where $P = P^T$ (symmetric) and $P > 0$ (positive definite). In addition, its time derivative along any trajectory of systems (3.3) is negative definite, or alternatively

$$A_i^T P + P A_i = -Q_i, \quad i \in \mathcal{I}$$

where, Q_i are symmetric and positive definite for all $i \in \mathcal{I}$. The existence of a CQLF for all its subsystems assures the quadratic stability of

the hybrid system. Quadratic stability is a special class of exponential stability, which implies asymptotic stability, and has attracted a lot of research efforts due to its importance in practice.

Actually, the determination of a CQLF for (3.3) can be obtained by solving the following linear matrix inequalities (LMIs) [Boyd et al., 1994]. Namely, there exists a positive definite symmetric matrix P, $P \in \mathbb{R}^{n\times n}$, such that

$$PA_i + A_i^T P < 0, \ \ \forall i \in \mathcal{I}, \tag{3.4}$$

hold simultaneously. However, the standard interior point methods for LMIs may become ineffective as the number of modes increases. This motivates us to identify easily verifiable conditions that guarantee the existence of a CQLF for (3.3).

Let us first look at a special case, where the subsystems' state matrices are pairwise commutative, i.e., $A_iA_j = A_jA_i$ for all $i, j \in \mathcal{I}$. Because of the commutativity, it is easy to derive that

$$A_i^{k_1} A_j^{k_2} = A_j^{k_2} A_i^{k_1},$$

for any nonnegative integer k_1 and k_2, and

$$e^{A_i t_1} e^{A_j t_2} = e^{A_j t_2} e^{A_i t_1},$$

for any nonnegative real number t_1 and t_2. By direct computation, it is straightforward to verify that in this case the switched systems is stable if and only if all its subsystems are stable [Narendra and Balakrishnan, 1994].

Theorem 3.1. For an arbitrary switching system (3.3), if all subsystem matrices are Hurwitz stable (i.e., all eigenvalues of the state matrix A_i have negative real parts) and pairwise commutative ($A_iA_j = A_jA_i, \forall i, j \in \mathcal{I}$), then the arbitrary switching system is stable.

Actually, a CQLF exists for this case, which can be determined by solving a collection of chained Lyapunov equations [Narendra and Balakrishnan, 1994].

Theorem 3.2. Assume that the index set $\mathcal{I} = \{1, \cdots, N\}$. Let $P_1, \cdots, P_N$ be the unique symmetric positive definite matrices that

satisfy the Lyapunov equations

$$A_1^T P_1 + P_1 A_1 = -I, \tag{3.5}$$
$$A_i^T P_i + P_i A_i = -P_{i-1}, \quad i = 2, \cdots, N \tag{3.6}$$

then the function $V(x) = x^T P_N x$ is a CQLF for systems $\dot{x}(t) = A_i x(t)$, $i = 1, \cdots, N$.

In addition, the matrix P_N is given by

$$P_N = \int_0^\infty e^{A_N^T t_N} \cdots (\int_0^\infty e^{A_1^T t_1} e^{A_1 t_1} dt_1) \cdots e^{A_N t_N} dt_N$$

Since the matrices A_i commute, for each $i \in \{1, \cdots, N\}$ we can rewrite this in the form

$$P_N = \int_0^\infty e^{A_i^T t_i} Q_i e^{A_i t_i} dt_i$$

with $Q_i > 0$. This result was extended to the discrete-time case in [Zhai et al., 2002], namely:

Theorem 3.3. Let $P_1, \cdots, P_N$ be the unique symmetric positive definite matrices that satisfy the Lyapunov equations

$$A_1^T P_1 A_1 + P_1 = -I, \tag{3.7}$$
$$A_i^T P_i A_i + P_i = -P_{i-1}, \quad i = 2, \cdots, N \tag{3.8}$$

then the function $V(x) = x^T P_N x$ is a CQLF for the systems $x[k+1] = A_i x[k]$, $i = 1, \cdots, N$.

Besides the case of commutative state matrices, it has been shown that CQLF exists for arbitrary switching systems with either all upper or lower triangular state matrices. A more general condition on the existence of CQLF for (3.3) can be characterized through the Lie algebra generated by the subsystems' state matrices [Liberzon et al., 1999]. The matrix Lie algebra generated by the matrices A_i, $i \in \mathcal{I}$ (with respect to standard Lie bracket $[A_i, A_j] = A_i A_j - A_j A_i$), denoted as $\mathfrak{g} = \{A_i : i \in \mathcal{I}\}_{LA}$, is the linear space (over $\mathbb{R}$) spanned by the iterated Lie brackets of these matrices. For example, consider a Lie algebra generated by two matrices A_1 and A_2, then

$$\mathfrak{g} = \{A_1, A_2\}_{LA} = span\{A_1, A_2, [A_1, A_2], [A_1, [A_1, A_2]], \cdots\}.$$

Given a Lie algebra $\mathfrak{g}$, the sequence $\mathfrak{g}^{(k)}$ is defined inductively as:

$$\mathfrak{g}^{(1)} = \mathfrak{g}; \;\; \mathfrak{g}^{(k+1)} = [\mathfrak{g}, \mathfrak{g}^{(k)}]$$

where $[\mathfrak{g}, \mathfrak{g}^{(k)}]$ is a linear space spanned by all the products $[a, b]$ with $a \in \mathfrak{g}$ and $b \in \mathfrak{g}^{(k)}$. It is easy to show that $\mathfrak{g}^{(k+1)} \subset \mathfrak{g}^{(k)}$. A Lie algebra $\mathfrak{g}$ is called solvable if $\mathfrak{g}^{(k)} = 0$ for a finite k.

Theorem 3.4. [Liberzon et al., 1999] If all the matrices A_i, $i \in \mathcal{I}$ are Hurwitz (all eigenvalues of A_i have negative real parts) and the Lie algebra $\{A_i : i \in \mathcal{I}\}_{LA}$ is solvable then there exists a CQLF.

This is because that the matrices A_i, $i \in \mathcal{I}$ in a solvable Lie algebra can be simultaneously put in the upper-triangular form, and that a family of linear systems with stable upper-triangular matrices possess a CQLF. Interested readers may refer to [Liberzon et al., 1999, Liberzon, 2003] for more detailed and formal discussions on the Lie algebraic conditions and extensions to nonlinear switched systems can be found in [Margaliot and Liberzon, 2006]. The solvability condition for the Lie algebra implies the existence of a CQLF. There are also some interesting necessary and sufficient algebraic conditions in the literature for the existence of a CQLF, see e.g., [King and Shorten, 2004].

It is worth pointing out that the existence of a CQLF is only sufficient for the stability of arbitrary switching systems. There are switched systems that do not have a CQLF, but they are exponentially stable under arbitrary switching [Liberzon, 2003]. Therefore, in general, the existence of a CQLF is only sufficient for the asymptotic or exponential stability of hybrid systems under arbitrary switching signals. Hence a less conservative class of Lyapunov functions, called switched quadratic Lyapunov functions, was proposed in the literature, see e.g., [Daafouz et al., 2002].

Here, we investigate the stability of the following discrete-time arbitrary switching LTI systems

$$x[k+1] = A_i x[k], \;\; k \in \mathbb{Z}^+, \tag{3.9}$$

where $x \in \mathbb{R}^n$, and $i \in \mathcal{I}$. Basically, since every subsystem is asymptotically stable (with the origin as the common equilibrium point), there

exists a positive definite symmetric matrix P_i that solves the Lyapunov equation for each i-th subsystem

$$A_i^T P_i A_i - P_i < 0,$$

for all $i \in \mathcal{I}$. Next, these matrices P_i are patched together based on the switching signals $\sigma(k)$ to construct a global Lyapunov function as

$$V(k, x[k]) = x^T[k] P_{\sigma(k)} x[k], \tag{3.10}$$

where $\sigma(k) : k \to \mathcal{I}$ stands for the switching signal at step k. Since all P_i are positive definite, it is clear that the function $V(k, x[k]) = x^T[k] P_{\sigma(k)} x[k]$ is positive definite. If it further holds that $\Delta V(k, x[k]) = V(k+1, x[k+1]) - V(k, x[k])$ is negative definite along the solution of (3.9), then the origin of the system (3.9) is globally asymptotically stable as stated in the following theorem.

Theorem 3.5. [Daafouz et al., 2002] If there exist positive definite symmetric matrices $P_i \in \mathbb{R}^{n \times n}$ ($P_i = P_i^T$) for $i \in \mathcal{I}$, satisfying

$$\begin{bmatrix} P_i & A_i^T P_j \\ P_j A_i & P_j \end{bmatrix} > 0 \tag{3.11}$$

for all $i, j \in \mathcal{I}$, then the discrete-time arbitrary switching system (3.9) is asymptotically stable.

Then, stability checking for the arbitrary switching linear systems can be performed by solving certain LMIs. It is clear that when $P_i = P_j$ for all $i, j \in \mathcal{I}$, the switched quadratic Lyapunov function becomes the CQLF. Therefore, the stability criteria based on the switched quadratic Lyapunov function generalizes the CQLF approach and usually gives us less conservative results. However, it is worth pointing out that the switched quadratic Lyapunov function method is still a sufficient only condition.

In the sequel, we will provide some necessary and sufficient conditions for the asymptotic stability of arbitrary switching linear systems. It is shown that the asymptotic stability problem for switched linear systems with arbitrary switching is equivalent to the robust asymptotic stability problem for polytopic uncertain linear time-variant systems, for which several strong stability conditions exist [Molchanov and Pyatnitskiy, 1989, Molchanov and Liu, 2002].

Theorem 3.6. The following statements are equivalent:

1. The switched linear system
$$\dot{x}(t) = A_{\sigma(t)}x(t),$$
where $A_{\sigma(t)} \in \{A_1, A_2, \cdots, A_N\}$, is asymptotically stable under arbitrary switching;
2. the linear time-variant system $\dot{x}(t) = A(t)x(t)$, where $A(t) \in \mathcal{A} = \{A \mid A = \sum_{i=1}^{N} \alpha_i A_i$ for some $\alpha_i \geq 0$, $\sum_{i=1}^{N} \alpha_i = 1\}$, is asymptotically stable;
3. there exist a full column rank matrix $L \in \mathbb{R}^{m\times n}$, $m \geq n$, and a family of matrices $\{\bar{A}_i \in \mathbb{R}^{m\times n} : \ i \in \mathcal{I}\}$ with strictly negative row dominating diagonal, i.e., for each $\bar{A}_i$, $i \in \mathcal{I}$ its elements satisfying
$$\hat{a}_{kk} + \sum_{k\neq l} |\hat{a}_{kl}| < 0, \ \ k = 1, \cdots, m,$$
such that the matrix relations $LA_i = \bar{A}_i L$ are satisfied.

It is interesting to notice that the nice property of $\bar{A}_i$ ($i \in \mathcal{I}$) implies the existence of a CQLF for the higher dimensional switched system. Unfortunately, applying the above theorem is still difficult because, in general, the numerical search for the matrix L is not simple. However, this equivalence bridges together two research fields. Existing results in the robust stability area, which has been extensively studied for over three decades, can be directly introduced to study the arbitrarily switching systems and vice versa. For example, it is known in the robust stability literature that the global attractiveness, (global) asymptotic stability, and (global) exponential stability are all equivalent for the polytopic uncertain linear time-variant systems. Hence, these stability concepts are also equivalent for switched linear systems under arbitrary switching.

Similar results can be developed for the discrete-time case based on results on the robust stability of discrete-time systems [Molchanov and Pyatnitskiy, 1989, Bhaya and Mota, 1994], namely:

Theorem 3.7. The arbitrarily switching linear system $x[k+1] = A_{\sigma(k)}x[k]$ where $A_{\sigma(t)} \in \{A_1, A_2, \cdots, A_N\}$, is asymptotically stable

under arbitrary switching if and only if there exists an integer $m \geq n$ and $L \in \mathbb{R}^{n\times m}$, $rank(L) = n$ such that for all A_i, $i \in \mathcal{I}$, there exists $\bar{A}_i \in \mathbb{R}^{m\times m}$ with the following properties:

1. $A_i^T L = L\bar{A}_i^T$,
2. each column of $\bar{A}_i$ has no more than n nonzero elements and

$$\|\bar{A}_i\|_\infty = \max_{1\leq k\leq m} \sum_{l=1}^{m} |\hat{a}_{kl}| < 1.$$

Interested readers may also refer to [Liberzon and Morse, 1999, DeCarlo et al., 2000, Liberzon, 2003, Lin and Antsaklis, 2009, Goebel et al., 2009] for further discussions and references on stability conditions under arbitrary switching.

3.1.3 Slow Switching

Hybrid systems may fail to preserve stability under arbitrary switchings. On the other hand, one may have some knowledge about possible discrete event dynamics in the hybrid systems, which may be deduced from partitions of the state space, such as invariant sets, guard sets etc. This knowledge may be transferred to restrictions on possible discrete transitions. For example, there must exist certain lower bounds on the time interval between two successive switchings, which may be due to the fact that the state trajectories have to spend some positive period of time in traveling from the initial set to certain guard sets, if these two sets are separated. With such kind of *prior* knowledge about the switching signals, we may derive less conservative stability results for a given hybrid system instead of just using the worst case arguments.

By studying the first motivating example where divergent trajectories are generated through switching between two stable systems, one notices that the unboundedness is caused by the failure to absorb the energy increase caused by the switching. In addition, when there is an unstable subsystem (e.g., controller failure or sensor fault), it may cost stability if one either stays too long at or switches too frequently to the unstable subsystem. Therefore, a natural question is what if we

restrict the switching signal to some constrained subclass of switchings. Intuitively, if one stays at stable subsystems long enough and switches less frequently, i.e., slow switching, one may dissipate the energy caused by switching or unstable modes, and maybe possible to attain stability. These ideas are proved to be reasonable and are captured by concepts like dwell time [Morse, 1996] and average dwell time [Hespanha and Morse, 1999] switching.

The most direct way to characterize the concept of slow switching is perhaps to request a lower bound on two consecutive switching times, which is known as dwell time [Morse, 1996]. Under the assumption that all subsystems in the hybrid system are exponentially stable with zero as the common equilibrium point, it can be shown that there exists a scalar $\tau_d > 0$ such that the hybrid system remains exponentially stable if the dwell time is larger than τ_d [Morse, 1996].

It fact, it really does not matter if one occasionally have a smaller dwell time between switching, provided this does not occur too frequently. This concept is captured by the concept of "average dwell-time" [Hespanha and Morse, 1999].

Definition 3.1. A positive constant τ_a is called the average dwell time if $N_\sigma(t) \leq N_0 + \frac{t}{\tau_a}$ holds for all $t > 0$ and some scalar $N_0 \geq 0$, where $N_\sigma(t)$ denotes the number of discontinuities of a given switching signal σ over $[0, t)$.

Here the constant τ_a is called the *average dwell time* and N_0 the *chatter bound*. The reason to call a class of switching signal satisfy

$$N_\sigma(t) \leq N_0 + \frac{t}{\tau_a}$$

have an average dwell no less than τ_a is because that

$$N_\sigma(t) \leq N_0 + \frac{t}{\tau_a} \Leftrightarrow \frac{t}{N_\sigma(t) - N_0} \geq \tau_a,$$

which means that on average the 'dwell time' between any two consecutive switchings is no smaller than τ_a. The idea is that there may exist consecutive switching separated by less than τ_a, but the average time interval between consecutive switchings is not less than τ_a.

Theorem 3.8. [Hespanha and Morse, 1999] Assume that all subsystems, $\dot{x} = A_i x$ for $i \in \mathcal{I}$, in the hybrid systems are exponentially stable. Then, there exists a scalar $\tau_a > 0$ such that the hybrid system is exponentially stable if the average dwell time is larger than τ_a.

Moreover, we can also obtain a bound on the decay rate [Hespanha and Morse, 1999].

Theorem 3.9. Given a positive scalar λ_0 such that $A_i + \lambda_0 I$ is stable for all $i \in \mathcal{I}$. Then, for any given $\lambda \in (0, \lambda_0)$, there is a finite constant τ_a such that the hybrid system is exponentially stable with decay rate λ provided that the average dwell time is no less than τ_a.

However, not all restrictions on switching signals can be easily captured by slow switching discussed above. For example, it is not straightforward to transform the invariant set constraints, guard set constraints and so on into dwell-time or average dwell time restrictions as these constraints are state dependent in the form of partitions of the state space. Hence, more general tools to study hybrid system stability are needed. Here, we introduce multiple Lyapunov functions to study hybrid system stability.

3.1.4 Multiple Lyapunov Functions

The stability analysis under constrained switching has been usually pursued in the framework of multiple Lyapunov functions (MLF). The basic idea is using multiple Lyapunov or Lyapunov-like functions, which may correspond to each single subsystem or certain region in the state space, concatenated together to produce a non-traditional Lyapunov function. The non-traditionality is in the sense that the MLF may not be monotonically decreasing along the state trajectories, may have discontinuities and be only piecewise differentiable. The reason for considering non-traditional Lyapunov functions is that traditional Lyapunov function may not exist for hybrid systems with restricted switching signals. For such cases, one still may construct a collection of Lyapunov-like functions, which only requires non-positive Lie-derivatives for certain subsystems in a particular region of the state space instead of being negative globally.

Lyapunov-like functions are defined as a family of real-valued functions $\{V_i, i = 1, \cdots, N\}$, each associated with the vector field $\dot{x} = f_i(x)$.

Definition 3.2. By saying that a subsystem has an associated Lyapunov-like function V_i in region Ω_i, we mean that

1. There exist constant scalars $\beta_i \geq \alpha_i > 0$ such that
$$\alpha_i \|x\|^2 \leq V_i(x) \leq \beta_i \|x\|^2$$
hold for any $x \in \Omega_i$;
2. For all $x \in \Omega_i$ and $x \neq 0$, $\dot{V}_i(x) < 0$.

Here $\dot{V}_i(x) = \frac{\partial V_i(x)}{\partial x} f_i(x)$. The first condition implies positiveness and radius unboundedness for $V_i(x)$ when $x \in \Omega_i$, while the second condition guarantees the decreasing of the abstracted energy, value of function $V_i(x)$, along trajectories of subsystem i inside Ω_i. Suppose that all these regions Ω_i cover the whole state space, then a cluster of Lyapunov-like functions is obtained. Via concatenating these Lyapunov-like functions together, we obtain a non-traditional Lyapunov function, called multiple Lyapunov function (MLF), which can be used to study the global stability of the switched and hybrid systems. MLF is proved to be a powerful tool for studying the stability of switched and hybrid systems.

Theorem 3.10. Consider a hybrid automaton H with $x = 0$ as its equilibrium point (i.e., $f(q, 0) = 0$ for all $q \in Q$) and $R((q_i, q_j), x) = x$ for any q_i, $q_j \in Q$. Assume that there exist an open set $D \subseteq X$ and a family of continuously differentiable functions $V_q : D \to \mathbb{R}$, $q \in Q$ such that for all $q \in Q$

- $V_q(0) = 0$,
- $V_q(x) > 0$ for all $x \in D \setminus \{0\}$,
- $\frac{\partial V_q}{\partial x}(x) f(q, x) \leq 0$ for all $x \in D$.

If for all $(\tau, q, x) \in \mathcal{E}_H(q_0, x_0)$ with $(q_0, x_0) \in Init$ and for all $\hat{q} \in Q$, the sequence $\{V_{q(\tau_i)}(x(\tau_i)) : q(\tau_i) = \hat{q}\}$ is non-increasing (or empty), then $x = 0$ is a stable equilibrium of H.

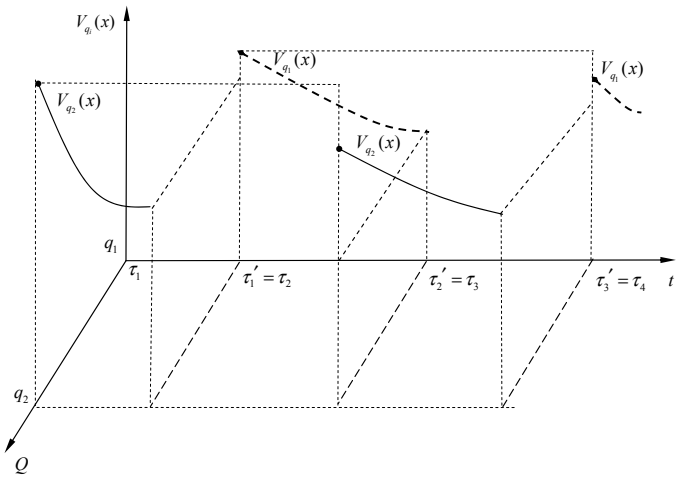


Figure 3.3: Illustration of Theorem 3.10. For every subsystem, its Lyapunov function's value V_i at the start point of each interval exceeds the value at the start point of the next interval on which the i-th subsystem is activated, then the switched or hybrid system is stable.

The above MLF theorem is adopted from the [Branicky, 1998] and written in the language of hybrid automata. Some comments are in order. Note that if the value of the Lyapunov like function V_q for every q at the entering point of each activating interval is always less than the value at the entering point of the previous activating interval, then the hybrid system is stable. This is illustrated in Figure 3.3. However, checking this condition could be difficult, since it may require the calculation of all possible trajectories and keeping track of all the time instants when each mode is being activated. Hence, it is difficult to construct the sequence $\{V_{q(\tau_i)}(x(\tau_i)) : q(\tau_i) = \hat{q}\}$. A simpler but more conservative condition is to request that $\{V_{q(\tau_i)}(x(\tau_i))\}$ forms a non-increasing sequence, as it implies $\{V_{q(\tau_i)}(x(\tau_i)) : q(\tau_i) = \hat{q}\}$ is non-increasing. Therefore, a direct corollary of above theorem can be presented as follows.

Corollary 3.1. Consider a hybrid automaton H with $x = 0$ as its equilibrium point, and $R((q_i, q_j), x) = x$ (or non-expansive) for any q_i,

$q_j \in Q$. Assume that there exists an open set $D \subseteq X$, and a family of continuously differentiable functions $V_q : D \to \mathbb{R},\ q \in Q$ such that for all $q \in Q$

- $V_q(0) = 0$,
- $V_q(x) > 0$ for all $x \in D \setminus \{0\}$,
- $\frac{\partial V_q}{\partial x}(x) f(q, x) \leq 0$ for all $x \in D$.

If for all $(\tau, q, x) \in \mathcal{E}_H(q_0, x_0)$ with $(q_0, x_0) \in Init$, the sequence $\{V_{q(\tau_i)}(x(\tau_i))\}$ is non-increasing, then $x = 0$ is a stable equilibrium of H.

For convenience, we also state the MLF theorem for the case when the model is given by switched systems.

Theorem 3.11. [DeCarlo et al., 2000] Consider a switched system consisting of a collection of subsystems $\dot{x} = f_i(x)$ with 0 as the common equilibrium point. Suppose that each subsystem has an associated Lyapunov-like function V_i in its active region Ω_i, each with equilibrium point $x = 0$. Also, suppose that $\bigcup_i \Omega_i = \mathbb{R}^n$. Let $s(t)$ be a class of piecewise-constant switching sequences such that $s(t)$ can take value i only if $x(t) \in \Omega_i$, and in addition

$$V_j(x(t_{i,j})) \leq V_i(x(t_{i,j}))$$

where $t_{i,j}$ denotes the time that the subsystems j is switched in from subsystem i, i.e., $x(t_{i,j}^-) \in \Omega_i$ while $x(t_{i,j}) \in \Omega_j$. Then, the switched system is stable under the switching signals $s(t)$.

The above MLF theorem requires that the Lyapunov-like functions do not increase their values at each switching instant, which is more conservative compared with Theorem 3.10. However, the conditions in Theorem 3.11 is easier to check as one do not have to memorize the previous activating history and store the values of Lyaounov-like functions at the entering or exiting point for each mode, which is needed to check for the conditions in Theorem 3.10. Due to its simplicity, in the next sections on the numerical analysis and synthesis using Multiple quadratic Lyapunov functions, we will adopt the form of Theorem 3.11.

3.1.5 Piecewise Quadratic Lyapunov Functions

The critical challenge of applying the MLF theorems to practical hybrid systems is how to construct a proper family of Lyapunov-like functions that satisfy

1. positive definiteness: $V_i(x_e) = 0$ and $V_i(x) > 0$ for $x \in \Omega_i$ and $x \neq x_e$;
2. have negative definite derivative: for $x \in \Omega_i$, $\dot{V}_i(x) = \frac{\partial V_i(x)}{\partial x} f_i(x) \leq 0$.

Usually it is very hard to find such $V_i(x)$. In the linear subsystem special case, piecewise quadratic Lyapunov-like functions could be good candidates. The most important advantage of piecewise quadratic Lyapunov-like functions might be that the problem can be formulated into LMIs where efficient software packages are available.

Therefore, in this subsection, we return to switched LTI system as

$$\dot{x}(t) = A_i x(t) \tag{3.12}$$

Notice that the origin ($x = 0$) is the common equilibrium. Since we do not assume that the subsystems, $\dot{x}(t) = A_i x(t)$, are stable, there may not exist a quadratic Lyapunov functions in a classical sense for each subsystem. However, it is still possible to restrict the subsystem in a certain region of the state space, say $\Omega_i \subset \mathbb{R}^n$, where the abstracted energy of the i-th subsystem could be decreasing along the trajectories inside this region (there is no requirement outside the region Ω_i). When all these regions Ω_i taken together cover the whole state space, then we obtain a cluster of Lyapunov-like functions. Broadly speaking, the problem entails searching for Lyapunov-like functions whose associated Ω-region cover the state space.

Assume that the state space $\mathbb{R}^n$ has a partition given by $\{\Omega_1, \cdots, \Omega_N\}$. Here, we aim to find conditions expressed in LMIs for the existence of quadratic Lyapunov-like function of the form $V_i(x) = x^T P_i x$ assigned to each region Ω_i. Dy definition, a Lyapunov-like function $V_i(x) = x^T P_i x$ needs to satisfy the following two conditions:

Condition 1: There exist constant scalars $\beta_i \geq \alpha_i > 0$ such that

$$\alpha_i \|x\|^2 \leq V_i(x) \leq \beta_i \|x\|^2$$

holds for any $x \in \Omega_i$.

Consider the quadratic Lyapunov-like function candidate, $V_i(x) = x^T P_i x$, and require that

$$\alpha_i x^T I x \leq x^T P_i x \leq \beta_i x^T I x,$$

holds for any $x \in \Omega_i$. That is

$$\begin{cases} x^T(\alpha_i I - P_i)x \leq 0 \\ x^T(P_i - \beta_i I)x \leq 0 \end{cases}$$

holds for any $x \in \Omega_i$. Here I is the identity matrix of proper dimensions.

Condition 2: For all $x \in \Omega_i$ and $x \neq 0$, $\dot{V}_i(x) < 0$.

This negativeness of the Lyapunov-like function's derivative along the trajectories of subsystem can be represented as: $\exists P_i$, $(P_i = P_i^T)$ such that

$$x^T[A_i^T P_i + P_i A_i]x < 0 \tag{3.13}$$

for $x \in \Omega_i$.

Note that the above positive definiteness and energy decreasing conditions both need to be satisfied in a local region, here Ω_i. In order to constrain the above two conditions to local regions, two steps are needed. First, the region must be expressed or constrained in regions that can be characterized by quadratic forms, $x^T Q x \geq 0$. Examples of such quadratic forms are cones and ellipsoids. If a region is described by half-planes

$$c_a^T x \geq 0 \text{ and } c_b^T x \geq 0$$

and

$$c_a^T x \leq 0 \text{ and } c_b^T x \leq 0$$

then the quadratic form characterizing the region is obtained by multiplying the two half-planes together.

$$x^T Q x \geq 0$$

where $Q = c_a c_b^T + c_b c_a^T$. Refer to [Boyd et al., 1994] where more general quadratic forms are used to express hyperplanes and polyhedra. Therefore, we assume that each region Ω_i has a quadratic representation or approximation

$$\Omega_i = \{x|\ x^T Q_i x \geq 0\}.$$

Second, a technique called $\mathcal{S}$-procedure [Boyd et al., 1994] is applied to replace a constrained stability conditions to conditions without constraints. To illustrate, let us consider the condition 2 above, that is $\exists P_i$, ($P_i = P_i^T$) such that

$$x^T[A_i^T P_i + P_i A_i]x < 0 \tag{3.14}$$

for $x \in \Omega_i$, i.e., $x^T Q_i x \geq 0$. By introducing a new unknown variable $\vartheta_i \geq 0$, there is potentially more freedom in finding a $P_i > 0$ and $\vartheta_i \geq 0$ satisfying the unconstrained relaxed condition

$$A_i^T P_i + P_i A_i + \vartheta_i Q_i < 0 \tag{3.15}$$

A solution to this relaxed problem is also a solution to the constrained problem, and the two problems are actually equivalent in certain cases, say $\exists x$ such that $x^T Q_i x > 0$.

Also applying $\mathcal{S}$-procedure to Condition 1, the above constrained inequalities are implied by the following LMIs

$$\begin{cases} \alpha_i I - P_i + \mu_i Q_i \leq 0 \\ P_i - \beta_i I + \nu_i Q_i \leq 0 \end{cases}$$

where $\mu_i \geq 0$ and $\nu_i \geq 0$ are unknown scalars. Define two scalars, $\alpha = \min_i\{\alpha_i\}$ and $\beta = \max_i\{\beta_i\}$. Notice that $0 < \alpha \leq \beta$.

Combining the above two conditions, we introduce methods to find quadratic Lyapunov-like functions within each partition, which guarantee that the abstract energy of the subsystem is decreasing for certain regions in the state space.

The condition that requires the decreasing of the Lyapunov-like functions' value at switching instant can be expressed as

$$x^T P_j x \leq x^T P_i x$$

for state x where the trajectory passes from region $\Omega_i$to Ω_j. The states where this condition is satisfied also have to be expressed or contained in regions characterized by quadratic forms. Assume this region, denoted as $\Omega_{i,j}$, can be expressed or approximated by a region,

$$\Omega_{i,j} = \{x|\ x^T Q_{i,j} x \geq 0\}.$$

Then the above inequality can be transformed into an LMI based also on $\mathcal{S}$-procedure as

$$P_j + \eta_{i,j} Q_{i,j} \leq P_i$$

In summary, the above discussion can be presented as the following form of sufficient conditions for the continuous-time linear hybrid system to be exponentially stable.

Theorem 3.12. [Pettersson and Lennartson, 2002] If there exist matrices P_i ($P_i = P_i^T$) and scalars $\alpha > 0$, $\beta > 0$, $\mu_i \geq 0$, $\nu_i \geq 0$, $\theta_i \geq 0$, $\vartheta_i \geq 0$ and $\eta_{i,j}$, solving the optimization problem:

$$\min \beta$$
$$s.t. \begin{cases} \alpha I + \mu_i Q_i \leq P_i \leq \beta I - \nu_i Q_i \\ A_i^T P_i + P_i A_i + \vartheta_i Q_i \leq -I \\ P_j + \eta_{i,j} Q_{i,j} \leq P_i \end{cases}$$

then the switched system (3.12) is exponentially stable.

The left-hand side of the first condition requires positive definiteness of the quadratic Lyapunov-like functions. The right-hand side is introduced to find an upper bound of the Lyapunov-like functions to determine an upper bound of the convergence rate. If only stability is of interest, this right-hand side of the inequality can be neglected. The second condition is the requirement that the energy is decreasing in every region Ω_i. The energy decrease has to be less than the negative identity matrix to conclude exponential stability. Stability is guaranteed if the right-hand side instead is zero. Finally, the third condition is the requirement that the energy is non-increasing at switching instants, where (i, j) is the set of tuples characterizing neighboring regions for which $x(t)$ can possibly travel from Ω_i to Ω_j.

Exponential stability is verified if there is a solution to the above LMI problem. In addition, a bound on the convergence rate can be estimated [Pettersson and Lennartson, 2002] as

$$\|x(t)\| \leq \sqrt{\frac{\beta}{\alpha}} e^{-\frac{1}{2\beta}t} \|x_0\|$$

where $x(t)$ is the continuous trajectory with initial state x_0.

3.2 Switching Stabilization

The above MLF results also provide methodologies to design switching logics between vector fields so as to achieve a stable trajectory since MLF results characterize the conditions on switching signals, under which the hybrid/switched system is stable. In this subsection, we will explicitly consider the design of stable switching signals for hybrid/switched systems. In particular, we consider the stabilizing switching signal design among a collection of LTI systems $\dot{x} = A_i x$. Interestingly, even when all subsystems are unstable, there still may exist stabilizing switching signals as illustrate in our second motivating example, i.e., Example 3.2.

3.2.1 Quadratic Switching Stabilization

In the switching stabilization literature, earlier efforts focused on quadratic stabilization for certain classes of systems. It can be shown that the existence of a stable convex combination of a pair of two stable LTI subsystem matrices implies the existence of a state-dependent switching rule that stabilizes the switched system along with a quadratic Lyapunov function [Wicks et al., 1998]. Formally, this result can stated as follows.

Theorem 3.13. If the matrix pencil $\gamma_\alpha(A_1, A_2)$ contains a stable matrix then there exists a piecewise constant switching signal that makes the switched system quadratically stable.

Here, the matrix pencil $\gamma_\alpha(A_1, A_2)$ is defined as a collection of matrices A_α that can be written as a convex combination of A_1 and A_2,

i.e., $A_\alpha = \alpha A_1 + (1-\alpha)A_2$ for some $\alpha \in [0,1]$. Furthermore, it can be shown that the stable convex combination condition is also necessary for the quadratic stabilizability of two-mode switched LTI system.

Theorem 3.14. [Feron, 1996] If there exists a quadratically stabilizing switching signal in the state feedback form, then the matrix pencil $\gamma_\alpha(A_1, A_2)$ contains a stable matrix.

Therefore, the existence of a stable convex combination state matrix is necessary and sufficient for the quadratic stabilizability of two mode switched LTI system.

Theorem 3.15. There exists a quadratically stabilizing switching signal in the state feedback form, if and only if the matrix pencil $\gamma_\alpha(A_1, A_2)$ contains a stable matrix.

A generalization to more than two LTI subsystems can be achieved by using a "min-projection strategy", i.e., $i = \arg\min_{i\in\mathcal{I}} x^T P A_i x$.

Theorem 3.16. [Pettersson and Lennartson, 2001] If there exist constants $\alpha_i \in [0,1]$, and $\sum_{i\in\mathcal{I}} \alpha_i = 1$ such that $A_\alpha = \sum_{i\in\mathcal{I}} \alpha_i A_i$, is stable then the min-projection strategy quadratically stabilizes the switched system.

However, the existence of a stable convex combination matrix A_α is only sufficient for switched LTI systems with more than two modes. There are example systems for which no stable convex combination state matrix exists, yet the system is quadratically stabilizable using certain switching signals [Liberzon et al., 1999]. A necessary and sufficient condition for the quadratic stabilizability of switched controller systems was derived in [Skafidas et al., 1999].

Theorem 3.17. [Skafidas et al., 1999] The switched system is quadratically stabilizable *if and only if* there exists a positive definite real symmetric matrix $P = P^T > 0$ such that the set of matrices $\{A_i P + P A_i^T\}$ is strictly complete, i.e., for any $x \in \mathbb{R}^n/\{0\}$, there exists $i \in \mathcal{I}$ such that $x^T(A_i P + P A_i^T)x < 0$. In addition, a stabilizing switching signal can be selected as $\sigma(t) = \min_i\{x^T(t)(A_i P + P A_i^T)x(t)\}$. □

Analogously, for the discrete-time case, it is necessary and sufficient for quadratic stabilizability to check whether there exists a positive symmetric matrix P such that the set of matrices $\{A_i^T P A_i - P\}$ is strictly complete [Skafidas et al., 1999]. Obviously, the existence of a convex combination of state matrices A_α automatically satisfies the above strict completeness conditions due to convexity, while the inverse statement is not true in general. Unfortunately, checking the strict completeness of a set of matrices is NP hard [Skafidas et al., 1999].

Other approaches include [Wicks et al., 1998] and extensions of [Wicks et al., 1998] to the output-dependent switching and discrete-time cases [Liberzon and Morse, 1999, Zhai et al., 2003]. For robust stabilization of polytopic uncertain switched linear systems, a quadratic stabilizing switching law was designed based on LMI techniques in [Zhai et al., 2003].

Quadratic stability means that there exists a positive constant ϵ such that $\dot{V}(x) \leq -\epsilon x^T x$. All of these methods guarantee stability by using a common quadratic Lyapunov function, which is conservative in the sense that there are switched systems that can be asymptotically or even exponentially stabilized without using a common quadratic Lyapunov function [Hespanha et al., 2005]. There have been some results in the literature that propose constructive synthesis methods in switched systems using multiple Lyapunov functions [DeCarlo et al., 2000]. A stabilizing switching law design based on multiple Lyapunov functions was proposed in [Wicks and DeCarlo, 1997], where piecewise quadratic Lyapunov functions were employed for two mode switched LTI systems. Exponential stabilization for switched LTI systems was considered in [Pettersson, 2003], also based on piecewise quadratic Lyapunov functions, and the synthesis problem was formulated as a bilinear matrix inequality (BMI) problem. In the next subsection, we will briefly describe the BMI conditions derived in [Pettersson, 2003].

3.2.2 Piecewise Quadratic Switching Stabilization

Recall the sufficient conditions stated in Theorem 3.12 for the continuous-time linear system (3.12) to be exponentially stable: If there exist matrices P_i ($P_i = P_i^T$) and scalars $\alpha > 0$, $\beta > 0$, $\mu_i \geq 0$, $\nu_i \geq 0$, $\theta_i \geq 0$, $\vartheta_i \geq 0$ and $\eta_{i,j}$, that satisfy

$$\begin{cases} \alpha I + \mu_i Q_i \leq P_i \leq \beta I - \nu_i Q_i \\ A^T P_i + P_i A + \vartheta_i Q_i \leq -I \\ P_j + \eta_{i,j} Q_{i,j} \leq P_i \end{cases}$$

then the switched linear system (3.12) is exponentially stable. There are two more problems needed to be solved for switching controller synthesis: the partition of the state space and the identification of the set that switching occurs, i.e., $\Omega_{i,j}$.

The purpose of dividing the whole state space $\mathbb{R}^n$ into partitions, denoted by Ω_i, is to facilitate the identification of a Lyapunov-like function for each one of these subsystems. After successfully obtaining these Lyapunov-like functions associated with each region Ω_i, one may patch them together using the following conditions in Theorem 3.12 so as to guarantee the global stability.

For this, it is necessary to require that all these regions Ω_i cover the whole state space, i.e.,

- Covering Property: $\Omega_1 \cup \cdots \cup \Omega_N = \mathbb{R}^n$.

This condition merely says that there are no regions in the state space where none of the subsystems is activated.

Since we will restrict our attention to quadratic Lyapunov-like functions for reason of computational efficiency, we will consider regions given (or approximated) by quadratic forms

$$\Omega_i = \{x \in \mathbb{R}^n | \; x^T Q_i x \geq 0\},$$

where $Q_i \in \mathbb{R}^{n \times n}$ are symmetric matrices, and $i \in \mathcal{I} = \{1, \cdots, N\}$.

The following lemma gives a sufficient condition for the covering property.

Lemma 3.1. [Pettersson, 2003] If for every $x \in \mathbb{R}^n$

$$\sum_{i=1}^{N} \theta_i x^T Q_i x \geq 0 \tag{3.16}$$

where $\theta_i \geq 0$, $i \in \mathcal{I}$, then $\bigcup_{i=1}^{N} \Omega_i = \mathbb{R}^n$.

Consider the *largest region function strategy*, i.e.,

$$i(x) = \arg\left(\max_{i\in\mathcal{I}} x^T Q_i x\right) \tag{3.17}$$

That is the selection of subsystems (at state x) corresponds to the largest value of the region function $x^T Q_i x$. This switching strategy was previously introduced in [Pettersson, 2003] for continuous-time switched linear systems.

In order to guarantee exponential stability we also need to make sure that

1. Subsystem i is active only when $x(t) \in \Omega_i$,
2. When switching occurs, it is required to guarantee that Lyapunov function value is not increasing.

To verify 1) for the largest region function strategy (3.17), suppose that the covering condition (3.16) holds, i.e.,

$$\sum_{i=1}^{N} \theta_i x^T Q_i x \geq 0$$

for some $\theta_i \geq 0$, $i \in \mathcal{I}$. Then, based on the largest region function strategy, $i(x) = \arg\left(\max_{i\in\mathcal{I}} x^T Q_i x\right)$, the state x with current active mode i satisfies $x^T Q_i x \geq 0$. This implies $x \in \Omega_i$. So the first condition holds for the largest region function strategy.

The second energy decreasing condition when switching is not easy to handle. It is because we lack the information about the direction of the vector fields on the switching hyperplane. In other words, when $x \in \Omega_i \cap \Omega_j$, i.e, on the switching plane, it is difficult to determine which is the direction of the trajectory where the switching is occurring.

Then, we make a compromise to require that

$$x^T P_i x = x^T P_j x$$

for states at the switching plane, i.e., $x \in \Omega_i \cap \Omega_j$.

The set $\Omega_i \cap \Omega_j$ can be represented as the following quadratic form

$$\Omega_i \cap \Omega_j = \{x | x^T(Q_i - Q_j)x = 0\}.$$

Again, applying $\mathcal{S}$-procedure, we obtain

$$P_i - P_j + \eta_{i,j}(Q_i - Q_j) = 0$$

for an unknown scalar $\eta_{i,j}$.

In summary, the above discussions can be presented as the following sufficient conditions for the continuous-time linear system (3.12) to be exponentially stabilized.

Theorem 3.18. [Pettersson, 2003] If there exist matrices P_i ($P_i = P_i^T$) and scalars $\alpha > 0$, $\beta > 0$, $\mu_i \geq 0$, $\nu_i \geq 0$, $\theta_i \geq 0$, $\vartheta_i \geq 0$ and $\eta_{i,j}$, solving the optimization problem:

$$\begin{aligned} &\min \beta \\ s.t. \quad &\begin{cases} \alpha I + \mu_i Q_i \leq P_i \leq \beta I - \nu_i Q_i \\ A^T P_i + P_i A + \vartheta_i Q_i \leq -I \\ P_j = P_i + \eta_{i,j}(Q_i - Q_j) \\ \theta_1 Q_1 + \cdots + \theta_N Q_N \geq 0 \end{cases} \end{aligned}$$

for all $i, j \in \mathcal{I} = \{1, \cdots, N\}$, then the linear system (3.12) can be exponentially stabilized by the largest region function strategy defined by Equation (3.17).

Some remarks are in order. First, the optimization problem above is a Bilinear Matrix Inequality (BMI) problem, due to the product of unknown scalars and matrices. BMI problems are NP-hard, and not computationally efficient. However, practical algorithms for optimization problems over BMIs exist and typically involve approximations, heuristics, branch-and-bound, or local search. One possible way to compute the BMI problem is to grid up the unknown scalars, and then solve a set of LMIs for fixed values of these parameters. It is argued that the gridding of the unknown scalars can be made quite sparsely [Pettersson, 2003]. The design procedure has been extended to the discrete time case in [Lin and Antsaklis, 2008].

For further information on the stability and stabilization of switched and hybrid systems, interested readers may refer to survey

papers [Liberzon and Morse, 1999, Michel, 1999, DeCarlo et al., 2000, Hespanha, 2004a, Lin and Antsaklis, 2009, Goebel et al., 2009], the monographs [Liberzon, 2003, Johansson, 2003a, Goebel et al., 2012] and the references cited therein.

3.3 Optimal Control

Optimal control has been an active research area in the control community for decades since the illustrious seminal contributions in the 50s by Pontryagin and Bellman. The optimal control problems for hybrid systems have attracted a lot of attention since the mid nineties, and many results may be found in the control and computer science literature. Early efforts have been devoted to the extensions of the maximum principle and dynamic programming techniques to hybrid systems, see e.g., [Branicky et al., 1998, Sussmann, 1999, Hedlund and Rantzer, 2002, Cassandras et al., 2001, Shaikh and Caines, 2007] and the references therein. Here we choose to focus on some recent developments on the computational approaches to the optimal control synthesis for switched systems and piecewise affine systems.

3.3.1 Optimal Control for Switched Systems

To illustrate the idea of hybrid optimal control design, we focus on switched systems (with continuous controls) and consider designing switching signals and continuous control signals for a collection of dynamical systems

$$\dot{x}(t) = f_{\sigma(t)}(x(t), u(t)) \tag{3.18}$$

where $x(t) \in X \subseteq \mathbb{R}^n$ is the state vector and $u(t) \in U$ is the continuous control input. The current mode is assumed to be assigned by a discrete input $\sigma(t)$, which selects the active mode: $\sigma(t) \in Q = \{1, 2, \cdots, N\}$. Therefore, all switchings are controlled, and there is no autonomous switching by assumption.

Suppose that the switching signal is an independent design parameter. The optimal control problem is to find both the optimal con-

trol input $u(t)$ and optimal switching signal $\sigma(t)$ to optimize the cost functional

$$J(\sigma, u) = \sum_{k=0}^{M-1} \int_{t_k}^{t_{k+1}} L_\sigma(t)(x, u)dt \tag{3.19}$$

subject to (3.18) while driving the system from an initial state (x_0, q_0) at time t_0 to a final state (x_f, q_f) at time t_f, where the end time t_f is free (not fixed). Here, M is a design variable representing the number of switchings occurring during the time period $[t_0, t_f]$. M is finite; the switching instants in the time period $[t_0, t_f]$ are

$$t_0 < t_1 < t_2 < \cdots < t_M < t_f.$$

For the switched control system, the aim of optimal control is to seek appropriate switching and control strategies to optimize a certain performance index.

Embedding Optimization

The basic idea of the embedding optimization method is to "smooth" the switching signal by a continuous-valued signal through a convex combination, see e.g., [Bengea and DeCarlo, 2005]. Hence the optimal control problem can be solved using traditional approaches as all signals are now continuous. Hopefully, a bang-bang type optimal control policy is derived that can be used to recover the switching signal. To illustrate the main idea, we consider a two-mode switched system as

$$\dot{x}(t) = f_{\sigma(t)}(x(t), u(t)), \quad \sigma(t) \in Q = \{1, 2\} \tag{3.20}$$

where $x(t) \in X \subseteq \mathbb{R}^n$ is the state vector, continuous input $u(t) \in U$. The current mode is assumed to be assigned by a discrete input $\sigma(t)$. Therefore, all switching is controlled, and there is no autonomous switching by assumption.

Suppose that the switching signal is an independent design parameter. The optimal control problem is to find both the optimal control input and optimal switching signal to optimize the cost function (3.19) subject to (3.20) while driving the system from an initial state x_0 at time t_0 to a final state x_f at time t_f, where the end time t_f is free.

Here, M is an design variable representing the number of switching occurring during the time period $[t_0, t_f]$, therefore, M is always finite. Denote the switching instants in the time period $[t_0, t_f]$ as

$$t_0 < t_1 < t_2 < \cdots < t_M < t_f.$$

First, we re-write the switched system as a convex combination

$$\dot{x}_E(t) = [1 - \sigma_E(t)] f_1(x_E(t), u_0(t)) + \sigma_E(t) f_2(x_E(t), u_1(t)), \quad (3.21)$$

where $\sigma_E(t)$ is a time-varying function that takes value in the closed set $[0, 1]$. It is clear that the equation (3.21) will reduce to switched system (3.20) if $\sigma_E(t)$ only take values at the boundary, i.e., only takes discrete values 0 or 1. It is not surprising that, given the same initial condition x_0, Equation (3.21) should be able to generate more trajectories than the switched system (3.20) due to the enlarged domain of "switching control" signals $\sigma_E(t) \in [0, 1]$ and independent continuous-time controls $u_0(t)$, $u_1(t) \in U$. However, it can be shown that any trajectory generated by the embedded system (3.21) can be approximated by a solution of the switched system (3.20).

Theorem 3.19. [Bengea and DeCarlo, 2005] For the embedded system (3.21) with a given initial condition x_0 and control triple $\sigma_E(t) \in [0, 1]$, and $u_0(t)$, $u_1(t) \in U$ during the period $[t_0, t_f]$, assume that a solution exists and is unique. Let's denote the solution as $x_E(t)$ with $x_E(t_0) = x_0$. Then, for any desired trajectory-approximation error $\epsilon > 0$, there are switching signals $\sigma_\epsilon(t) \in \{0, 1\}$, and control inputs $u_\epsilon(t) \in U$ defined on $[t_0, t_f]$ such that the generating switching trajectory from the same initial condition $x(t_0) = x_0$ has the property $\|x(t) - x_E(t)\| < \epsilon$ for all $t \in [t_0, t_f]$.

Theorem 3.19 implies that the set of state trajectories of switched system (3.20) is dense in the set of state trajectories of the embedded system (3.21). Therefore, we can approximate the solutions of the embedded system (3.21) by forcing the signal $\sigma_E(t)$ take values only at the boundary, i.e., bang-bang type implementation. The basic idea is to transform the optimal control for the switched system (3.20) into its embedded form (3.21), and then solve the optimal control problem for

the embedded system (3.21) with respect to the following embedded cost function

$$\begin{aligned} & J_E(\sigma_E, u_0, u_1) \\ = & \int_{t_0}^{t_f} \{[1-\sigma_E(t)]L_0(x_E(t), u_0(t)) + \sigma_E(t)L_1(x_E(t), u_1(t))\}\, dt \end{aligned}$$

This becomes a classical optimal control problem, and many methods exist to solve it. The solution is denoted as $(\sigma_E^*, u_0^*, u_1^*) = \arg\min_{\sigma_E \in [0,1], u_0, u_1 \in U} J_E(\sigma_E, u_0, u_1)$

If the optimal control $\sigma_E^*(t)$ is of "bang-bang" type, i.e., $\sigma_E^*(t) \in \{0, 1\}$, then we already solved the optimal control problem for the original switched system (3.20). If $\sigma_E^*(t) \in (0, 1)$ for almost all time instants $t \in [t_0, t_f]$, then based on the above theorem we can obtain a sub-optimal solution with arbitrary small degradation of performance.

Consider a simplified car mode with two gears from the literature

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= g_{\sigma(t)}(x_2)u(t), \end{aligned}$$

where x_1 is the position, x_2 is the velocity of the car respectively. The inputs $u(t) \in U = [-1, 1]$ represent control of the break or throttle. The switching signal $\sigma(t) \in \{0, 1\}$, which captures the speed-dependent efficiencies $g_0(x_2)$ and $g_1(x_2)$.

The embedded system can be represented as follows

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = (1 - v(t)) \begin{bmatrix} x_2 \\ g_0(x_2)u_0(t) \end{bmatrix} + v(t) \begin{bmatrix} x_2 \\ g_1(x_2)u_1(t) \end{bmatrix}$$

We consider time optimal control with $L_0 = L_1 = 1$. So, the cost function

$$\begin{aligned} & J_E(x, v, u_0, u_1) \\ = & \int_0^{t_f} \{[1 - v(t)]L_0(x(t), u_0(t)) + v(t)L_1(x(t), u_1(t))\} dt \\ = & \int_0^{t_f} dt = t_f \end{aligned}$$

It is assumed that $t_0 = 0$, $x(0) = [-5, 0]^T$ and $x(t_f) = [0, 0]^T$ with further constraints $\sigma(0) = 0$ and $\sigma(t_f) = 0$. The optimal control signals for the embedded optimal control problem can be solved, and we obtain the following conclusion.

The Hamiltonian associated with the embedded system is with $\lambda = [\lambda_1, \lambda_2]^T$

$$\begin{aligned} & H_E(t, x, u_0, u_1, v, \lambda^0, \lambda) \\ = \; & \lambda_0[(1-v)L_0 + vL_1] + \lambda^T \left[(1-v) \begin{bmatrix} x_2 \\ g_0(x_2)u_0 \end{bmatrix} + v \begin{bmatrix} x_2 \\ g_1(x_2)u_1 \end{bmatrix} \right] \\ = \; & v \cdot \lambda_2[g_1(x_2)u_1 - g_0(x_2)u_0] + [\lambda_0 + \lambda_1 x_2 + \lambda_2 g_0(x_2)u_0] \end{aligned}$$

Based on the Maximal Principle, the following costate equations can be obtained

$$\begin{bmatrix} \dot{\lambda}_1 \\ \dot{\lambda}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -\lambda_1 - \lambda_2(1-v^*)u_0^* \frac{g_0}{dx_2}|_{x_2^*} - \lambda_2 v^* u_1^* \frac{g_1}{dx_2}|_{x_2^*} \end{bmatrix} \tag{3.22}$$

There exists $0 < t_1 < t_f$ such that

1. If $t < t_1$ and $x_2^* < 0.5$, then $v^*(t) = 0$ and $u_0^*(t) = 1$;
2. If $t < t_1$ and $x_2^* > 0.5$, then $v^*(t) = 1$ and $u_1^*(t) = 1$;
3. If $t > t_1$ and $x_2^* < 0.5$, then $v^*(t) = 0$ and $u_0^*(t) = -1$;
4. If $t > t_1$ and $x_2^* > 0.5$, then $v^*(t) = 1$ and $u_1^*(t) = -1$.

Here, the t_1 can be obtained by solving $\lambda_2(t) = 0$.

It can be observed that the optimal control is of bang-bang type, while the optimal throttle/break control is either $+1$ or -1.

Two-stage Optimization

The embedding optimization method is effective if the optimal signal $\sigma_E(t)$ takes value on the boundary. However, it is difficult to guarantee this situation and not straightforward to recover if this does not happen. Hence, it may be unavoidable to deal with switching signals as discrete values directly. As the switching signal is a discontinuous function of time and maybe highly nonlinear, the optimization is usually difficult and non-convex in nature. To deal with such

a difficulty, one popular approach is two-stage optimization, see e.g., [Xu and Antsaklis, 2004].

To illustrate the main idea of two-stage optimization, we consider a continuous-time switched control system

$$\dot{x}(t) = A_{\sigma(t)}x(t) + B_{\sigma(t)}u(t), \;\; x(0) = x_0 \tag{3.23}$$

and a fixed end-time t_f. The optimal control problem is to find a piecewise continuous input $u(t)$, and a switching signal $\sigma(t)$, such that the quadratic cost functional

$$J(\sigma, u) = \frac{1}{2}x(t_f)^T Q_f x(t_f) + \int_{t_0}^{t_f} \left(\frac{1}{2}x^T Q x + \frac{1}{2}u^T R u\right) dt \tag{3.24}$$

is minimized, where $Q_f \geq 0$, $Q > 0$, and $R > 0$.

In the problem, we need to find an optimal control solution (σ^*, u^*) such that

$$J(\sigma^*, u^*) = \min_{\sigma, u} J(\sigma, u).$$

Note that if we fix the switching signal, then the problem reduces to a conventional optimal control problem for linear time-varying systems. This idea leads to the two-stage strategy for solving the problem.

Stage 1: Fixing a switching signal, solve the optimal control problem for the corresponding time-varying system.

Stage 2: Regarding the optimal control for each switching signal as a function

$$J_1(\sigma) = \min_{u} J(\sigma, u),$$

minimize J_1 with respect to the switching signal σ.

The following lemma from [Xu and Antsaklis, 2004] provides support to this two-stage decomposition.

Lemma 3.2. Consider the optimal control problem (3.23)-(3.24). For a given $x_0 \in \mathbb{R}^n$, suppose that

1. an optimal solution (σ^*, u^*) exists; and
2. for any fixed switching index $i_0, \cdots, i_k$, there exist a time sequence $t_1, \cdots, t_k$ and a control input u, such that the cost function $J(x_0, t_1, \cdots, t_k, i_0, \cdots, i_k, u)$ is minimized.

Then, we have

$$J(x_0, \sigma^*, u^*) = \min_{\{i_0,\cdots,i_k\}} \min_{\{t_1,\cdots,t_k\}} J(x_0, t_1, \cdots, t_k, i_0, \cdots, i_k, u). \quad (3.25)$$

The two-stage optimization method actually provides a basic framework for approaching the optimization problems of switched and hybrid systems. To better illustrate the two-stage strategy, consider the following simple example from the literature with two subsystems and only one switching with fixed index sequence.

For a switched system

$$\begin{aligned} \dot{x}(t) &= A_1 x(t) + B_1 u(t), \;\; t_0 < t < t_1 \\ \dot{x}(t) &= A_2 x(t) + B_2 u(t), \;\; t_1 \le t \le t_f \end{aligned}$$

where t_0 and t_f are given, we are to find an optimal switching time t_1 and an optimal input u to minimize the quadratic cost function

$$J = x(t_f)^T Q_f x(t_f) + \int_{t_0}^{t_f} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} dt.$$

By introducing a state variable x_{n+1} corresponding to the switching instant t_1, and a new scaled time variable τ with

$$\begin{aligned} t &= t_0 + (x_{n+1} - t_0)\tau, && 0 \le \tau \le 1 \\ t &= x_{n+1} + (t_f - x_{n+1})(\tau - 1), && 1 \le \tau \le 2 \end{aligned}$$

the problem is converted into finding optimal x_{n+1} and optimal control u for system

$$\begin{aligned} &\tfrac{dx(\tau)}{d\tau} = (x_{n+1} - t_0)(A_1 x + B_1 u) \\ &\tfrac{dx_{n+1}}{d\tau} = 0 \end{aligned} \quad 0 \le \tau \le 1$$

$$\begin{aligned} &\tfrac{dx(\tau)}{d\tau} = (t_f - x_{n+1})(A_2 x + B_2 u) \\ &\tfrac{dx_{n+1}}{d\tau} = 0 \end{aligned} \quad 1 \le \tau \le 2$$

with the quadratic cost function

$$\begin{aligned} J = x(2)^T Q_f x(2) &+ \int_0^1 (x_{n+1} - t_0) \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} dt \\ &+ \int_1^2 (t_f - x_{n+1}) \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} dt. \end{aligned}$$

The latter can be addressed by solving parameterized general Riccati equations which can be derived using dynamic programming. To determine the optimal switching time, we need to know $\partial J/\partial x_{n+1}$, that is, the derivative of J with respect to the switching instant. This derivative can be obtained based on the solution of the ordinary differential equations obtained by differentiating the Riccati equations with respect to the switching instants. Together with the corresponding Hamilton-Jacobi-Bellman (HJB) equations, the optimal solution can be obtained by solving a set of differential-algebraic equations.

The optimal control problem of switched systems is in general difficult to solve due to the involvement of the switching signal. For piecewise affine systems with state/input constraints, some efficient computational schemes have been established, which will be developed in the next subsection.

3.3.2 Optimal Control of Piecewise Affine Systems

Contrary to switched systems, all switchings in piecewise affine (PWA) systems are autonomous, induced by the partition of the state space.

Continuous-time Piecewise Affine Systems

Consider the class of continuous-time PWA systems with state+input constraints

$$\begin{array}{l} \dot{x} = A_i x + B_i u \\ G_i x + H_i u \geq 0 \end{array} \quad \text{if } x \in \Omega_i \tag{3.26}$$

where $\{\Omega_i\}_{i\in I}$ is a partition of the state space into a number of closed polyhedral regions, each of which contains the origin.

For the polyhedrons Ω_i's, we can construct matrices E_i and F_i such that

$$E_i x \geq 0, \;\; x \in \Omega_i,$$

and

$$F_i x = F_j x, \;\; x \in \Omega_i \cap \Omega_j.$$

The optimal control problem is to bring the system to $x(\infty) = 0$ from an arbitrary initial state $x(0)$, while limiting the piecewise quadratic cost

$$J(x_0, u) = \int_0^\infty (x^T Q_i x + u^T R_i u) dt,$$

where $Q_i \geq 0$, $R_i \geq 0$ and $i(t)$ is defined so that $x(t) \in \Omega_{i(t)}$.

Theorem 3.20. [Rantzer and Johansson, 2000, Johansson, 2003a] Assume the existence of symmetric matrices T and U_i, such that U_i has nonnegative entries and satisfy

$$\begin{bmatrix} A_i^T P_i + P_i A_i + Q_i & P_i B_i \\ B_i^T P_i & R_i \end{bmatrix} - \begin{bmatrix} E_i & 0 \\ G_i & H_i \end{bmatrix}^T U_i \begin{bmatrix} E_i & 0 \\ G_i & H_i \end{bmatrix} > 0, \;\; i \in I$$

where $P_i = F_i^T T F_i$, $i \in I$. Then, every continuous and piecewise continuously differentiable trajectory $x(t) \in \cup \Omega_i$ of PWA system with $x(\infty) = 0$, $x(0) = x_0 \in \Omega_0$ satisfies

$$J(x_0, u) \geq \sup_{T, U_i} x_0^T P_{i_0} x_0.$$

The theorem gives a lower bound on the optimal cost. The computation of an upper bound can be obtained by studying specific control laws. To illustrate, let's consider the following example from [Johansson, 2003a].

Example 3.3. Consider the system

$$\dot{x}(t) = \begin{cases} A_1 x(t) + B_1 u(t) & x_1 < 0 \\ A_2 x(t) + B_2 u(t) & x_1 \geq 0 \end{cases}$$

where $A_1 = \begin{bmatrix} -5 & 4 \\ -1 & -2 \end{bmatrix}$, $A_2 = \begin{bmatrix} -2 & -4 \\ 10 & -2 \end{bmatrix}$, $B_1 = B_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, and the associated cost function

$$J = \int_0^{+\infty} \left(x^T(t) Q_i(t) x(t) + r_i u^2(t) \right) dt,$$

where

$$Q_1 = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \qquad Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}$$
$$r_1 = 2 \qquad r_2 = 4$$

with initial condition $x = \begin{bmatrix} -1 & 1 \end{bmatrix}^T$.

Solving the matrix inequalities in Theorem 3.20 gives the lower bound

$$J(x_0, u) \geq 1.9171,$$

where the bound can be attained with the control law

$$u(t) = \begin{cases} -r_1^{-1} B_1 P_1 x(t) & x_1 < 0 \\ -r_2^{-1} B_2 P_2 x(t) & x_1 \geq 0 \end{cases}$$

with

$$P_1 = \begin{bmatrix} 0.1457 & -0.2444 \\ -0.2444 & 1.2826 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 2.6076 & 0.5199 \\ 0.5199 & 1.2826 \end{bmatrix}$$

□

Discrete-time Piecewise Affine Systems

For discrete-time piecewise affine (PWA) systems, the optimal control problems have been extensively investigated in the literature. Discrete-time PWA systems are defined by partitioning the state and input space into polyhedral regions, and associating with each region an affine control system, namely

$$x_{k+1} = A_i x_k + B_i u_k + f_i, \quad \text{for } \begin{bmatrix} x_k \\ u_k \end{bmatrix} \in \Omega_i \tag{3.27}$$

Here the state $x \in \mathbb{R}^n$, the input $u \in \mathbb{R}^m$, and $\Omega_i \subseteq \mathbb{R}^{n+m}$ are partitions of the state and input space into a number of compact polyhedral regions with no common interiors where each polyhedron contains the origin, matrices A_i, B_i and vector f_i of compatible dimensions.

Define the following quadratic cost function with a given finite time horizon N:

$$J(U_N, x(0)) = x_N^T P x_N + \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k),$$

where $U_T = \{u_0, \cdots, u_{N-1}\}$ is the control sequence, matrices $P = P^T$, $Q = Q^T \geq 0$ and $R = R^T > 0$.

The problem of optimal control for the PWA system is to seek a control sequence $u_0, \cdots, u_{N-1}$ to minimize the cost function:

$$\begin{aligned} J^*(x_0) &= \min_{U_N)} J(U_N, x_0) \\ s.t. & \quad \begin{cases} x_{k+1} = A_i x_k + B_i u_k + f_i, \text{ if } \begin{bmatrix} x_k \\ u_k \end{bmatrix} \in \Omega_i \\ x_N \in \chi_f \end{cases} \end{aligned}$$

where χ_f is the terminal region.

Any optimal control sequence

$$U_N^* = \{u_0^*(x_0), \cdots, u_{N-1}^*(x_0)\},$$

is said to be a minimizer of the cost function.

We will also denote $\chi_k \in \mathbb{R}^n$ the set of initial states x_k at time k ($0 \leq k \leq N$), for which the optimal control problem is feasible. Mathematically, χ_k is recursively defined as

$$\begin{aligned} \chi_k &= \left\{ x \in \mathbb{R}^n | \exists u, i, \ A_i x + B_i u + f_i \in \chi_{k+1}, \text{ and } \begin{bmatrix} x \\ u \end{bmatrix} \in \Omega_i \right\} \\ \chi_N &= \chi_f \end{aligned}$$

We will assume that the optimal control problem admits at least one minimizer for each feasible $x(0)$. The following result characterizes the structural properties of the optimal control law.

Theorem 3.21. [Borrelli et al., 2005] The solution of the above optimal control problem is a feedback control law of the form

$$u_k^*(x_k) = F_k^i x_k + G_k^i, \ \text{ if } x_k \in \chi_k^i.$$

where $\{\chi_k^i\}_{i=1}^{N_k}$ is a partition of the set χ_k of feasible states x_k and the closure $\bar{\chi}_k^i$ of χ_k^i is in the form

$$\bar{\chi}_k^i = \{x : x^T L_k^i(j) x + M_K^i(j) \leq K_k^i(j), \ j = 1, \cdots, n_K^I\}, \ i = 1, \cdots, N_k$$

In addition, if the minimizer $U_N^*(x_0)$ is unique for all x_0, then, $\{\chi_k^i\}_{i=1}^{N_k}$ is a polyhedral partition of the set χ_k of feasible state x_k.

The theorem illustrates that the optimal control law possesses a PWA form and the partition is defined by quadratic surfaces. In the case that the minimizer is unique, each region of the partition turns out to be a polyhedron, which in general is much simpler than a region defined by quadratic surfaces.

Computation of Optimal Control Inputs

The computation of optimal controls so far is based on the enumeration of all possible switching sequences of the hybrid systems, the number of which grows exponentially with the time horizon. Here, we will transform a PWA system into its equivalent mixed logic dynamical system (MLDS) framework [Bemporad and Morari, 1999], within which mixed-integer programming is employed. In particular, when the model of the system is an MLDS model and the performance is quadratic, the optimization problem can be casted as a mixed-integer quadratic programming [Bemporad and Morari, 1999, Borrelli et al., 2005, Borrelli, 2003].

Mixed logical dynamical systems are computationally motivated representations of hybrid systems that consist of a collection of linear difference equations involving both real and $\{0, 1\}$ variables, subject to a set of linear inequalities. The key idea of the approach is to embed the logic part in the state equations by transforming propositional logic into *mixed-integer inequalities*, i.e. linear inequalities involving both *continuous variable* $x \in \mathbb{R}^n$ and binary/ logical variables $\delta \in \{0, 1\}$.

As an illustrative example, we convert a piecewise affine system into its mixed logic dynamical system representation from [Bemporad and Morari, 1999].

Example 3.4. Consider the following discrete-time piecewise affine system

$$x(t+1) = \begin{cases} 0.8x(t) + u(t) & \text{if } x(t) \geq 0 \\ -0.8x(t) + u(t) & \text{if } x(t) < 0 \end{cases}$$

where $x(t) \in [-10, 10]$, and $u(t) \in [-1, 1]$. The condition $x(t) \geq 0$ can

be associated to a binary variable $\delta(t)$ such that

$$[\delta(t) = 1] \leftrightarrow [x(t) \geq 0].$$

This equivalence can be expressed by the inequalities

$$\begin{aligned} -m\delta(t) &\leq x(t) - m, \\ -(M+\epsilon)\delta(t) &\leq -x(t) - \epsilon \end{aligned}$$

where $M = \max_{x\in[-10,10]}(x) = 10$, $m = \min_{x\in[-10,10]}(x) = -10$, and ϵ is a small positive scalar. Then the state equation can be written as

$$x(t+1) = 1.6\delta(t)x(t) - 0.8x(t) + u(t),$$

By defining a new variable $z(t) = \delta(t)x(t)$, the above equation can be expressed as

$$\begin{cases} x(t+1) = 1.6z(t) - 0.8x(t) + u(t) \\ z(t) \leq M\delta(t) \\ z(t) \geq m\delta(t) \\ z(t) \leq x(t) - m(1-\delta(t)) \\ z(t) \geq x(t) - M(1-\delta(t)) \end{cases}$$

□

In general, mixed logical dynamical systems are described through the following linear relations,

$$\begin{cases} x(t+1) &= Ax(t) + B_1u(t) + B_2\delta(t) + B_3z(t) \\ y(t) &= Cx(t) + D_1u(t) + D_2\delta(t) + D_3z(t) \\ E_2\delta(t) &+ E_3z(t) \leq E_1u(t) + E_4x(t) + E_5 \end{cases} \tag{3.28}$$

where

$$x = \begin{bmatrix} x_c \\ x_l \end{bmatrix}, \quad x_c \in \mathbb{R}^{n_c}, \quad x_l \in \{0,1\}^{n_c}, \quad n = n_c + n_l$$

is the state of the system, with the x_c components being continuous and the x_l components being 0 or 1. The input u space is partitioned similarly. The auxiliary logical and continuous variables are represented by $\delta \in \{0,1\}^{r_l}$ and $z \in \mathbb{R}^{r_c}$, respectively. Furthermore, the

optimal control problem for PWA system can be rewritten as an optimal control problem for MLDS:

$$\begin{aligned} & J(U_N, x_0) \\ = \quad & \min_{U_N} \|Px_N\|_2 + \sum_{k=0}^{N-1} \{\|Q_1 u_k\|_2 + \|Q_2 \delta_k\|_2 + \|Q_3 z_k\|_2 + \|Q_4 x_k\|_2\} \\ s.t. \quad & \begin{cases} x_{k+1} = Ax_k + B_1 u_k + B_2 \delta_k + B_3 z_k \\ E_2 \delta_k + E_3 z_k \leq E_1 u_k + E_4 x_k + E_5 \\ x_N \in \chi_f \end{cases} \end{aligned}$$

Next we show how to transform the optimal control problem for MLDS into a mixed-integer programming problem. In fact, the translation for a quadratic performance index is simply obtained by substituting the state update equation, namely

$$x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j (B_1 u_{k-1-j} + B_2 \delta_{k-1-j} + B_3 z_{k-1-j})$$

and the optimization vector

$$\zeta = \{u_0, \cdots, u_{N-1}, \delta_0, \cdots, \delta_{N-1}, z_0, \cdots, z_{N-1}\}$$

Then, the above optimal control problem can be formulated as a mixed-integer quadratic programming problem (MIQP)

$$\begin{aligned} & \min_{\zeta} \zeta^T H_1 \zeta + \zeta^T H_2 x_0 + x_0^T H_3 x_0 + c_1^T \zeta + c_2^T x_0 + c \\ s.t. \quad & G\zeta \leq W + Sx_0 \end{aligned}$$

where H_1, H_2, H_3, c_1, c_2, G, W, S are matrices of suitable dimensions.

Similarly, if the cost function is measured by 1-norm or ∞-norm performance indices, it is also passible to transform the finite horizon optimal control problem into a mixed-integer linear programming (MILP) [Borrelli et al., 2005, Borrelli, 2003].

Given a value of the initial state $x(0)$, the MILP or MIQP can be solved to obtain the optimizer $\zeta^*(x(0))$ and therefore the optimal input $U_N^*(x(0))$. There exists a method called multi-parametric programming [Borrelli et al., 2005, Borrelli, 2003] that can be used to efficiently compute optimal control law, which is in the form of a piecewise affine state feedback.

3.4 Notes and Further Reading

Due to space limit, we only give a very brief introduction of the results in the stability, stabilization and optimal control for switched systems and piecewise affine systems with emphasis on the computation methods. In addition, the reachability, controllability and observability for switched and piecewise affine systems have been extensively studied in the literature, see e.g., [Sun et al., 2002, Xie and Wang, 2003, Bemporad et al., 2000] and the references therein.

The MLF Theorem 3.10 is based on the results in [Branicky, 1998], where a proof for the theorem can be found. It is worth pointing out that there exist different versions of the MLF theorem, which could be less conservative than the version we presented above. For example, the Lyapunov-like function for the q-th subsystem may be not monotonically decreasing when the q-th subsystem is activated. The value may actually increase, but the hybrid system remains stable provided that the increase of the Lyapunov-like function is bounded by a continuous function, see e.g., [Ye et al., 1998]. Surveys on the switched systems stability and stabilization can be found in [Liberzon and Morse, 1999, DeCarlo et al., 2000, Liberzon, 2003, Shorten et al., 2007, Lin and Antsaklis, 2009, Goebel et al., 2009].

The descriptions of switched quadratic Lyapunov function and LMI conditions are based on [Daafouz et al., 2002], and the synthesis approach based on piecewise quadratic Lyapunov function is based on the work [Pettersson, 2003]. Notice that the stability analysis based on piecewise quadratic Lyapunov functions is sufficient only and could be conservative. To reduce the possible conservativeness, a new kind of polynomial Lyapunov functions was introduced and investigated for the stability analysis of hybrid systems. The computation of such polynomial Lyapunov functions can be efficiently performed using convex optimization, based on the sum of squares (SOS) decomposition of multivariate polynomials [Prajna and Papachristodoulou, 2003]. It is also possible to use SOS techniques together with the $\mathcal{S}$-procedure to construct piecewise polynomial Lyapunov functions [Papchristodoulou and Prajna, 2009], with each polynomial as an SOS while incorporating the state constraints, so to generalize piecewise

quadratic Lyapunov functions.

We only reviewed sufficient conditions for the existence of stabilizing switching signals for a given collection of linear systems. A more elusive problem has been the necessity part of the switching stabilizability problem, and a particularly challenging part has been the problem of necessary and sufficient conditions for switching stabilizability. In [Lin and Antsaklis, 2007], a necessary and sufficient condition was proposed for the existence of a switching control law (in static state feedback form) for asymptotic stabilization of continuous-time switched linear systems.

The literature on the optimal control of hybrid systems is very rich, and our treatment is very biased and superficial. Most efforts have been devoted to the extensions of the maximum principle and dynamical programming techniques to hybrid systems, see e.g., [Branicky et al., 1998, Sussmann, 1999, Cassandras et al., 2001, Shaikh and Caines, 2007]. The embedding optimization approaches were proposed in [Bengea and DeCarlo, 2005], where readers may find proofs and more examples. The idea of embedding or converting the switched system into a large family of system that is more suitable for traditional numerical or theoretical methods has been pursued by other researchers as well, see e.g., [Das and Mukherjee, 2008, Mojica-Nava et al., 2013]. Two-stage optimization mainly follows [Xu and Antsaklis, 2004]. Two stage optimization approach has also been adopted in [Egerstedt et al., 2006, Gonzalez et al., 2010, Wardi and Egerstedt, 2012]. Interested readers may refer to these work and the references therein for recent developments along this line.

The discussion on optimal control of piecewise affine systems follows the results in [Rantzer and Johansson, 2000, Johansson, 2003a], while the mixed logic dynamical systems and parametric optimization results are based on [Bemporad and Morari, 1999, Bemporad et al., 2002, Borrelli, 2003]. It is also worthy pointing out that there are free software tool boxes available to support the computation of the optimal control solutions of switched and hybrid systems. For example, Multi-Parametric Toolbox (MPT) [Borrelli, 2003] is a free MATLAB toolbox for design, analysis and deployment of optimal con-

trollers for constrained linear, nonlinear and hybrid systems, and Convex Dynamic Programming (CDP) Tool [Hedlund and Rantzer, 1999] is another MATLAB toolbox developed to solve hybrid optimal control problems. Interested readers may refer to the survey papers [Xu and Antsaklis, 2003, Zhu and Antsaklis, 2011] and their references.

4

Verification of Hybrid Systems

The verification of real-time code implemented in embedded systems is a very important problem, as many of these systems, such as autopilot systems and medical devices, are safety critical and need guarantees of their proper operations [Edwards et al., 2001]. These embedded systems are interacting with the physical world, and continuous variables, such as time clocks, have to be taken into consideration. Hence, modeling the system as a hybrid system becomes a natural choice. A typical question is whether certain properties, such as safety (e.g., bad things wont happen), liveness (e.g., good things eventually happen) and other properties, hold true or not for the given hybrid system model. These are verification problems. However, it is not straightforward to apply traditional formal methods for verification, such as model checking [Clarke et al., 1999] and deductive verification [Kaufmann et al., 2000], to hybrid systems since these methods were originally developed for circuits and communication protocols and usually require extensive search of all reachable states. However, this is not possible as the states in hybrid systems are uncountable. Motivated by this challenge, significant research activities have been devoted to the verification problems for hybrid systems. In this

chapter, we focus on formal methods for verification based on model checking. In particular, we mainly emphasize the abstraction based approaches, where the basic idea is to obtain an equivalent abstracted model (with finite states) so to perform the verification using traditional approaches. Hence, we start this chapter with a brief overview of traditional model checking for finite state systems. Other verification methods and software tools will be briefly reviewed at the end of this chapter.

The rest of this chapter is organized as follows. Section 4.1 gives a brief introduction of temporal logic that is interpreted over labeled transition systems. Both finite automata and hybrid automata can be seen as labeled transition systems. Model checking approaches are briefly reviewed in Section 4.1 for finite state transition systems with respect to temporal logic, and then are extended to the case of infinite state transition systems. The main idea behind the extension is to obtain an equivalent finite transition system, called an abstraction, for an infinite transition system under consideration. The equivalence is in the sense of bisimulation, which is formally defined in Section 4.2. Then, we consider the verification problem for hybrid systems. In particular, timed automata (Section 4.3), multirate automata and rectangular automata (Section 4.4) are introduced and their finite quotient transition systems are obtained. Unfortunately, it is known that even a slight generalization of the multirate automata or rectangular automata could make the reachability problem undecidable [Henzinger et al., 1995, 1998], i.e., it becomes equivalent to the halting problem. Hence many efforts have been devoted to developing sophisticated techniques drawn from optimal control, game theory, and computational geometry to calculate or approximate the reachable sets for various classes of hybrid systems. We will briefly review these efforts and available software tools at the end of this chapter.

4.1 Model Checking

Model checking is a method to verify algorithmically whether a model, which is usually derived from a hardware or software de-

sign, satisfies certain properties. The properties we are interested in are more in the dynamical than in static sense. For example, whether the statement "the machine is busy" is true or not will vary in time and depends on the current state of the dynamical system. Sometimes the statement is true, and sometimes the statement is false, but the statement is never true and false simultaneously. Also, one may be interested in checking whether a statement eventually becomes true or not, e.g., "proposal gets approved." Formally, these properties can be formulated in temporal logic, and the model is expressed as a transition system. In this section, we will give a brief tutorial on the model checking for finite transition systems. Our treatment mainly follows [Baier and Katoen, 2008]. Interested readers may also refer to [Clarke et al., 1999, Baier and Katoen, 2008] for a comprehensive and detailed discussion on model checking.

4.1.1 Transition Systems

Transition systems are graph models that describe the evolution of the states under the action of transitions.

Definition 4.1. A *transition system* T is a four tuple $T = (S, S_0, U, \rightarrow)$ defined by

- A set of states S;
- A set of initial states $S_0 \subseteq S$;
- A set of actions U;
- A transition relation $\rightarrow \subseteq S \times U \times S$.

A transition system is called finite when the state set S and the action set U contain only a finite number of elements. Clearly, a finite automaton can be cast as a transition system with finite states. A transition system may have infinite number of states, and can be used to represent a large class of dynamical systems. For example, hybrid automata can be rewritten in the form of transition systems.

Example 4.1. A hybrid automaton $H = \{Q, X, f, Init, Inv, E, G, R\}$, can be represented as a transition system $T_H = (S, S_0, U, \longrightarrow)$, where

- $S = Q \times X$;
- $S_0 = Init$;
- $U = E \cup \mathbb{R}_{>0}$;
- $(q, x) \longrightarrow^u (q', x')$ if one of the following condition holds

    1. Discrete transition: when $u = (q, q')$, $x \in G(q, q')$ and $x' \in R(q, q', x)$;

    2. Continuous transition: when $u \in \mathbb{R}_{>0}$, $q = q'$ and there exists a solution $(q, x(t)) \in Inv$ for $0 \leq t \leq u$ such that $\dot{x}(t) = f(q, x(t))$ for $0 \leq t \leq u$, and $x(0) = x$, $x(u) = x'$.

Hence, transition systems provide us with a very general framework for dynamical systems. The dynamical behavior of a transition system is conveniently described by the strings of its state evolution. Formally, we have the following definitions and notations.

Definition 4.2. A string $\alpha \in S^*$ ($\alpha \in S^\omega$) is a *run* of transition system $T = (S, S_0, U, \rightarrow)$ if

1. $\alpha(1) \in S_0$;
2. there exists a string $\beta \in U^*$ ($\beta \in U^\omega$) such that $(\alpha(i), \beta(i), \alpha(i+1)) \in \rightarrow$, for $i = 1, \cdots, |\alpha| - 1$ ($i \geq 1$ for $\beta \in U^\omega$).

Note that $|\alpha|$ stands for the length of the run α, which potentially contains infinite number of transitions, i.e., $|\alpha| \in \mathbb{N} \cup \{\omega\}$. Here $\mathbb{N}$ denotes the set of natural numbers and ω stands for infinity. For $i < |\alpha|$, the i-th state of α, written as $\alpha(i)$, is the state s_i reached after i transitions. A complete execution is a run which is *maximal*, that is, which cannot be extended. It is either infinite, or it ends in a state s_n out of which no transition is defined. If the second case happens, we call it a *deadlock*.

Consider a transition system $T = (S, S_0, U, \rightarrow)$. For a particular state $s \in S$ and action $a \in U$, the set of successor states of s by action a are given by $post_a(s) = \{s' \in S | (s, a, s') \in \rightarrow\}$. The *successor states* of s in T for all possible actions is $post(s) = \bigcup_{a \in U} post_a(s)$. Accordingly,

the set of successor states for a set $P \subseteq S$ can be defined by $post(P) = \bigcup_{s \in P} post(s)$. Similarly, we can define $pre_a(s) = \{s' \in S | s \in post_a(s')\}$, $pre(s) = \bigcup_{a \in U} pre_a(s)$, and $pre(P) = \bigcup_{s \in P} pre(s)$.

Intuitively, $pre(P)$ describes the set of states that can be transited to a state in P within one step transition, while $post(P)$ contains the states that can be reached by a state in P within one step transition. The set of states that are accessible from P in two transition steps can be characterized by $post(post(P))$, and denoted as $post^2(P)$. Inductively, one can denote the states that are accessible from P in $n \in \mathbb{N}$, $n \geq 0$ transition steps as $post^n(P)$ that can be calculated recursively by $post^0(P) = P$, $post^n(P) = post(post^{n-1}(P))$. Then, the states that can be accessible from P are the union of all $post^i(P)$ for $i \geq 0$, that is $post^*(P) = \bigcup_{i \in \mathbb{N},\, i \geq 0} post^i(P)$. In particular, $post^*(Q_0)$ is the set of *reachable states* for the transition system T and is denoted as $Reach(T)$. Similar definitions can be provided for $pre^n(P)$ and $pre^*(P)$.

Actions can be seen as inputs, and we can also introduce outputs for transition systems. Instead, we call them labels, which associate the states of a transition system with properties that hold true for the corresponding states. The properties of interest are denoted as symbols p_i, say p_1 ="the machine is busy," p_2 ="the machine is broken" and so on. The collection of such symbols (assumed to be finite) forms a set, denoted as $\mathcal{P} = \{p_1, p_2, \cdots\}$ and called an atomic proposition set. A labeled transition system is a transition system with all its states being labeled with true or false for *atomic propositions* in $\mathcal{P}$.

Definition 4.3. A *labeled transition system* is a tuple (T, l), where $T = (S, S_0, U, \rightarrow)$ is a transition system and $l : S \rightarrow 2^{\mathcal{P}}$ is a label function that assigns each state s in T a subset of predicates $l(s) \subseteq \mathcal{P}$ satisfied by the state s.

Given a finite run α of the transition system T, we can define a *trace* generated from the labeled transition system (T, l) corresponding to the run α as a string $\gamma \in \left(2^{\mathcal{P}}\right)^*$, where $\gamma(i) = l(\alpha(i))$. The collection of all finite traces that can be generated by the labeled transition system (T, l) is called the trace generated by (T, l), denoted as $\mathcal{T}(T, l)$. Note that the above definitions can be extended to the case where α is of

infinite length. Then, γ is an infinite trace as defined above, i.e., $\gamma \in \left(2^{\mathcal{P}}\right)^{\omega}$, and the collection of all such infinite length traces is called the *ω-trace* generated by (T, l), denoted as $\mathcal{T}_{\omega}(T, l)$.

4.1.2 Linear Temporal Logic

Next, we need to introduce a formal way to construct more complex expressions describing properties of states in a labeled transition system (T, l), whose truth value can vary with resect to time. For such a purpose, temporal logic was proposed [Pnueli, 1977]. Temporal logic is a formalism for describing properties of sequences of states as well as tree structures of states. There are many variations of temporal logic, and interested readers may refer to the survey paper [Emerson, 1990] and books [Clarke et al., 1999, Baier and Katoen, 2008]. We will first introduce linear temporal logic and then computation tree logic. Their relationship will be demonstrated using examples.

Linear Temporal Logic (LTL) is an extension of propositional logic geared to reasoning about infinite sequences of states. Formulas of LTL are built from a set of atomic propositions, like "the machine is busy," and are closed under the application of Boolean connectives, such as conjunction, disjunction and negation, and temporal operators. In particular, the following *temporal operators* are used for describing the properties along a specific path:

- $\circ$ ("next state"): requires that a property hold in the next state of the path. Let's use φ to denote the property of interest, then $\circ\varphi$ can be illustrated as
  $\bullet \longrightarrow \bullet^{\varphi} \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet$.

- $\Diamond$ ("eventually"): used to assert that a property will hold at some future state on the path. For example, the expression $\Diamond\varphi$ can be illustrated as
  $\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet^{\varphi} \longrightarrow \bullet$.

- $\Box$ ("always"): specifies that a property holds at every state on the path. For example, $\Box\varphi$ can be illustrated as $\bullet^{\varphi} \longrightarrow \bullet^{\varphi} \longrightarrow \bullet^{\varphi} \longrightarrow \bullet^{\varphi} \longrightarrow \bullet^{\varphi}$.

- $\sqcup$ ("until"): used to combine two properties. The combined property holds if there is a state on the path where the second property holds, and at every preceding state on the path, the first property holds. For example, the expression $\varphi_1 \sqcup \varphi_2$ can be illustrated as $\bullet^{\varphi_1} \longrightarrow \bullet^{\varphi_1} \longrightarrow \bullet^{\varphi_2} \longrightarrow \bullet \longrightarrow \bullet$.

We have the following relations among the above operators, where φ denotes a temporal logic specification: $\Diamond\varphi = \texttt{true} \sqcup \varphi$ and $\Box\varphi = \neg\Diamond\neg\varphi$. Therefore, one can just use $\circ$ and $\sqcup$ to express the rest of temporal operators. These temporal operators can be nested with Boolean connectives to generate more complex temporal logic specifications.

Example 4.2. The LTL formula $\Box\Diamond\varphi$ is true for traces (generated from a labeled transition system) that satisfy φ infinitely often, e.g., $\Box\Diamond\texttt{hungry}$. The formula $\Diamond\Box\varphi$ means that φ will become true eventually and holds for ever, e.g., $\Diamond\Box\texttt{battery dies}$. □

Let's see a more complicated example.

Example 4.3. To specify the traffic light behavior, we define a set of atomic propositions $\mathcal{P} = \{\texttt{red}, \texttt{green}, \texttt{yellow}\}$. The specification

$$\varphi = \Box(\texttt{red} \rightarrow \circ(\texttt{red} \sqcup (\texttt{yellow} \wedge \circ(\texttt{yellow} \sqcup \texttt{green}))))$$

describes the traffic light behavior. It basically requests that if the traffic light is red, it cannot immediately become green and has to be in yellow for a while. □

Now, let's formally define the syntax of LTL formulas.

Definition 4.4. Linear Temporal Logic (LTL) formulas are recursively defined from predicates in $\mathcal{P}$ according to the following rules.

1. $\texttt{true}$, $\texttt{false}$, and p_i are LTL formulas for all $p_i \in \mathcal{P}$;
2. if φ_1 and φ_2 are LTL formulas,
   then $\varphi_1 \wedge \varphi_2$ and $\neg\varphi_1$ are LTL formulas;
3. if φ_1 and φ_2 are LTL formulas,
   then $\circ\varphi_1$ and $\varphi_1 \sqcup \varphi_2$ are LTL formulas

An LTL formula φ is interpreted over infinite sequences of sets of propositions, called a word $s = \mathcal{P}_1\mathcal{P}_2\mathcal{P}_3 \cdots \in \left(2^{\mathcal{P}}\right)^{\omega}$, where each $\mathcal{P}_i$ is a subset of $\mathcal{P}$. The satisfaction of a formula φ at position $t \in \mathbb{N}$ can be defined as follows.

Definition 4.5. A word $s \in \left(2^{\mathcal{P}}\right)^{\omega}$ satisfies an LTL formula φ at t denoted by $s(t) \models \varphi$ if the following hold:

1. if $\varphi = p$, then $s(t) \models \varphi$ iff $p \in s(t)$ and $s(t) \models \neg\varphi$ iff $p \notin s(t)$;
2. if $\varphi = \varphi_1 \wedge \varphi_2$ then $s(t) \models \varphi$ iff $s(t) \models \varphi_1$ and $s(t) \models \varphi_2$;
3. if $\varphi = \circ\varphi_1$ then $s(t) \models \varphi$ iff $s(t+1) \models \varphi_1$;
4. if $\varphi = \varphi_1 \sqcup \varphi_2$ then $s(t) \models \varphi$ iff $\exists t' \geq t$ such that for all $k \in [t, t')$, $s(k) \models \varphi_1$ and $s(t') \models \varphi_2$.

A word $s \in \left(2^{\mathcal{P}}\right)^{\omega}$ satisfies φ if and only if $s(1) \models \varphi$. LTL formulas can be evaluated over traces generated from a labeled transition system. A labeled transition system (T, l) satisfies an LTL formula φ, denoted as $(T, l) \models \varphi$, if all ω-traces generated by (T, l) satisfy φ. If we denote all words satisfying the LTL formula φ as $\mathcal{W}(\varphi)$, then $(T, l) \models \varphi$ if and only if all ω-traces generated from (T, l) are contained in $\mathcal{W}(\varphi)$, i.e., $\mathcal{T}_\omega(T, l) \subseteq \mathcal{W}(\varphi)$.

4.1.3 LTL model checking

The LTL model checking problem is to determine whether a given labeled transition system (T, l) satisfies an LTL formula φ. There exist automated approaches to LTL model checking, and its basic idea is to reduce the model checking problem to an inclusion problem between automata [Vardi, 1996]. In particular, *Büchi automata* are employed. A Büchi automaton is an extension of a finite automaton to accept an infinite input sequence. Büchi automata have the same structure as finite automata and is defined as a tuple $(Q, Q^0, \Sigma, \delta, F)$, but the accepting conditions for a run is different. In Büchi automata, a run is accepting if it visits the marked states infinitely often. The words corresponding to all accepting runs of are called languages accepted by the Büchi automaton.

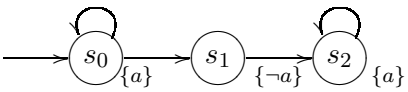
To do LTL model checking based on automata theory, we first convert the labeled transition system (T, l) to a Büchi automaton B_T, such that the languages accepted by the Büchi automaton B_T, denoted as $\mathcal{L}_\omega(B_T)$, coincide with the set of ω-traces generated from (T, l), i.e., $\mathcal{L}_\omega(B_T) = \mathcal{T}_\omega(T, l)$.

Next, we take a negation of the specification, i.e., $\neg\varphi$, and translate $\neg\varphi$ into an equivalent Büchi automaton, denoted as $B_{\neg\varphi}$. The equivalence is in the sense that $\mathcal{L}_\omega(B_{\neg\varphi})$ is exactly the set of paths satisfying the formula $\neg\varphi$, that is $\alpha \models \neg\varphi$ if and only if $\alpha \in \mathcal{L}_\omega(B_{\neg\varphi})$. The basic idea of the translation is to use the collections of all sub-formulas as the state of the Büchi automaton, and the state should contain exactly those sub-formulas that hold true for all runs starting from this state. The obtained Büchi automaton could be very large in the sense that the size of its states could be of an exponential growth compared with the length of the formula, see e.g., [Baier and Katoen, 2008].

After obtaining B_T and $B_{\neg\varphi}$, the next step is to build a Büchi automaton B such that $\mathcal{L}_\omega(B) = \mathcal{L}_\omega(B_{\neg\varphi}) \cap \mathcal{L}_\omega(B_T)$, and then check the emptiness of $\mathcal{L}_\omega(B)$. The rationale behind this procedure is the following simple argument. Since $\mathcal{L}_\omega(B) = \mathcal{L}_\omega(B_{\neg\varphi}) \cap \mathcal{L}_\omega(B_T)$, represents all the runs in B_T starting from q_0 that satisfy $\neg\varphi$, i.e., do not satisfy φ. Therefore, $\mathcal{L}_\omega(B) = \emptyset$ implies that there is no execution generated from T that violates the property φ. In other words, φ holds true for all executions generated from the plant.

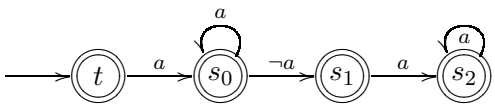
We use the following simple example from [Baier and Katoen, 2008] to illustrate the basic idea of LTL model checking.

Example 4.4. Consider $\mathcal{P} = \{a, \neg a\}$ and the following labeled transition system (T, l):

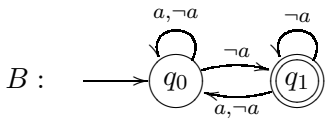


Our task is to check whether (T, l) satisfies the LTL $\varphi = \Diamond\Box a$, i.e., any trace generated from (T, l) eventually holds a forever.

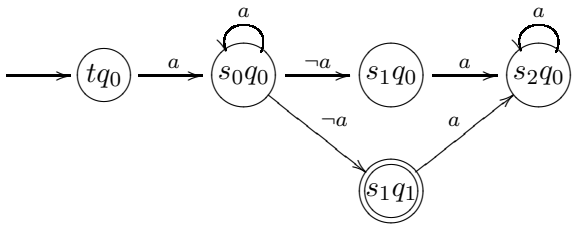
First, (T, l) is converted to the following Büchi automaton, B_T:



Next, we take negation of the formula $\neg\varphi = \neg\Diamond\Box a$ and construct $B_{\neg\varphi}$, which is generated using LTL2BA software [Gastin and Oddoux, 2001].



Then we compute the intersection of B_T and $B_{\neg\varphi}$ as follows



It is clear that the intersection automaton accepts empty infinite runs, i.e., $\mathcal{L}_\omega(B) = \mathcal{L}_\omega(B_{\neg\varphi}) \cap \mathcal{L}_\omega(B_T) = \emptyset$. Hence, we can conclude that $(T, l) \models \varphi$. □

4.1.4 Computation Tree Logic

Linear temporal logic is used to describe properties of sequences of states, and a transition system satisfies an LTL formula if all its paths satisfy the specification. However, in some applications we may be interested in the existence of at least one successful/false path [Clarke et al., 1986]. To describe the branching time structure starting at a particular state, two *path quantifiers* are used:

- $\forall$: for all paths and
- $\exists$: for some paths.

These two quantifiers are used in a particular state to specify that all the paths or some of the paths starting from that state satisfy certain property. Based on these quantifiers and temporal operators, we can define computation tree logic formulas [Clarke et al., 1999, Baier and Katoen, 2008].

Definition 4.6. Let $\mathcal{P}$ be a finite set of atomic propositions. Computation Tree Logic (CTL) formulas are state formulas recursively defined from predicates in $\mathcal{P}$ according to the following rules.

1. `true`, `false`, and p_i are state formulas for all $p_i \in \mathcal{P}$;
2. if ϕ_1 and ϕ_2 are state formulas,
   then $\phi_1 \wedge \phi_2$ and $\neg\phi_1$ are state formulas;
3. if ϕ_1 and ϕ_2 are state formulas,
   then $\circ\phi_1$ and $\phi_1 \sqcup \phi_2$ are path formulas;
4. if φ is a path formula,
   then $\exists\varphi$ and $\forall\varphi$ are state formulas;

CTL uses atomic propositions as its building blocks to make statements about the states of a system. CTL then combines these propositions into formulas using logical operators and temporal operators. Next, we are going to show that using only three operators $\exists\circ$, $\exists\sqcup$, and $\exists\Box$ is sufficient.

- $\forall \circ \phi = \neg\exists \circ (\neg\phi)$
- $\exists\Diamond\phi = \exists[\texttt{true} \sqcup \phi]$
- $\forall\Box\phi = \neg\exists\Diamond(\neg\phi)$
- $\forall\Diamond\phi = \neg\exists\Box(\neg\phi)$
- $\forall[\phi_1 \sqcup \phi_2] = \neg\exists[\neg\phi_2 \sqcup (\neg\phi_1 \wedge \neg\phi_2)] \wedge \neg\exists\Box(\neg\phi_2)$

Let's see some examples of CTL formulas [Clarke et al., 1999, Baier and Katoen, 2008]:

- $\exists\Diamond(\texttt{start} \wedge \neg\texttt{ready})$ means that it is possible to get to a state where `start` holds but not `ready`;
- $\forall\Box(\texttt{request} \rightarrow \forall\Diamond\texttt{ack})$ stands for the property that once a request is made it should be acknowledged eventually;
- $\forall\Box(\forall\Diamond\texttt{blink})$ means that everyone blinks infinitely often;

- $\forall\Box(\exists\Diamond\texttt{recover})$ means that it is always possible to recover from an error.

CTL formulas are all state formulas, and are interpreted over states in a labeled transition system. They ask whether there exists a path (or for all paths) from the state satisfying a certain property. Formally, we define satisfaction of CTL formulas as follows.

Definition 4.7. A state $s \in S$ in a labeled transition system (T, l) satisfies a CTL formula ϕ, denoted by $s \models \phi$ if the following hold:

1. if $\phi = p$, then $s \models \phi$ iff $p \in l(s)$ and $s \models \neg\phi$ iff $p \notin l(s)$;
2. if $\phi = \phi_1 \wedge \phi_2$ then $s \models \phi$ iff $s \models \phi_1$ and $s \models \phi_2$;
3. if $\phi = \exists \circ \phi_1$ then $s \models \phi$ iff there exists $s' \in post(s)$ and $s' \models \phi_1$;
4. if $\phi = \forall \circ \phi_1$ then $s \models \phi$ iff for all $s' \in post(s)$, it holds that $s' \models \phi_1$;
5. if $\phi = \exists\phi_1 \sqcup \phi_2$ then $s \models \phi$ iff there exists a string $\alpha \in S^\omega$ such that $\alpha(1) = s$, $\alpha(i) \to \alpha(i+1)$ for $i \in \mathbb{N}$, $\alpha(j) \models \phi_2$ for some $j \geq 1$ and $\alpha(i) \models \phi_1$ for all $1 \leq i < j$;
6. if $\phi = \forall\phi_1 \sqcup \phi_2$ then $s \models \phi$ iff for all strings $\alpha \in S^\omega$ such that $\alpha(1) = s$, $\alpha(i) \to \alpha(i+1)$ for $i \in \mathbb{N}$, $\alpha(j) \models \phi_2$ for some $j \geq 1$ and $\alpha(i) \models \phi_1$ for all $1 \leq i < j$.

For a CTL formula ϕ, we say that a labeled transition system (T, l) satisfies ϕ, denoted as $(T, l) \models \phi$, if and only if for all $s_0 \in S_0$, we have $s_0 \models \phi$.

4.1.5 CTL Model Checking

Once such a temporal logic formula has been specified, the next step is to make sure that the (designed) dynamical system, which can be modeled as a labeled transition system, satisfies the specification. Formally, the CTL model checking problem can be formulated as: Given a labeled transition system (T, l) and a CTL formula ϕ, determine if (T, l) satisfies ϕ.

The basic idea of CTL model checking is to mark each state of T with the set $Sat(s)$ of sub-formulas of ϕ which are true in the state s

[Clarke et al., 1999, Baier and Katoen, 2008]. Initially, $Sat(s)$ is just $l(s)$. The marking process then goes through series of stages. During the ith stage, sub-formulas with $i-1$ nested CTL operators are processed. When a sub-formula is processed, it is added to the labeling of each state in which it is true. Once the algorithm terminates, we will have $(T, l) \models \phi$ if and only if $\phi \in Sat(s_0)$ for all $s_0 \in S_0$.

Note that any CTL formula can be expressed in terms of $\neg$, $\vee$, $\exists\circ$, $\exists\sqcup$, and $\exists\Box$, which is called *Existential Normal Form* (ENF) of CTL. Thus, for the intermediate stages of the algorithm it is sufficient to be able to handle the following cases: ϕ is atomic or ϕ has forms $\neg\phi_1$, $\phi_1 \vee \phi_2$, $\exists \circ \phi_1$, $\exists[\phi_1 \sqcup \phi_2]$, and $\exists\Box\phi_1$.

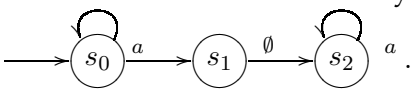
Then, we need to label the states based on the following rules:

- For the case when ϕ is atomic, it has been handled by the labeling function $l(s)$.
- For formulas of the form $\neg\phi_1$, we label those states that are not labeled by ϕ_1.
- For $\phi_1 \vee \phi_2$, we label any state that is labeled either by ϕ_1 or by ϕ_2.
- For $\exists \circ \phi_1$, we label every state that has some successor labeled by ϕ_1.
- For a formula ϕ of the form $\exists[\phi_1 \sqcup \phi_2]$, we first find all states that are labeled by ϕ_2, and then search backwards along a path till an initial state with all the states along the path are labeled with ϕ_1. All such states should be labeled with ϕ.
- The case that ϕ is of the form $\exists\Box\phi_1$ can be handled by decomposition of the graph into nontrivial strongly connected components. A strongly connected components (SCC) C is a maximal subgraph such that every node in C is reachable from every other node in C along a directed path entirely contained within C. C is nontrivial if either it has more than one node or it contains one node with a self-loop. The following steps are taken:

1. First, we obtain a sub-transition system T' with states in S', which are labeled with ϕ_1, by deleting the states where ϕ_1 is not labeled and all links to or from these states.
2. Then, we partition T' into strongly connected components.
3. Next, we find those states that belong to nontrivial components, and then work backwards to find all those states that can be reached by a path in which each state is labeled by ϕ_1. All such states should be labeled with ϕ.

In summary, in order to handle an arbitrary CTL formula ϕ, we successively apply the state labeling algorithm to the sub-formulas of ϕ, starting from the shortest, most deeply nested sub-formulas, and work onwards to include all of ϕ. By proceeding in this manner we guarantee that whenever we process a sub-formula of ϕ, all of its sub-formulas have already been processed. It is known that the computation of CTL model checking is of polynomial complexity [Clarke et al., 1999, Baier and Katoen, 2008].

Example 4.5. Let's revisit the labeled transition system (T, l)



We know that the labeled transition system satisfies the LTL formula $\varphi = \Diamond\Box a$ from Example 4.4. Now, we will check whether it satisfies the CTL formula $\Phi = \forall\Diamond\forall\Box a$.

First, we rewrite Φ in ENF as $\Phi = \neg(\exists\Box(\exists\Diamond(\neg a)))$. From sub-formula inside to the more complicated formula

- $Sat(a) = \{s_0, s_2\}$
- $Sat(\neg a) = \{s_1\}$
- $Sat(\exists\Diamond\neg a) = \{s_0, s_1\}$
- $Sat(\exists\Box(\exists\Diamond\neg a)) = \{s_0\}$
- $Sat(\neg(\exists\Box(\exists\Diamond\neg a))) = \{s_1, s_2\}$.

Hence, $s_0 \notin Sat(\Phi)$, so that the CTL formula $\Phi = \forall\Diamond\forall\Box a$ is not satisfied. □

The difference between the semantics of LTL and CTL is that LTL formulas are interpreted over words, whereas CTL formulas are interpreted over the tree of trajectories generated from a given state of a transition system. CTL and LTL have different expressive power. For example, the CTL formula $\phi = \forall\Box\exists\Diamond p$ cannot be expressed by any LTL formula. Also, it can be shown that the LTL formula $\varphi = \Diamond\Box a$ cannot be expressed by any CTL formula. So, LTL and CTL are incomparable, but LTL and CTL do overlap, as it can be seen from the fact that the CTL formula $\forall\Box p$ is equivalent to the LTL formula $\Box p$. Both LTL and CTL are subsets of CTL*, which is a generalization of CTL by allowing Boolean combinations and nestings of temporal operators. CTL* is also called full branching time logic because of its branching time structure, i.e., at each moment, there may exist alternate courses representing different possible futures. It was proposed as a unifying framework which subsumes both CTL and LTL, as well as a number of other logic systems. Due to space limitation, we will not give details for CTL*, interested readers may refer to [Clarke et al., 1999, Baier and Katoen, 2008] and references therein for further details.

4.2 Bisimulation

Model checking tools face a combinatorial increase of the size of the state-space, commonly known as the state explosion problem. Researchers have developed symbolic algorithms, partial order reduction methods, abstractions and on the fly model checking in order to cope with this problem, see e.g., [Clarke et al., 1999, Baier and Katoen, 2008]. These tools were initially developed to reason about the logical correctness of discrete state systems, but have since been extended to deal with real-time and some other special cases of hybrid systems, see e.g., [Alur et al., 1995, Henzinger et al., 1997].

4.2.1 Simulation Relation

In particular, we focus on abstraction based approaches by obtaining equivalent quotient transition systems that satisfy the same temporal logics. Here the quotient is taken with respect to simulation or bisim-

ulation equivalences [Milner, 1989] as defined for labeled transition systems as below. Here, we follow the notations in [Baier and Katoen, 2008].

Definition 4.8. Let $T_i = (S_i, S_i^0, \rightarrow)$ with $i = 1, 2$ be two transition systems[1], and $l_i : S_i \rightarrow 2^{\mathcal{P}}$ label their states respectively. A relation $R \subseteq S_1 \times S_2$ is said to be a *simulation relationship* from labeled transition system (T_1, l_1) to (T_2, l_2) if the following hold:

1. For any $(s_1, s_2) \in R$, their labels are the same, i.e., $l_1(s_1) = l_2(s_2)$;
2. For any initial state $s_1 \in S_1^0$, there exists $s_2 \in S_2^0$ such that $(s_1, s_2) \in R$;
3. For any pair $(s_1, s_2) \in R$, if $s_1' \in post(s_1)$ in T_1 then there exists $s_2' \in S_2$ such that $s_2' \in post(s_2)$ in T_2 and $(s_1', s_2') \in R$.

Intuitively, a labeled transition system simulates another system if, for every computation in the simulated system, there is a matching (w.r.t. labels) computation in the simulating system. If there exists a simulation relationship R from labeled transition system (T_1, l_1) to (T_2, l_2), we also say that (T_1, l_1) is simulated by (T_2, l_2), or (T_2, l_2) simulates (T_1, l_1), denoted as $(T_1, l_1) \prec_R (T_2, l_2)$, since for any trace in (T_1, l_1) one can find a corresponding equivalent trace in (T_2, l_2). Hence, if $(T_1, l_1) \prec_R (T_2, l_2)$, which also results $\mathcal{T}(T_1, l_1) \subseteq \mathcal{T}(T_2, l_2)$ and $\mathcal{T}_\omega(T_1, l_1) \subseteq \mathcal{T}_\omega(T_2, l_2)$.

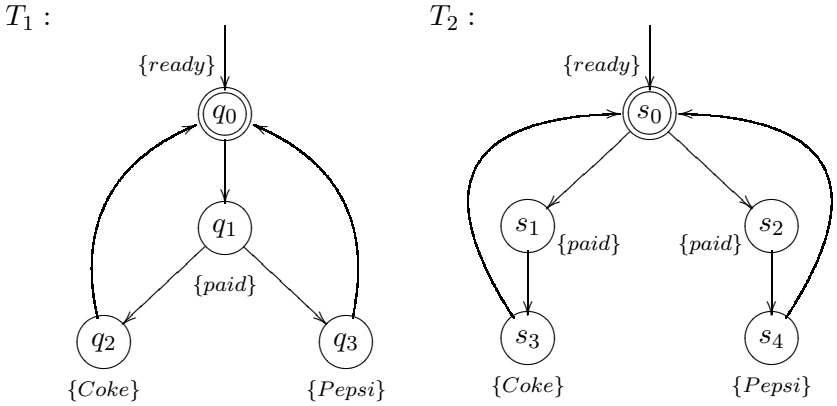
Relation R is said to be *bi-simulation relation* between (T_1, l_1) and (T_2, l_2) if R is a simulation relation from (T_1, l_1) to (T_2, l_2) and R^{-1} is a simulation relation from (T_2, l_2) to (T_1, l_1), i.e., $(T_2, l_2) \prec_{R^{-1}} (T_1, l_1)$. Here $R^{-1} \subseteq S_2 \times S_1$ is a binary relation (between S_2 and S_1) such that $(s_2, s_1) \in R^{-1}$ if and only if $(s_1, s_2) \in R$. If such a bi-simulation relation R exists between (T_1, l_1) and (T_2, l_2), then we say that (T_1, l_1) is *bisimilar* to (T_2, l_2), denoted as $(T_1, l_1) \cong_{R \cup R^{-1}} (T_2, l_2)$. Usually, we only care about the existence of such an R, so we usually omit R and simply write $(T_1, l_1) \prec (T_2, l_2)$ or $(T_1, l_1) \cong (T_2, l_2)$. Since $(T_1, l_1) \cong$

[1]Since we adopt the label based simulation relation and only concern the existence of actions, it is possible to omit the definition of action sets U_i and treat all actions equally.

(T_2, l_2) implies both $(T_1, l_1) \prec (T_2, l_2)$ and $(T_2, l_2) \prec (T_1, l_1)$, so two bisimilar labeled transition systems are trace equivalent, i.e., $(T_1, l_1) \cong (T_2, l_2)$ implies $\mathcal{T}(T_1, l_1) = \mathcal{T}(T_2, l_2)$ and $\mathcal{T}_\omega(T_1, l_1) = \mathcal{T}_\omega(T_2, l_2)$.

Since two bisimilar labeled transition systems generate exactly the same traces, they will satisfy the same collection of LTL formulas. Namely, they cannot be distinguished by LTL formulas, i.e., they are LTL equivalent, denoted as $(T_1, l_1) \equiv_{LTL} (T_1, l_1)$. However, two trace equivalent labeled transition systems may not be bisimilar. The following example illustrates this fact.

Example 4.6. Consider the labeled transition systems T_1 and T_2 over atomic proposition $\mathcal{P} = \{ready, paid, Coke, Pepsi\}$ as follows



For these labeled transition systems,

$$\mathcal{T}_\omega(T_1, l_1) = (ready \cdot paid \cdot (Coke + Pepsi))^\omega = \mathcal{T}_\omega(T_2, l_2).$$

Hence, they are trace equivalent that implies $(T_1, l_1) \equiv_{LTL} (T_1, l_1)$. However, they are not bisimilar, as we are going to show in Example 4.8 later. Actually, these two labeled transition systems really behave differently. T_1 allows customers to choose either Pepsi or Coke after payment, while T_2 refuses to provide any options to customers and picks the next state nondeterministically by itself. To see the difference, we can check a CTL formula $\Phi = \exists \circ (\exists \circ Coke \wedge \exists \circ Pepsi)$, which basically asks whether it is possible to choose after payment. For this particular example we have $(T_1, l_1) \models \Phi$ but $(T_2, l_2) \not\models \Phi$. $\square$

From this example, we can see that it is possible to distinguish two trace equivalent but not bisimilar labeled transition systems using CTL formulas. Actually, the reverse is also true. If two labeled transition systems cannot be distinguished by CTL formulas, then they are bisimilar. Namely, two labeled transition systems are CTL equivalent if and only if they are bisimilar.

Therefore, a CTL model checking problem on a labeled transition system (T_1, l_1) can be conducted on a bisimilar labeled transition system (T_2, l_2). We hope that the bisimilar system (T_2, l_2) has much smaller state space compared to (T_1, l_1), and the computational complexity can be reduced. Furthermore, bisimulation preserves not only CTL properties but also all CTL* properties [Baier and Katoen, 2008, Clarke et al., 1999].

4.2.2 Bisimulation Quotient

To reduce the computational complexity of model checking, we try to reduce the size of the state space of a transition system by clustering all bisimilar (equivalent) states. For such a purpose, we introduce self-bisimulation relation for a labeled transition system first.

Definition 4.9. Consider a labeled transition system (T, l) and a binary relation $R \subseteq S \times S$. The relation R is called a *self-bisimulation relation* for (T, l) if the following hold:

- $\forall (s_1, s_2) \in R : l(s_1) = l(s_2)$.
- $\forall s_1^{'} \in post(s_1), \exists s_2^{'} \in post(s_2)$ with $(s_1^{'}, s_2^{'}) \in R$.
- $\forall s_2^{'} \in post(s_2), \exists s_1^{'} \in post(s_1)$ with $(s_1^{'}, s_2^{'}) \in R$.

States s_1 and s_2 are bisimilar denoted as $s_1 \sim s_2$ if there exists such a binary relation R, defined in the above definition, in (T, l). The bisimulation relation defined above forms an equivalence relation as it is reflexive, symmetric and transitive. Since R is an equivalence relation on S, it therefore induces a partition of the state set into a number of equivalent classes $S = \bigcup_{s \in S} [s]_R$, where $[s]_R$ is a collection if all bisimilar states to s, namely $[s]_R = \{s^{'} \in S | (s, s^{'}) \in R\}$.

It can be easily shown that for any $s, s' \in S$

- $s \in [s]_R$;
- if $s' \in [s]_R$, then $[s]_R = [s']_R$;
- if $s' \notin [s]_R$, then $[s]_R \cap [s']_R = \emptyset$;
- $S = \bigcup_{s \in S} [s]_R$.

Hence, the bisimulation relation R does induce a partition of the state set S. For simplicity, we denote such a partition as $S/_R$, which stands for the quotient space, i.e., the set consisting of all equivalent classes. Based on the quotient space, we can define a quotient transition system as below.

Definition 4.10. Given a labeled transition system (T, l) and a bisimulation relation $R \subseteq S \times S$, the *quotient transition system* can be defined as $T/_R = (S/_R, S^0/_R, \rightarrow_R)$ where $S/_R = \{[s]_R\}_{s \in S}$, the initial states

$$S^0/_R = \{[s]_R \in S/_R : S^0 \cap [s]_R \neq \emptyset\},$$

and for $[s_1]_R, [s_2]_R \in S/_R$, $([s_1]_R, [s_2]_R) \in \rightarrow_R$ if and only if there exist $s_1 \in [s_1]_R$ and $s_2 \in [s_2]_R$ such that $(s_1, s_2) \in \rightarrow$. The new label map $l_R : Q/_R \rightarrow 2^{\mathcal{P}}$ is defined as $l_R([s]_R) = l(s)$.

Note that since if $s' \in [s]_R$ then $l(s') = l(s)$ by definition, the new label map l_R is well-defined. Also, it can be checked that there exists a relation on $S \times S/_R$ that satisfies the bisimulation relation definition. Particularly, we can choose $\mathcal{R} \subseteq S \times S/_R$ with $(s', [s]_R) \in \mathcal{R}$ if and only if $s' \in [s]_R$. Then, we can conclude that $(T, l) \cong_{\mathcal{R} \cup \mathcal{R}^{-1}} (T/_R, l_R)$, see e.g., [Baier and Katoen, 2008, Clarke et al., 1999] for more detailed discussions. Hence, $(T, l) \cong (T/_R, l_R)$. The hope is that the quotient transition systems $T/_R$ has much fewer number of states compared to the original transition system T, so the model checking problem can be significantly simplified. This idea is used for timed automata model checking in the next section.

4.2.3 Computing Bisimulations

In the previous subsection, we obtain a quotient transition system $T/_R$, which is bisimilar to the original transition system and is called

a *bisimulation quotient* of T. The crucial property of bisimulation is that for every equivalence class $P \in S/_R$, the predecessor region $pre(P)$ is a union of equivalence classes. Therefore, if $P_1, P_2 \in S/_R$, then $pre(P_1) \cap P_2$ is either the empty set or all of P_2.

Finding an equivalent relation is equivalent to finding the equivalent classes in $S/_R$. Along this line the following algorithm from [Baier and Katoen, 2008] tries to determine a partition of S so that the corresponding equivalent relation is a bisimulation.

Algorithm 4.1 (Bisimulation Algorithm)**.**
Initialization: $\{(s_1, s_2)|l(s_1) = l(s_2)\} = S/_R$
Refine:
while $\exists P, P' \in S/_R$ such that $P \cap pre(P') \neq P$ and $P \cap pre(P') \neq \emptyset$ **do**
$P_1 = P \cap pre(P')$,
$P_2 = P \setminus pre(P')$
$S/_R = (S/_R \setminus \{P\}) \cup \{P_1, P_2\}$;
end while
return $S/_R$

If the algorithm terminates within a finite number of iterations of the loop, then there is a finite bisimulation quotient, and the algorithm returns a finite partition of the state space which is the coarsest bisimulation (i.e., the bisimulation with the fewest equivalence classes).

This is a pseudo algorithm. Implementation and termination for general transition systems are not obvious. For finite state systems, one can implement the algorithm and guarantee that it terminates because we can enumerate the states for the finite state system.

Example 4.7. Consider the labeled transition system (T_1, l_1) in Example 4.6. For this example, the state is $S = \{s_0, s_1, s_2, s_3, s_4\}$. As the initial partition, we obtain $S/_R = \{\{s_0\}, \{s_1, s_2\}, \{s_3\}, \{s_4\}\}$ that is consistent with the labels. Then, apply the bisimulation algorithm and consider $P = \{s_3\}$ and $P^{'} = \{s_3\}$ from $S/_R$. Calculation shows that $pre(P') = \{s_1\}$. So $P_1 = P \cap Pre(P') = \{s_1\}$ and $P_2 = P \setminus Pre(P') = \{s_2\}$. Then, we update the partition of S as $S/_R = \{\{s_0\}, \{s_1\}, , \{s_2\}, \{s_3\}, \{s_4\}\}$, which is trivial as it contains all the individual states. The algorithm stops and derives a quotient transition

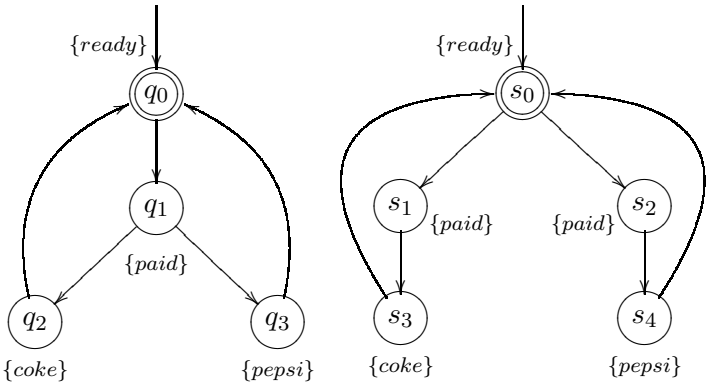
system of T_1 as T_1 itself. □

The bisimulation algorithm can also be used to check the bisimilarity of two transition systems (T_1, l_1), (T_2, l_2) with mutually exclusive states, i.e., $S_1 \cap S_2 = \emptyset$. For such a purpose, the following method can be used.

- Compose (T_1, l_1) and (T_2, l_2) as $T_1 \oplus T_2 = (S_1 \cup S_2, \rightarrow_1 \cup \rightarrow_2, S_1^0 \cup S_2^0, l)$, where $l(s) = l_1(s)$ for $s \in S_1$ and $l(s) = l_2(s)$ for $s \in S_2$.
- Use bisimulation algorithm to calculate the quotient of $(T_1 \oplus T_2, l)$, and check whether $\forall s_1^0 \in S_1^0, \exists s_2^0 \in S_2^0$, such that $s_1^0 \sim s_2^0$, and whether $\forall s_2^0 \in S_2^0, \exists s_1^0 \in S_1^0$, such that $s_2^0 \sim s_1^0$. If yes, then $(T_1, l_1) \cong (T_2, l_2)$. Otherwise, $(T_1, l_1) \ncong (T_2, l_2)$

We illustrate the idea using the vending machine example.

Example 4.8. Consider the labeled transition systems (T_1, l_1) and (T_2, l_2) in Example 4.6. To check whether $(T_1, l_1) \cong (T_2, l_2)$, we first compose these two labeled transition systems together as $(T_1 \oplus T_2, l)$:



Then, we run the bisimulation algorithm on the composed system with the initial partition as $S/_R = \{\{q_0, s_0\}, \{q_1, s_1, s_2\}, \{q_2, s_3\}, \{q_3, s_4\}\}$

Consider $P = \{q_1, s_1, s_2\}$ and $P' = \{q_2, s_3\}$. We have $Pre(P') = \{q_1, s_1\}$, so $P \cap Pre(P') = \{q_1, s_1\} \neq P$ and $P \cap Pre(P') \neq \emptyset$, so we refine the partition as $S/_R = \{\{q_0, s_0\}, \{q_1, s_1\}, \{s_2\}, \{q_2, s_3\}, \{q_3, s_4\}\}$.

Next, consider $P = \{q_1, s_1\}$ and $P' = \{q_3, s_4\}$. We have $Pre(P') = \{q_1, s_2\}$, so $P \cap Pre(P') = \{q_1\} \neq P$ and $P \cap Pre(P') \neq \emptyset$, so we refine the partition as $S/_R = \{\{q_0, s_0\}, \{q_1\}, \{s_1\}, \{s_2\}, \{q_2, s_3\}, \{q_3, s_4\}\}$.

Next, consider $P = \{q_0, s_0\}$ and $P' = \{q_1\}$. We have $Pre(P') = \{q_0\}$, so $P \cap Pre(P') = \{q_0\} \neq P$ and $P \cap Pre(P') \neq \emptyset$, so we refine the partition as
$S/_R = \{\{q_0\}, \{s_0\}, \{q_1\}, \{s_1\}, \{s_2\}, \{q_2, s_3\}, \{q_3, s_4\}\}$.

The problem now is that there does not exist an initial state in T_2 which is in the same equivalent class of q_0, so we terminate the algorithm and conclude that (T_1, l_1) and (T_2, l_2) are not bisimilar. □

In the next sections, we will use the idea of bisimulation quotient transition system and model checking techniques to investigate the verification problems for timed and hybrid automata.

4.3 Timed Automata

Timed automata were introduced in [Alur and Dill, 1994] as formal models for real time programs. Timed automata have been successfully used in modeling, analysis and design of manufacturing systems [Asarin and Maler, 1999], scheduling and robotic systems [Quottrup et al., 2004]. In this subsection, we first formally define timed automata and show them to be a special case of hybrid automata. Then, the model checking of real-time properties for timed automata is discussed by obtaining a finite bisimulation quotient transition system of timed automata. The development mainly follows [Baier and Katoen, 2008].

4.3.1 Timed Automata

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

Definition 4.11. The set, $\Phi(\mathbb{C})$, of *clock constraints* of $\mathbb{C}$, is a set of finite logical expressions defined inductively by $\delta \in \Phi(\mathbb{C})$ if:

$$\delta := (x_i \leq c) | (x_i < c) | (x_i \geq c) | (x_i > c) | \delta_1 \wedge \delta_2,$$

where $\delta_1, \delta_2 \in \Phi(\mathbb{C})$, $x_i \in \mathbb{C}$ and $c \geq 0$ is a rational number.

For example, let $\mathbb{C} = \{x_1, x_2\}$, then the clock constraints $\Phi(\mathbb{C})$ may consist of the following logical expressions:

- $(x_1 \leq 3)$, $(x_2 \geq 1)$, $(x_1 > 3)$, $(x_2 > 1) \in \Phi(\mathbb{C})$; also
- $(x_2 = 1) \in \Phi(\mathbb{C})$, since $(x_2 = 1) \Leftrightarrow (x_2 \leq 1) \wedge (x_2 \geq 1)$;
- $(1 \leq x_2 \leq 3) \in \Phi(\mathbb{C})$, since $(1 \leq x_2 \leq 3) \Leftrightarrow (x_2 \geq 1) \wedge (x_2 \leq 3)$;

But, the expression $(x_1 \leq x_2) \notin \Phi(\mathbb{C})$. The *atomic clock constraints*, denoted as $\mathcal{ACC}(\mathbb{C})$, over a set of clock variables $\mathbb{C}$ are those clock constraints in the form of $x_i \geq c$, $x_i > c$, $x_i \leq c$ or $x_i < c$, where $x_i \in \mathbb{C}$ and $c \geq 0$ is a rational number. In addition, $|\mathbb{C}|$ denotes the number of clocks. Then, the clock values can be seen as vectors in $\mathbb{R}^{|\mathbb{C}|}$ or in $\mathbb{R}^{|\mathbb{C}|}_{\geq 0}$ as clocks cannot be negative.

Definition 4.12. [Alur and Dill, 1994] A *timed automaton* A is a tuple $(Q, Q^0, \mathcal{E}, \mathbb{C}, I, \rightarrow)$, where

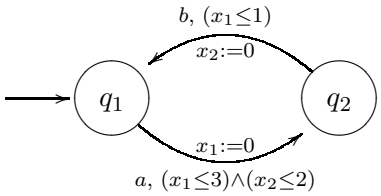
- Q: finite set of states,
- Q^0: finite set of initial states,
- $\mathcal{E}$: finite set of events,
- $\mathbb{C}$: finite set of clocks,
- $I : Q \rightarrow \Phi(\mathbb{C})$ labels each $q \in Q$ with a clock constraint,
- $\rightarrow \subseteq Q \times \mathcal{E} \times \Phi(\mathbb{C}) \times 2^{\mathbb{C}} \times Q$ is a transition relation. An element $(q, e, \varphi, \lambda, q')$ represents a transition from q to q' on event e such that the clock constraint $\varphi \in \Phi(\mathbb{C})$ and resetting the clocks $\lambda \subseteq \mathbb{C}$ to zero .

Let's consider a timed automaton example to illustrate the definition.

Example 4.9. Consider a timed automaton $A = (Q, Q^0, \mathcal{E}, \mathbb{C}, I, \rightarrow)$, where $Q = \{q_1, q_2\}$, $Q_0 = \{q_1\}$, $\mathcal{E} = \{a, b\}$, $\mathbb{C} = \{x_1, x_2\}$, $I(q_1) = I(q_2) = \mathbb{R}^2$, and the transition relation $\rightarrow$ is defined as

$$\rightarrow = \{(q_1, a, (x_1 \leq 3) \wedge (x_2 \leq 2), \{x_1\}, q_2), (q_2, b, (x_1 \leq 1), \{x_2\}, q_1)\}.$$

Graphically, the timed automaton A can be represented as



□

A timed automaton $A = (Q, Q^0, \mathcal{E}, \mathbb{C}, I, \rightarrow)$ can be viewed as a special case of a hybrid automaton $H_A = (Q, X, f, init, Inv, E, G, R)$ where

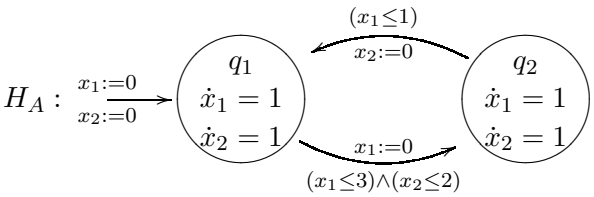
- Q is the same;
- $X = \mathbb{R}_{\geq 0}^{|\mathbb{C}|}$ defines the domain of clocks;
- $f(q, x) = \mathbf{1}$ for all $q \in Q$, where $\mathbf{1}$ stands for the vector with all elements equal one;
- $init = Q^0 \times \{\mathbf{0}\}$, where $\mathbf{0}$ stands for the null vector;
- $Inv(q) = \{x \in X \mid x \text{ satisfies the clock constraints in } I(q)\}$;
- $(q, q') \in E$ when there exist $e \in \mathcal{E}$, clock constraint $\varphi \in \Phi(\mathbb{C})$ and clocks $\lambda \subseteq \mathbb{C}$ such that $(q, e, \varphi, \lambda, q') \in \rightarrow$; and for this case, $G(q, q') = \{x \in X \mid x \text{ satisfies the clock constraints in } \varphi\}$ and $R(q, q', x_i) = 0$ if $x_i \in \lambda$ while x_i remains unchanged otherwise.

Example 4.10. Consider the timed automaton $A = (Q, Q^0, \mathcal{E}, \mathbb{C}, I, \rightarrow)$ in the previous example. It can be represented as a hybrid automaton $H_A = (Q, X, f, init, Inv, E, G, R)$ with

- $Q = \{q_1, q_2\}$;
- $x = \{x_1, x_2\}$, and $X = \mathbb{R}_{\geq 0}^2$;
- $Init = \{(q_1, \begin{bmatrix} 0 \\ 0 \end{bmatrix})\}$;
- $f(q, x) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ for all (q, x);

- $Inv(q) = \mathbb{R}^2$ for all $q \in Q$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbb{R}^2 : (x_1 \leq 3) \wedge (x_2 \leq 2)\}$, and $G(q_2, q_1) = \{x \in \mathbb{R}^2 : (x_1 \leq 1)\}$;
- $R(q_1, q_2, x) = \{0, x_2\}$, $R(q_2, q_1, x) = \{x_1, 0\}$,

Graphically,



So, a timed automaton can be seen as a special case of hybrid automata with constant one continuous flow rates, zero initial condition, rational constant bounds, and elements always being reset to zero. □

4.3.2 Timed Computation Tree Logic

In temporal logic, we can specify some properties that eventually hold true. However, we cannot explicitly specify when the property will be satisfied. Timed automata, on the other hand, provide us with the modeling power to explicitly record the time when certain states are visited and how long the system stays in that state. Hence, the temporal logic needs to be extended so to include time variables. In particular, we briefly take a look at timed computation tree logic (TCTL), which is a real-time variant of computation tree logic [Alur et al., 1990].

Definition 4.13. Let $\mathcal{P}$ be a finite set of atomic propositions, and $\mathbb{C}$ be a finite set of clock variables. Timed Computation Tree Logic (TCTL) formulas are state formulas recursively defined from predicates in $\mathcal{P}$ and atomic clock constraints of $\mathbb{C}$ according to the following rules.

1. `true`, `false`, p and g are state formulas for all predicates $p \in \mathcal{P}$ and atomic clock constraints $g \in \mathcal{ACC}(\mathbb{C})$;

2. if Φ_1 and Φ_2 are state formulas, then $\Phi_1 \wedge \Phi_2$ and $\neg\Phi_1$ are state formulas;
3. if Φ_1 and Φ_2 are state formulas, then $\Phi_1 \sqcup^J \Phi_2$ is a path formula, where $J \subseteq \mathbb{R}_{\geq 0}$ is an interval whose bounds are rational numbers;
4. if φ is a path formula, then $\exists\varphi$ and $\forall\varphi$ are state formulas;

Compared with the definition of CTL, there is no "next" operator in TCTL. This is due to the fact that we are arguing timed properties over the dense real time, so there is no meaningful next step from the current time instant. But, similar to CTL, we can use the "until" operator to define "eventually" and "always" as

- $\exists\Diamond^J\Phi = \exists\mathtt{true} \sqcup^J \Phi$,
- $\forall\Diamond^J\Phi = \forall\mathtt{true} \sqcup^J \Phi$,
- $\exists\Box^J\Phi = \neg\forall\Diamond^J\neg\Phi$,
- $\forall\Box^J\Phi = \neg\exists\Diamond^J\neg\Phi$.

In TCTL, one can explicitly specify properties hold within a time interval J. The interval J is usually written in an inequality form, e.g., $\Box^{\geq 2}$ denotes $\Box^{[2,\infty)}$ and $\sqcup^{<8}$ denotes $\sqcup^{[0,8)}$. Note that the special case $J = [0,\infty)$ can be reduced to a traditional CTL operator as the timing constraints $t \geq 0$ will trivially hold. That is $\Diamond^{[0,\infty)}\Phi = \Diamond\Phi$, $\Box^{[0,\infty)}\Phi = \Box\Phi$ and $\Phi_1 \sqcup^{[0,\infty)} \Phi_2 = \Phi_1 \sqcup \Phi_2$.

As an example, $\forall\Box(\mathtt{on} \rightarrow \forall\Diamond^{>2}\neg\mathtt{on})$ is a valid TCTL that requests the switch be always turned off within an interval of length 2, immediately after it is turned on.

The semantics of TCTL are interpreted over a timed automaton as a labeled transition system. First, some notation is helpful before translating a timed automaton into a transition system between its states. The state of a timed automaton $A = (Q, Q^0, \mathcal{E}, \mathbb{C}, I, \rightarrow)$ is a tuple (q, η), where $q \in Q$ stands for the discrete location of the state and $\eta \in \mathbb{R}^{|\mathbb{C}|}$ stands for the clock value. Since time values cannot be negative, we restrict $\eta \in \mathbb{R}^{|\mathbb{C}|}_{\geq 0}$. Here, $|\mathbb{C}|$ stands for the number of the clock variables, and the value of a particular clock variable is represented

as $\eta(x) \in \mathbb{R}_{\geq 0}$ for some $x \in \mathbb{C}$. The time elapse is captured by the notation of $\eta + d$ for nonnegative real $d \geq 0$, and $\eta + d$ is defined as new clock values with $(\eta + d)(x) = \eta(x) + d$ for all $x \in \mathbb{C}$. We can also define the reset option of a clock variable due to discrete transitions. A subset of clock variable $\mathbb{X} \subseteq \mathbb{C}$ being reset is denoted as $\mathcal{R}_{\mathbb{X}} \cdot \eta$, where

$$\mathcal{R}_{\mathbb{X}} \cdot \eta(x) = \begin{cases} \eta(x) & x \notin \mathbb{X}, \\ 0 & x \in \mathbb{X}. \end{cases}$$

The label of a timed automaton is a map from the discrete location Q to a subset of atomic propositions $\mathcal{P}$, i.e., $l : Q \to \mathcal{P}$. A labeled timed automaton is then denoted as (A, l). A labeled timed automaton (A, l) with $A = (Q, Q^0, \mathcal{E}, \mathbb{C}, I, \to)$ can be seen as a labeled transition system (T_A^t, L), where the transition system $T_A^t = (Q_A, Q_A^0, E \cup \mathbb{R}_{\geq 0}, \to_A)$ is defined by

- $Q_A = \{(q, \eta) \in Q \times \mathbb{R}^{|\mathbb{C}|} \mid \eta(x) \text{ satisfies } I(q)\}$,
- $Q_A^0 = \{(q, \eta) \in Q^0 \times \mathbb{R}^{|\mathbb{C}|} \mid \eta(x) = 0,\ \forall x \in \mathbb{C}\}$,
- $(q, \eta) \xrightarrow[A]{\delta} (q, \eta + \delta)$ for $\delta \in \mathbb{R}_{>0}$ if for all $0 < \delta' \leq \delta$, $(\eta + \delta')(x)$ satisfies $I(q)$,
- $(q, \eta) \xrightarrow[A]{e} (q', \mathcal{R}_\lambda \cdot \eta)$ if $(q, e, \varphi, \lambda, q') \in \to$ and $\eta(x)$ satisfies φ.

Note that the transition system T_A^t has infinite number of states, and two kinds of transitions, time driven transitions and discrete-event transitions. The label L for T_A^t is a map from Q_A to the subset of atomic propositions $\mathcal{P}$ or subsets of atomic clock constraints $\mathcal{ACC}(\mathbb{C})$, i.e., $L(q, \eta) \subseteq \mathcal{P} \cup \mathcal{ACC}(\mathbb{C})$. More precisely,

$$L((q, \eta)) = l(q) \cup \{g \in \mathcal{ACC}(\mathbb{C}) \mid \eta \models g\},$$

which basically labels the atomic propositions that hold for the location q and the atomic clock constraints satisfied by the clock value of the state (q, η). Here, by saying that the clock value η satisfies an atomic clock constraint $g \in \mathcal{ACC}(\mathbb{C})$, denoted as $\eta \models g$, for atomic clock constraint, we mean

- $\eta \models \texttt{true}$;
- $\eta \models x < c$ iff $\eta(x) < c$ for $x \in \mathbb{C}$;
- $\eta \models x \leq c$ iff $\eta(x) \leq c$ for $x \in \mathbb{C}$;
- $\eta \models x > c$ iff $\eta(x) > c$ for $x \in \mathbb{C}$;
- $\eta \models x \geq c$ iff $\eta(x) \geq c$ for $x \in \mathbb{C}$;

where $\eta(x) : \mathbb{C} \to \mathbb{R}_{\geq 0}$ is a clock valuation.

Then, we are ready to interpret a TCTL formula over a labeled timed automaton (A, l). TCTL formulas are state formulas arguing about a state (q, η) of (A, l), and the satisfaction of the state (q, η) for a TCTL formula Φ, denoted as $(q, \eta) \models \Phi$, is defined recursively as:

- $(q, \eta) \models \texttt{true}$
- $(q, \eta) \models p$ iff $p \in L(q, \eta)$ for $p \in \mathcal{P}$
- $(q, \eta) \models g$ iff $\eta \models g$ for $g \in \mathcal{ACC}(\mathbb{C})$
- $(q, \eta) \models \neg\Phi$ iff not $(q, \eta) \models \Phi$
- $(q, \eta) \models \Phi_1 \wedge \Phi_2$ iff $(q, \eta) \models \Phi_1$ and $(q, \eta) \models \Phi_2$
- $(q, \eta) \models \exists\varphi$ iff there exists a path from (q, η) satisfies φ
- $(q, \eta) \models \forall\varphi$ iff all the paths from (q, η) satisfy φ

A path formula φ is interpreted over a path π,

$$(q_0, \eta_0) \xrightarrow[A]{\delta_0} (q_0, \eta_0 + \delta_0) \xrightarrow[A]{e_0} (q_1, \eta_1) \xrightarrow[A]{\delta_1} (q_1, \eta_1 + \delta_1) \xrightarrow[A]{e_1} \cdots.$$

- The path π satisfies the path formula $\varphi = \Phi \sqcup^J \Psi$, denoted as $\pi \models \Phi \sqcup^J \Psi$ iff there exists i s.t. $(q_i, \eta_i + d) \models \Psi$ for some $0 \leq d \leq \delta_i$ with $\sum_{k=0}^{i-1} \delta_k + d \in J$, and $\forall j \leq i$, $(q_j, \eta_j + d') \models \Phi \vee \Psi$, for any $d' \in [0, \delta_j]$, with $\sum_{k=0}^{j-1} \delta_k + d' \leq \sum_{k=0}^{i-1} \delta_k + d$.
- The path π satisfies the path formula $\varphi = \Diamond^J \Psi$, denoted as $\pi \models \Diamond^J \Psi$ iff there exists $i \geq 0$ s.t. $(q_i, \eta_i + d) \models \Psi$ for some $0 \leq d \leq \delta_i$ with $\sum_{k=0}^{i-1} \delta_k + d \in J$.

- The path π satisfies the path formula $\varphi = \Box^J \Psi$, denoted as $\pi \models \Box^J \Psi$ iff for all $i \geq 0$ s.t. $(q_i, \eta_i + d) \models \Psi$ for any $0 \leq d \leq \delta_i$ with $\sum_{k=0}^{i-1} \delta_k + d \in J$.

Given a TCTL state formula Φ, we denote all states in (A, l) satisfying Φ as $Sat(\Phi)$ with

$$Sat(\Phi) = \{(q, \eta) \mid (q, \eta) \models \Phi\}.$$

A labeled timed automaton (A, l) satisfies a TCTL formula Φ, denoted as $(A, l) \models \Phi$, if and only all its initial states satisfy Φ, i.e., $(q_0, \mathbf{0}) \in Sat(\Phi)$ for all $q_0 \in Q_0$. From the definition, it is clear that $(A, l) \models \Phi$ if and only if $(T_A^t, L) \models \Phi$.

Note that CTL can be considered as a subclass of TCTL with all intervals J being $[0, \infty)$, for which $t \in J$ trivially holds. On the other hand, it can be shown that any TCTL Φ can be converted to an equivalent TCTL with all intervals being $[0, \infty)$, denoted as $\bar{\Phi}$, see e.g., [Baier and Katoen, 2008]. The basic idea for such a conversion is to introduce additional clock variables z in the timed automaton to track how long a certain sub-formula of Φ remains true. More precisely, given a labeled timed automaton (A, l), with $A = (Q, Q^0, \mathcal{E}, \mathbb{C}, I, \rightarrow)$ and $l : Q \rightarrow 2^{\mathcal{P}}$ and a TCTL formula over $\mathbb{C}$ and $\mathcal{P}$. We introduce a new clock variable z into $\mathbb{C}$, and re-define $A \oplus z = (Q, Q^0, \mathcal{E}, \mathbb{C} \cup \{z\}, I, \rightarrow)$. The state in $A \oplus z$ will be $(q, \eta \oplus z)$, and it holds that

- $(q, \eta) \models \exists(\Phi \sqcup^J \Psi)$ iff $(q, \eta \oplus z) \models \exists((\Phi \vee \Psi) \sqcup ((z \in J) \wedge \Psi))$.
- $(q, \eta) \models \forall(\Phi \sqcup^J \Psi)$ iff $(q, \eta \oplus z) \models \forall((\Phi \vee \Psi) \sqcup ((z \in J) \wedge \Psi))$.

For example, the TCTL formula $\exists\Diamond^{\leq 2}\Phi$ can be converted to a CTL formula $\exists\Diamond((z \leq 2) \wedge \Phi)$. They are equivalent in the sense that a labeled timed automaton $(A, l) \models \exists\Diamond^{\leq 2}\Phi$ if and only if $(A \oplus z, l) \models \exists\Diamond((z \leq 2) \wedge \Phi)$. Note that the TCTL model checking problem has been converted to a CTL model checking problem. Hence, without loss of generality, we will focus on CTL model checking for timed automata.

4.3.3 Timed Automata Model checking

The CTL model checking problem for a labeled timed automata is still difficult since there are infinite number of states and transitions

in timed automata. To overcome this difficulty, we first abstract all transitions due to time elapse into one transition, which is called the *time-abstract transition system* of the timed automaton, and then derive a finite quotient transition system, called a *region transition system* of the timed automaton, which is bisimilar to the time-abstract transition system.

First, we introduce the *time-abstract transition system* of the corresponding timed automata. Given a labeled timed automaton (A, l), with $A = (Q, Q^0, \mathcal{E}, \mathbb{C}, I, \rightarrow)$ and $l : Q \rightarrow 2^{\mathcal{P}}$, its corresponding labeled time-abstract transition system is (T_A, L) with $T_A = (Q_A, Q_A^0, \mathcal{E} \cup \{\tau\}, \rightharpoonup)$ and $L : Q_A \rightarrow 2^{\mathcal{P}} \cup \mathcal{ACC}(\mathbb{C})$. The state sets Q_A, Q_A^0 are the same as defined in T_A^t, i.e., $Q_A = \{(q, \eta) \in Q \times \mathbb{R}^{|\mathbb{C}|} \mid \eta(x) \text{ satisfies } I(q)\}$ and $Q_A^0 = \{(q, \eta) \in Q^0 \times \mathbb{R}^{|\mathbb{C}|} \mid \eta(x) = 0,\ \forall x \in \mathbb{C}\}$. The transition relation $\rightharpoonup \subseteq Q_A \times (\mathcal{E} \cup \{\tau\}) \times Q_A$ is defined as:

- For $q \in Q$, $((q, \eta), \tau, (q, \eta')) \in \rightharpoonup$ if there exists $d \geq 0$ such that $\eta' = \eta + d$ and for all $0 < \delta \leq d$, $(\eta + \delta)(x)$ satisfies $I(q)$;
- For an $e \in \mathcal{E}$, $((q, \eta), e, (q', \eta')) \in \rightharpoonup$ if $(q, e, \varphi, \lambda, q') \in \rightarrow$ and $\eta(x)$ satisfies φ and $\eta' = \mathcal{R}_\lambda \cdot \eta$.

The rationale for introducing the labeled time-abstract transition system for a labeled timed automaton is due to the fact that a CTL formula is satisfied by (A, l) if and only if it is satisfied by (T_A, L). However, the direct application of CTL model checking techniques to (T_A, L) is infeasible since the state space of T_A is infinite, i.e., T_A is an infinite transition system. Next, we are going to show that there exists a finite quotient transition system of (T_A, L), called region transition system, that is bisimilar to (T_A, L).

It can be shown that all constants can be assumed to be integers without loss of generality. For a timed automaton $A = (Q, Q^0, \mathcal{E}, \mathbb{C}, I, \rightarrow)$, $\alpha A = (Q, Q^0, \mathcal{E}, \mathbb{C}, I_\alpha, \rightarrow_\alpha)$ is another timed automaton obtained simply by replacing all constant c in A by αc, where α is an arbitrary positive rational number. For a clock valuation $\eta : \mathbb{C} \rightarrow \mathbb{R}_{\geq 0}^{|\mathbb{C}|}$, $\alpha\eta : \mathbb{C} \rightarrow \mathbb{R}_{\geq 0}^{|\mathbb{C}|}$ is defined as $\alpha\eta(x) = \alpha(\eta(x))$ for all $x \in \mathbb{C}$. Let T_A and $T_{\alpha A}$ be the time-abstract transition systems correspond to timed automata A and αA respectively. The labels for $T_{\alpha A}$ is

defined as $L_\alpha(q, \eta) = L(q, \frac{1}{\alpha}\eta)$.

Lemma 4.1. [Alur and Dill, 1994] (T_A, L) is bisimilar to $(T_{\alpha A}, L_\alpha)$.

Since all constants c are assumed to be rational, there exists an integer α, say the common multiple of their denominators, such that all constants in αA are integers. Therefore, we may assume without loss of generality that all constants are integers. Let c_i denote the largest integer constant with which x_i is compared.

In the following, we are going to show that there exists an auto-bisimulation relation R for (T_A, L) such that its quotient system has finite states. Let's define the following equivalent relationship and prove that it is an auto-bisimulation relationship.

Consider the relationship R, we say $((q, \eta), (q', \eta')) \in R$ if the following hold:

1. $q = q'$;
2. for any $x_i \in \mathbb{C}$, it holds that $\eta(x_i) > c_i$ and $\eta'(x_i) > c_i$, or
3. for any x_i, $x_j \in \mathbb{C}$ with $\eta(x_i), \eta'(x_i) \leq c_i$ and $\eta(x_j), \eta'(x_j) \leq c_j$, all the following conditions hold
   (a) $(\langle\eta(x_i)\rangle \leq \langle\eta(x_j)\rangle) \Leftrightarrow (\langle\eta'(x_i)\rangle \leq \langle\eta'(x_j)\rangle)$;
   (b) $\lfloor\eta(x_i)\rfloor = \lfloor\eta'(x_i)\rfloor$ and $(\langle\eta(x_i)\rangle = 0) \Leftrightarrow (\langle\eta'(x_i)\rangle = 0)$.

Here we use the following notation. For any $\delta \geq 0$, $\langle\delta\rangle$ denotes the fraction part of δ, while $\lfloor\delta\rfloor$ the integer part, and $\delta = \lfloor\delta\rfloor + \langle\delta\rangle$.

Two clock values η and η' are called equivalent, denoted as $\eta \approx \eta'$, if there exists $q \in Q$ such that $((q, \eta), (q, \eta')) \in R$. The collection of all equivalent clock values is called a clock region, denoted by $[\eta] = \{\eta' \mid \eta \approx \eta'\}$. The clock regions are either open triangles, open line segments, open parallelograms or points as shown in the following example.

Example 4.11. Consider the clock set $\mathbb{C} = \{x, y\}, C_x = 2, C_y = 1$, the clock region is depicted in Figure 4.1. For example, the collection of all equivalent clock values for $x = 3$, $y = 2$ is $\{x > 2, y > 1\}$. The

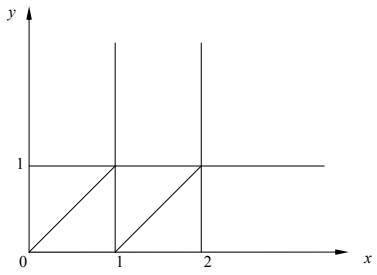


Figure 4.1: Example of clock equivalence.

clock region contains $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ is the origin itself. A clock region could be an open triangle, for example $\{0 < x < 1, 0 < y < 1, x > y\}$. Line segments, e.g., $\{x > 2, y = 1\}$, $\{x = 1, 0 < y < 1\}$, $\{1 < x < 2, x = y\}$, are also clock regions. □

Also, clock values η and η' from the same clock region remain equivalent after resetting. That is if $\eta \approx \eta'$ then we have $\mathcal{R}_\lambda \cdot \eta \approx \mathcal{R}_\lambda \cdot \eta'$ for any subset of clocks $\lambda \subseteq \mathbb{C}$.

Since the clock constants are all assumed to be integers, the clock valuations η and η' will satisfy the same clock constraints if they belong to the same clock region. Hence, for any $((q, \eta), (q', \eta')) \in R$ we have $L(q, \eta) = L(q', \eta')$. It can be easily seen that R defined above is an equivalence relation.

Theorem 4.1. [Alur and Dill, 1994] R is a bisimulation relation.

Since R is a bisimulation relation for (T_A, L), then we can derive a quotient transition system using R that is bisimilar to (T_A, L). In particular, the quotient transition system can be obtained as

$$T_A/_R = (Q_A/_R, Q_A^0/_R, \mathcal{E} \cup \{\tau\}, \rightharpoonup_R),$$

where

- $Q_A/_R = \{(q, [\eta]) \mid (q, \eta) \in Q_A\}$;
- $Q_A^0/_R = \{(q_0, [\eta]) \mid (q_0, \eta) \in Q_A^0\}$;

- The transition relation $\rightharpoonup_R$ is defined as
  - τ transition: For $q \in Q$, $((q,[\eta]),\tau,(q,[\eta'])) \in \rightharpoonup_R$ if for all $\eta \in [\eta]$ there exists $d > 0$ such that $\eta' = \eta + d$ and for all $0 \leq \delta \leq d$, $(\eta + \delta)(x) \in [\eta] \cup [\eta'] \subseteq I(q)$;
  - Discrete-transitions: For an $e \in \mathcal{E}$, $((q,[\eta]),e,(q',[\eta'])) \in \rightharpoonup_R$ if for any $\eta \in [\eta]$ there exists $\eta' \in [\eta']$ such that $((q,\eta),e,(q',\eta')) \in \rightharpoonup$ in T_A.

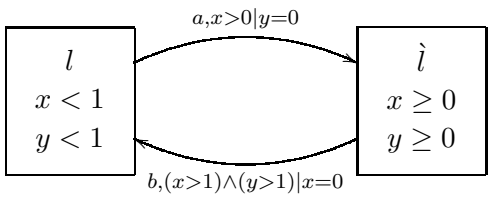
The label for the quotient transition system $T_A/_R$ is a map $L_R : Q_A/_R \to 2^{\mathcal{P}} \cup 2^{\mathcal{ACC}}(\mathbb{C})$ an is defined as

$$L_R(q,[\eta]) = L(q,\eta).$$

Note that for any η_1, $\eta_2 \in [\eta]$, $L(q,\eta_1) = L(q,\eta_2)$. So, the above definition is consistent.

The quotient transition system $T_A/_R$ is using the equivalent clock regions as its states, so it is called a *region transition system* of the timed automaton. Since the clock region is finite, $T_A/_R$ is a finite transition system. This is significant since we know how to do model checking on finite state machines and algorithms always terminate. Combining with our knowledge that CTL* preserves on bisimilar transition systems, so we can do CTL* or CTL model checking for a timed automaton on its corresponding region transition system, as long as the CTL* or CTL formula is arguing over atomic propositions $\mathcal{P}$ defined for discrete locations.

Example 4.12. [Baier and Katoen, 2008] To illustrate the region transition system, consider a timed automaton A with two locations:



In this example, we do not label discrete locations, and we are just interested in the property whether it is possible to keep the clock y

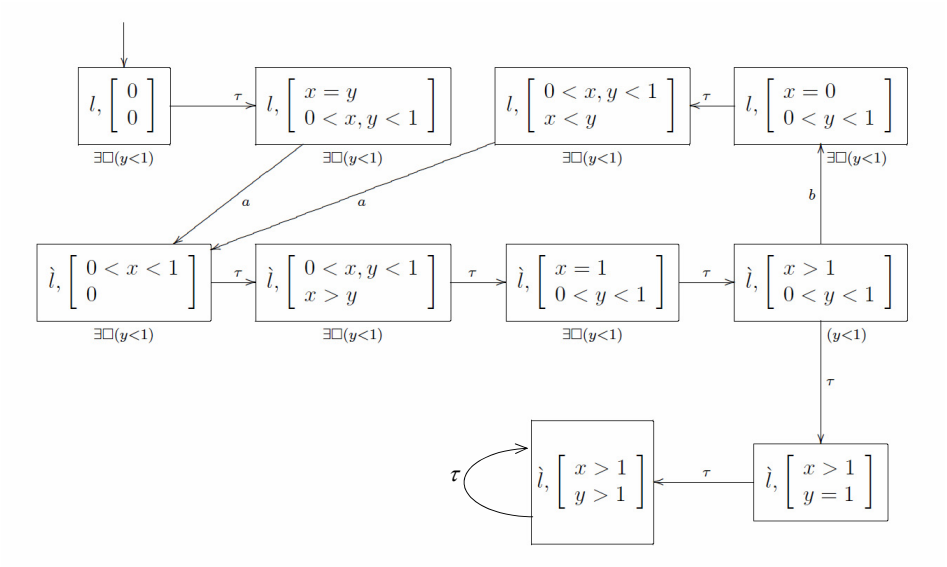


Figure 4.2: The region transition system for the timed automata A in Example 4.12 in [Baier and Katoen, 2008].

less than one, i.e., the CTL formula $\exists\Box(y < 1)$. Using the definition of clock region and region transition system above, we obtain the region transition system of A, denoted as $T_A/_R$, in Figure 4.2. Clearly, $T_A/_R$ has finite states and the CTL model checking problem for A can be carried on $T_A/_R$, which is a traditional CTL model checking problem as we solved in the previous section. It is easy to show that $T_A/_R \models \exists\Box(y < 1)$. So, $A \models \exists\Box(y < 1)$ as $A \cong T_A/_R$. □

4.4 Hybrid Automata

In timed automata, the continuous flow rates are constants equal to one for all continuous variables (clocks). This maybe restrictive in real applications, and motivates researchers to consider more general continuous dynamics in each mode. In particular, we consider multirate automata and rectangular automata, which are extensions of timed automata and are also special cases of hybrid automata [Henzinger,

1995, Alur et al., 2000a].

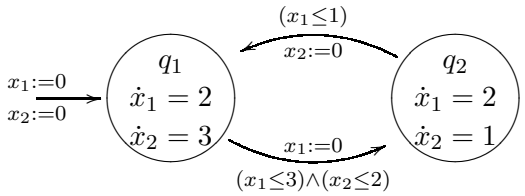
4.4.1 Multirate Automata

Multirate automata [Alur et al., 1995] are generalizations of timed automata, where each variable follows constant, rational slopes, which may be different in different locations.

Definition 4.14. A *multirate automaton* is a hybrid automaton $H = (Q, X, Init, f, Inv, E, G, R)$, where

- Q is a set of discrete variables, $Q = \{q_1, \cdots q_m\}$;
- $X = \{x_1 \cdots, x_n\}$, $\mathbf{X} = \mathbb{R}^n$;
- For each location $q \in Q$, the set $Init(q)$ is either empty or a singleton set;
- $f(q, x) = b_q$, where b_q is a constant vector with all rational components;
- $Init(q)$ is a rectangle for all q;
- $E \subseteq Q \times Q$;
- The guard set $G(e)$ is a rectangle[2] for all $e = (q, q') \in E$; and
- For all $e \in E$, the reset map $R(e, x) = R_1(e, x) \times \cdots \times R_n(e, x)$, where $R_i(e, x)$ either equals x_i (unchanged) or a constant (the i-th component of x is reset to be a constant).

As an example, a multi-rate automaton is plotted below



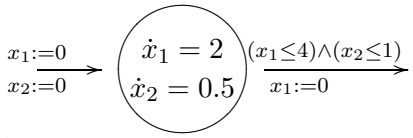
A multirate automaton is called *initialized* if after a discrete transition the bounds on the derivative of a variable change, then its

[2] A set $R \subset \mathbb{R}^n$ is called a rectangle if $R = \prod_{i=1}^{n} R_i$ where R_i are bounded or unbounded intervals whose finite end points are rational. For example, the set $R = \prod_{i=1}^{3} R_i$ with $R_1 = (1, \infty)$, $R_2 = \{3\}$, $R_3 = (-2, \frac{3}{4})$ is a rectangle.

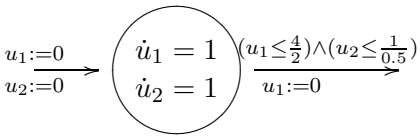
value must be nondeterministically reset ("re-initialized") within a fixed interval. Mathematically, an initialized multirate automaton requests that if for all transitions $e = (q, q') \in E$ and all $1 \leq i \leq n$, $R_i(e, x) = \{x_i\}$ then $f_i(q, \cdot) = f_i(q', \cdot)$.

It is known that initialized multirate automata have finite bisimulation quotient systems [Alur et al., 1995]. The main idea behind this is to re-scale the slope of each variable to 1 so to convert the multirate automaton into a timed automaton. In addition, it is necessary to appropriately adjust all initial, invariant and guard sets, as well as reset maps. From the region equivalence of the resulting timed automaton, a bisimulation of the multirate automaton can be obtained.

To illustrate the idea, we consider a part of a multirate automaton as below



It can be converted into a timed automaton



Hence, the LTL and CTL model checking problem for an initialized multirate automaton can be carried on the corresponding region transition system that is of a finite number of states, provided every position occurring in temporal formulas is either an automaton location or a rectangular set. Hence, the LTL and CTL model checking problem for an initialized multirate automaton can terminate in finite steps (called decidable) [Alur et al., 1995].

4.4.2 Rectangular Automata

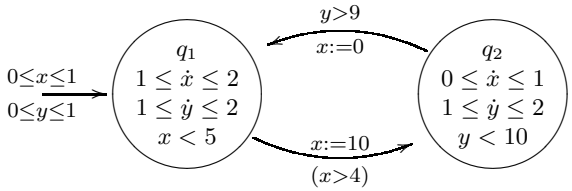
A rectangular automaton is a special case of a hybrid automaton where all initial regions, invariant sets, vector fields, guard sets and reset maps are rectangular. Formally,

Definition 4.15. [Henzinger et al., 1995] A *rectangular automaton* is a hybrid automaton $H = (Q, X, Init, f, Inv, E, G, R)$, where

- Q is a set of discrete variables, $Q = \{q_1, \cdots q_m\}$;
- $X = \{x_1 \cdots, x_n\}$, $\mathbf{X} = \mathbb{R}^n$;
- $f(q, x) = F(q)$ for all (q, x), where $F(q) = F_1(q) \times \cdots \times F_n(q)$ is a rectangle;
- $Init(q)$ and $Inv(q)$ are all rectangles for all q;
- $E \subseteq Q \times Q$;
- $G(e) = G_1(e) \times \cdots \times G_n(e)$ is a rectangle for all $e = (q, q') \in E$;
- For all $e \in E$, the reset map $R(e, x) = R_1(e, x) \times \cdots \times R_n(e, x)$, where $R_i(e, x)$ either equals x_i (unchanged) or a fixed interval (the i-th component of x is reset to any value within this interval).

In other words, the derivative of each variable stays between two fixed bounds, which may be different in different discrete modes, i.e., $l_q(i) \leq \dot{x}_i \leq u_q(i)$, for all $1 \leq i \leq n$ and $q \in Q$. With each discrete transition between two modes $(q, q') \in E$, the value of variable x_i is either left unchanged $R_i(e, x) = \{x_i\}$ or reset nondeterministically to a new value within some fixed, constant interval when $R_i(e, x)$ is a fixed interval.

As an example, a rectangular automaton is plotted below.

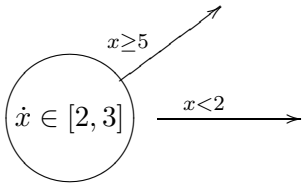


A rectangular automaton is called initialized if after a discrete transition the bounds on the derivative of a variable change, then its value must be nondeterministically reset ("re-initialized") within a fixed interval. Mathematically, an initialized rectangular automaton requests that if for all transitions $e = (q, q') \in E$ and all $1 \leq i \leq n$, $R_i(e, x) = \{x_i\}$ then $F_i(q) = F_i(q')$.

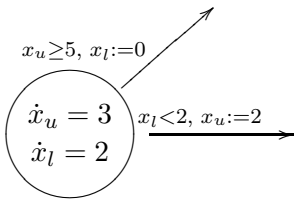
An initialized rectangular automaton can be converted into a language equivalent initialized multirate automaton [Henzinger et al., 1995]. The main idea is to replace each variable x_i, which satisfies a

differential inclusion of the form $\dot{x}_i \in [a_i, b_i]$ by two variables named x_i^l and x_i^u, which satisfy $\dot{x}_i^l = a_i$ and $\dot{x}_i^l = b_i$, respectively. The variables x_i^l and x_i^u keep track of the lower and upper bounds of x_i. The initial, invariant, and guard sets, as well as the reset maps must be adjusted accordingly. This conversion from rectangular to a multirate automaton is language preserving. Hence, from the existence of finite bisimulation for an initialized multirate automaton, we can construct a finite language equivalence of the original initialized rectangular automaton.

To illustrate the idea, we consider a part of a rectangular automaton as below.

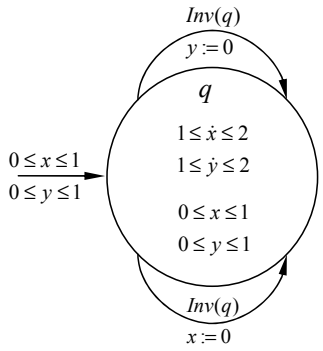


It can be converted into a multirate automaton by introducing two variables x_u and x_l, representing the upper and lower bounds of x respectively.



It then can be converted into a timed automaton. Since we can obtain a language equivalent finite transition system for an initialized rectangular automaton, its LTL model checking problem can terminate in finite steps (called decidable). However, the conversion from rectangular to a multirate automaton may not preserve branching properties, and in general, initialized rectangular automata do not admit finite bisimulation quotients.

Consider a rectangular automaton from [Alur et al., 2000a]:



It only contains one location $Q = \{q\}$, and all regions are given as rectangles. Since it only has one location, it is trivially initialized. However, it does not have a finite bisimulation quotient.

4.5 Notes and Further Reading

Temporal logic, transition systems and model checking have been successfully used in applications such as the verification of software and communication protocols [Clarke et al., 1999, Bérard et al., 2010]. Software model checkers have been developed, such as SPIN [Holzmann, 1997] for LTL model checking, symbolic model checker NUSMV [Cimatti et al., 2002] and PRISM [Kwiatkowska et al., 2002] for probabilistic model checking. Due to space limitation, our treatment of these topics are unavoidably very brief and even superficial in some cases. Interested readers may find a more detailed and comprehensive treatment of these topics in books, e.g., [Clarke et al., 1999, Baier and Katoen, 2008].

Timed automata were proposed in the early 1990s [Alur and Dill, 1994], where a formal timed language theory on timed automata was developed. Tools such as KRONOS [Yovine, 1997] and UPPAAL [Behrmann et al., 2004] have been developed for the verification of real time systems modeled as timed automata. Our discussions on multirate automata and rectangular automata are mainly based on re-

sults from [Henzinger, 1995, Alur et al., 2000a]. Generally speaking, the reachability (hence verification) problem is undecidable for hybrid systems. Only a small portion of hybrid systems with simple or specific continuous variable dynamics, such as timed automata and initialized rectangular automata, have finite quotient transition systems, and thus the reachability problem for such specific hybrid systems is known to be decidable [Henzinger et al., 1998]. It is known that even a slight generalization of the multirate automata or rectangular automata could make the reachability problem undecidable [Henzinger et al., 1995, 1998]. Another class of hybrid automata with finite bisimulation, called o-minimal hybrid automata, was introduced in [Lafferriere et al., 2000]. In [Alur et al., 2000a], an excellent survey on abstractions of hybrid automata was given.

Motivated by the undecidability of general hybrid systems, many efforts have been devoted to the computation/approximation of reachable sets for a given hybrid system. A good deal of research effort has focused on developing sophisticated techniques drawn from optimal control, game theory, and computational geometry to calculate or approximate the reachable sets for various classes of hybrid systems, and several software tools have been developed for such a purpose, see e.g., [Chutinan and Krogh, 2003, Tomlin et al., 2003]. The tool HYTECH [Henzinger et al., 1997] uses symbolic reachability analysis for linear hybrid automata and represents the reachable sets as polyhedrons. Linear hybrid automata are a special case of hybrid automata, where guards, reset and invariants only involve linear expressions, and the continuous dynamics are restricted to linear constraints such as $\dot{x} = \dot{y} \wedge 1 \leq \dot{x} \leq 2$. More general continuous dynamics are handled in tools like CHECKMATE [Silva et al., 2000] and later refined by the tool `d/dt` [Asarin et al., 2002], which compute over approximations of reachable sets using polyhedral-based representations, and the resulting approximation of the reachable set is called the flowpipe approximation.

Polyhedra-based representations are appealing as they are widely used and there are open source libraries available for manipulating them. However, the complexity of operations on polyhedrons is ex-

ponential with respect to the dimensions of the continuous variables, so the size of systems that could be handled was modest. To reduce computational complexity, the tool SPACEEX [Frehse et al., 2011] represents continuous sets using Zonotopes and support functions to compute an over-approximation of the reachable states.

Deductive verification methods for hybrid system verification are also being developed in the literature. A good example is the use of barrier certificate for safety analysis of continuous, stochastic and hybrid systems [Prajna and Jadbabaie, 2004]. The basic idea is to find a barrier function that is decreasing along system trajectories and have zero level set (barrier) that no solution trajectory crosses. Should the barrier separate the initial states from unsafe regions, the safety of the dynamical system can be deduced. The barrier certificate is conceptually intuitive, but the finding of a barrier certificate is not easy and is similar to the generation of a Lyapunov function. Computational methods based on, e.g., sum of squares (SOS) [Papachristodoulou and Prajna, 2005], have been developed to generate the barrier certificate, but the dimension of the systems that can be handled by the computational tool is limited.

To handle higher dimensional dynamical systems and/or systems without precise mathematical models, which is usually the case in practical applications, simulation based [Nghiem et al., 2010, Dang et al., 2008] and sampling-based approaches [Branicky et al., 2006, Plaku et al., 2013] have been developed recently in the literature. These numerical based approaches aim to find wittiness traces of the model to be checked that falsify certain safety properties in concern. It is more like a testing approach as if the methods cannot find such a wittiness trace after a long period of testing, one cannot conclude that the safety property holds for sure. Another trend in the software verification community is to automatically synthesize codes from required specifications [Pnueli and Rosner, 1989]. This has a close relationship with the supervisory control problems that will be discussed in the next chapter.

5

Hybrid Supervisory Control

So far, we have briefly reviewed some basic results for the analysis and synthesis of hybrid systems. These results have been introduced by both the computer science and control engineering communities. The computer science community has been primarily interested in verifying the correctness of embedded codes in the face of real time constraints and interactions with the physical world; while the control engineering community has been mainly focusing on the effects of discontinuousness (such as switching of dynamics and jumping of states) caused by the use of digital processors and communication networks that are becoming more and more popular nowadays.

The past decade has seen efforts to merge these two schools of thought, and a noticeable trend in the recent hybrid system literature is the emphasis on the synthesis of hybrid controllers for continuous or hybrid dynamical systems to satisfy complicated temporal logic specifications. This approach is known as symbolic control or hybrid supervisory control. The basic idea is first to obtain equivalent or approximating finite abstractions of the hybrid or continuous systems to be controlled. Then the design is carried out in the discrete domain using model checking [Clarke et al., 1999], game theoretic ap-

proaches [Pnueli and Rosner, 1989] or discrete event supervisory control theory [Ramadge and Wonham, 1989]. The final step is to convert the designed discrete supervisor back to a hybrid controller (since the controller usually contains both discrete transitions and continuous flows) so that it can be used to control the original hybrid or continuous plants. The effectiveness of the abstraction-based method depends on whether or not there exists a finite state discrete abstracted model for the original hybrid or continuous system. Significant research efforts have been devoted to developing abstraction methods based on reachability analysis, see e.g., [Alur et al., 2000a, Tabuada, 2009].

This chapter introduces the work on hybrid supervisory control. It is organized as follows. First, the basic theory for supervisory control of discrete event systems is briefly introduced in Section 5.1. Then, the supervisory control for timed automata is introduced in Section 5.2 as an extension of supervisory control theory to timed languages. The supervisory control of discrete event systems is directly applicable to hybrid systems provided that we could obtain a finite abstraction of the hybrid system, e.g., a finite bisimilar quotient transition system. Then Section 5.3 is mainly on illustrating the abstraction-based supervisory control design process for linear and nonlinear continuous systems, and hybrid systems with multi-affine dynamics.

5.1 Discrete Event Supervisory Control

Discrete event systems (DESs) refer to a class of dynamical systems with discrete states, where the evolution of discrete states is driven by the occurrence of discrete events [Ramadge and Wonham, 1989, Cassandras and Lafortune, 2008]. It is assumed that these events happen instantaneously, and the concern is mainly on the ordering of occurrence of events and the logic behind it. We usually use symbols, e.g., $\{a, b, c\}$, to represent these events, and the collection of all possible events of the DES under study is called the *event set*, denoted as Σ. The dynamical behavior exhibited by the DES is then captured by a collection of traces with symbols from Σ, which are called *languages* generated from the DES. Formally, let Σ^* denote the set of all finite

strings over Σ, including the empty string ϵ. Any subset of Σ^* is a *language*. A language $K \subseteq \Sigma^*$ is said to be *prefix-closed* if $K = pr(K)$, where $pr(K) = \{s \in \Sigma^* : (\exists t \in \Sigma^*)\ s \cdot t \in K\}$ is the prefix closure of the language K. Here $s \cdot t$ stands for the concatenation of two strings s and t.

This section aims to briefly introduce the discrete event system supervisory control theory, and our treatment here mainly follows [Ramadge and Wonham, 1989]. The basic supervisory control problem for discrete event systems (DES) can be stated as follows: Given a plant modeled as a finite automaton, $G = (X, \Sigma, \alpha, x^0, X^M)$, and a specification $K \subseteq \Sigma^*$, the control objective is to design a supervisor/controller f such that the closed-loop system, denoted as G^f, behaves in a desirable way, in the sense that the generated language of G^f, denoted as $\mathcal{L}(G^f)$, equals K. Besides all possible strings, we also interested in some subsets (sublanguage) of $\mathcal{L}(G^f)$, called *marked language* of G^f and denoted as $\mathcal{L}_M(G^f)$, which consist of the traces in the generated language $\mathcal{L}(G^f)$ whose executions imply the completion of a certain task.

To achieve such a desired behavior, the supervisor restricts the events happening in the plant G. Some events may not be disabled, for example the failure of a machine. Hence, the event set Σ is partitioned into two disjoint subsets: $\Sigma = \Sigma_c \cup \Sigma_u$, in which Σ_c and Σ_u are the sets of controllable and uncontrollable events, respectively. The supervisor restricts the behavior of the plant by dynamically disabling some of the controllable events. Mathematically, a supervisor can be represented as a map, $f : \mathcal{L}(G) \to 2^{\Sigma_c}$, which maps a string of the plant $s \in \mathcal{L}(G)$ to a set of controllable events. The set $f(s) \subseteq \Sigma_c$ stands for the set of events that are disabled by the supervisor after the execution of s.

Definition 5.1. The generated language, denoted by $\mathcal{L}(G^f)$, and the marked language $\mathcal{L}_M(G^f)$, of the controlled system under supervision are defined by:

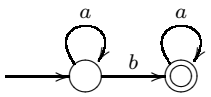
1. $\epsilon \in \mathcal{L}(G^f)$;
2. $\forall s \in \Sigma^*, \sigma \in \Sigma$, if $s \in \mathcal{L}(G^f)$, $s\sigma \in \mathcal{L}(G)$ and $\sigma \notin f(s)$ then $s\sigma \in \mathcal{L}(G^f)$;

3. $\mathcal{L}_M(G^f) = \mathcal{L}(G^f) \cap \mathcal{L}_M(G)$.

In other words, if a string s is in the controlled generated behavior, and an event σ, which is not disabled or uncontrollable, i.e., $\sigma \notin f(s)$, can occur in the plant after s, namely $s\sigma \in \mathcal{L}(G)$, then $s\sigma$ should be in the controlled generated behavior as well. The marked language $\mathcal{L}_M(G^f) = \mathcal{L}(G^f) \cap \mathcal{L}_M(G)$ means that the controlled marked language equals the set of marked strings of the plant that survive under control. It is clear that $\mathcal{L}(G^f) \subseteq \mathcal{L}(G)$, $\mathcal{L}_M(G^f) \subseteq \mathcal{L}_M(G)$, $pr(\mathcal{L}_M(G^f)) \subseteq \mathcal{L}(G^f)$ and $\mathcal{L}(G^f)$ is prefix closed. A supervisor f is said to be *non-blocking* if $pr(\mathcal{L}_M(G^f)) = \mathcal{L}(G^f)$. For illustration, consider the following example from the DES literature.

Example 5.1. Consider an automaton

G :



$\Sigma = \{a, b\}$ and assume $\Sigma_c = \{a\}$.

$$\mathcal{L}(G) = pr(a^*ba^*) = a^* + a^*ba^*$$

and

$$\mathcal{L}_M(G) = a^*ba^*.$$

Consider the desired generated behavior to be:

$$K = pr(\{a^kba^k, k \leq m\}),$$

which basically requests that b needs to happen exactly once and a occurs the same number of times before and after b.

When $m = 1$, $K = \{\epsilon, b, a, ab, aba\}$, it is easy to check that

$$\emptyset \neq K = pr(K) \subseteq \mathcal{L}(G),$$

and we can specify the supervisor as follows.

$$f(s) = \begin{cases} \{a\} & s = a, b, aba \\ \emptyset & otherwise \end{cases}$$

It is not difficult to check that the closed-loop system behavior satisfies that $\mathcal{L}(G^f) = K$ for $m = 1$. □

5.1.1 Existence of Supervisors

A basic question is when the supervisory control problem has a feasible solution, i.e., the existence of a supervisor f for the plant G with respect to a given specification K. Recall that the basic idea in the supervisory control is to disable certain events and their associated transitions of the plant G so that its behavior lies within some prescribed range. However, the transitions triggered by uncontrollable events cannot be disabled. A supervisor should never try to disable an uncontrollable event. Instead, it needs to disable upstream controllable events. This requirement on f is called Σ_u-enabling.

Definition 5.2. A supervisor $f : \Sigma^* \to 2^\Sigma$ is called Σ_u-*enabling* if $\forall s \in \Sigma^*,\ \sigma \in \Sigma_u$, we have $(s \in \mathcal{L}(G^f)) \wedge (s\sigma \in \mathcal{L}(G)) \Rightarrow s\sigma \in \mathcal{L}(G^f)$.

Intuitively, it means that if a string s is in the controlled generated behavior (i.e., $s \in \mathcal{L}(G^f)$), and an uncontrollable event σ can occur in the plant after s, namely $s\sigma \in \mathcal{L}(G)$, then $s\sigma$ should be in the controlled generated behavior as well (i.e., $s\sigma \in \mathcal{L}(G^f)$).

Due to the requirement of Σ_u-enabling, a specification K should not request a supervisor to disable uncontrollable events. This can be captured by the definition of controllability.

Definition 5.3. Given a plant G, a language $K \subseteq \Sigma^*$ is said to be *controllable* if $pr(K)\Sigma_u \cap \mathcal{L}(G) \subseteq pr(K)$.

This means that $\forall s \in \Sigma^*, \sigma \in \Sigma_u$, if $s \in pr(K)$ and $s\sigma \in \mathcal{L}(G)$ then $s\sigma \in pr(K)$. It is easy to see from the definition that a Language K is controllable if and only if $pr(K)$ is controllable. Also, a supervisor f is Σ_u enabling if and only if $\mathcal{L}(G^f)$ is controllable.

Example 5.2. Continuing the previous example, consider the language $K_1 = \{a, b, ab\}$ and $K_2 = \{a, ba, ab\}$. Both K_1 and K_2 are controllable, so is their union. In general, the union of controllable languages is still controllable. However, controllability does not hold for intersections. For example, $K_1 \cap K_2 = \{a, ab\}$ is not controllable (since $\epsilon \in pr(K_1 \cap K_2)$, $b \in L(G)$ but $b \notin pr(K_1 \cap K_2)$). □

The following theorem states necessary and sufficient conditions for the existence of a Σ_u-enabling and non-blocking supervisor f for a given specification language K.

Theorem 5.1. [Ramadge and Wonham, 1989] Let G be a plant, and a prefix closed nonempty language $\emptyset \neq K = pr(K) \subseteq \mathcal{L}(G)$ be a given specification. There exists a Σ_u-enabling supervisor f such that $\mathcal{L}(G^f) = K$ if and only if K is controllable.

To illustrate the result, let's revisit the previous example.

Example 5.3. Consider the previous example. The desired generated behavior $K = pr(\{a^k b a^k, k \leq 1\})$. Then, $K = \{\epsilon, b, a, ab, aba\}$, so $\emptyset \neq K = pr(K) \subseteq \mathcal{L}(G)$. Further, to check the controllability of K, it is easy to see that $\forall s \in \Sigma^*$, if $s \in pr(K)$, for $b \in \Sigma_u$, $sb \in \mathcal{L}(G)$ then $sb \in pr(K)$. So, K is controllable. Hence, there exists a Σ_u-enabling supervisor f such that $\mathcal{L}(G^f) = K$ as given in Example 5.1. □

Example 5.4. Consider another language $K_a = \{a^*\}$. $K_a \subseteq \mathcal{L}(G)$ is prefix closed but not controllable since for example $s = a^n \in K_a$ for any integer n, $sb \in \mathcal{L}(G)$ but $sb \notin K_a$. Hence, by the above theorem, we can say that there does not exist a controller to enforce the generated behavior K_a. □

The following theorem states necessary and sufficient conditions for the existence of a Σ_u-enabling and non-blocking supervisor f for a given specification language K.

Theorem 5.2. [Ramadge and Wonham, 1989] Let G be a plant, and $\emptyset \neq K \subseteq \mathcal{L}_M(G)$ be a given specification. There exists a Σ_u-enabling and non-blocking supervisor f such that $\mathcal{L}_M(G^f) = K$ if and only if K is controllable and relative closed, i.e., $K = pr(K) \cap \mathcal{L}_M(G)$.

Example 5.5. Consider the previous example. A language $K = \{a^k b a^k,\ k \geq 0\}$, K is controllable, but not relative closed (for example, $ab \in pr(K) \cap \mathcal{L}_M(G)$ but $ab \notin K$ since any string in K needs to have the same number of a before and after b). Hence, there does not exist any Σ_u-enabling and non-blocking, supervisor f to enforce the marked behavior to be K exactly. □

5.1.2 Realization of Supervisors

In this subsection, we show that the supervisor defined as a map f in the previous section can equivalently be realized as a finite automaton operating in synchronous composition with the plant G under certain conditions. Formally, the synchronous composition between two automata can be defined as follows.

Definition 5.4. The *synchronous composition* of two automata $G_i = (X_i, \Sigma_i, \alpha_i, x_i^0, X_i^M)$, for $i = 1, 2$, denoted as $G_1 \| G_2 = (X, \Sigma, \alpha, x^0, X^M)$, is defined by

1. $X = X_1 \times X_2$;
2. $\Sigma = \Sigma_1 \cup \Sigma_2$;
3. $x^0 = (x_1^0, x_2^0)$;
4. $X^M = X_1^M \times X_2^M$;
5. For each $x \in (x_1, x_2)$, $\sigma \in \Sigma$,

$$\alpha(x, \sigma) = \begin{cases} (\alpha_1(x_1, \sigma), \alpha_2(x_2, \sigma)), & \sigma \in \Sigma_1 \cap \Sigma_2,\ \alpha_1(x_1, \sigma) \neq \emptyset, \alpha_2(x_2, \sigma) \neq \emptyset \\ (\alpha_1(x_1, \sigma), x_2), & \sigma \in \Sigma_1 - \Sigma_2,\ \alpha_1(x_1, \sigma) \neq \emptyset, \\ (x_1, \alpha_2(x_2, \sigma)), & \sigma \in \Sigma_2 - \Sigma_1,\ \alpha_2(x_2, \sigma) \neq \emptyset, \\ \emptyset, & \text{otherwise.} \end{cases}$$

In other words, the two automata need to be synchronized with respect to all common events $\sigma \in \Sigma_1 \cap \Sigma_2$.

Lemma 5.1. [Cassandras and Lafortune, 2008] Let G_1 and G_2 be two finite automata, if $\Sigma_1 = \Sigma_2$, then $\mathcal{L}(G_1 \| G_2) = \mathcal{L}(G_1) \bigcap \mathcal{L}(G_2)$ and $\mathcal{L}_M(G_1 \| G_2) = \mathcal{L}_M(G_1) \bigcap \mathcal{L}_M(G_2)$.

It is assumed that the plant G is given as a finite automaton and the specification K is regular (i.e., K can be accepted by a finite automaton). In addition, it is assumed that K is controllable and relative closed, so there exists a supervisor $f : \mathcal{L}(G) \to 2^{\Sigma_c}$ such that $\mathcal{L}_M(G^f) = K$. In order for the control achieved by a synchronous composition based supervisor S to be equivalent to the one achieved by a control law based supervisor f, we build S as the state machine that $\mathcal{L}(S) = \mathcal{L}_M(S) = pr(K)$.

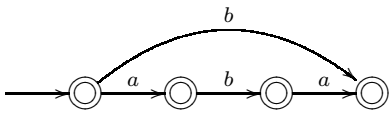
Let $G = (X, \Sigma, \alpha, x^0, X^M)$ and $S = (Y, \Sigma, \beta, y_0, Y^M)$ denote the plant and supervisor respectively. Then, the control loop is closed through connecting S to G by the synchronous composition $G\|S = (Z, \Sigma, \gamma, z_0, Z^M)$. Since $\mathcal{L}(G\|S) = \mathcal{L}(G) \bigcap \mathcal{L}(S)$ and $\mathcal{L}_M(G\|S) = \mathcal{L}_M(G) \bigcap \mathcal{L}_M(S)$, synchronous composition restricts the behavior of the plant. An event σ can occur when $G\|S$ is in the state (x, y) only if σ is possible in both G and S for state x and y respectively. In this case, the control action of S on G is implicit in the transition structure of S. In particular, if $s \in \mathcal{L}(G^f)$ then $s \in \mathcal{L}(S)$, and $s\sigma \in \mathcal{L}(S)$ only if $\sigma \notin f(s)$, which ensures that those transitions disabled by f do not appear in the transition structure of S. In addition, if $s \in \mathcal{L}(G^f)$, $s\sigma \in \mathcal{L}(G)$ and $\sigma \notin f(s)$ then $s\sigma \in \mathcal{L}(S)$, which means that the supervisor S does allow all transitions that is feasible in G and not disabled by f. In this manner, the supervisor S restricts the transitions that can occur in G, and therefore controls the behavior of the closed-loop system. Next, we are going to show the equivalence of control effects between $G\|S$ and the mapping f that has been adopted so far.

$$\begin{aligned} \mathcal{L}(G\|S) &= \mathcal{L}(G) \cap \mathcal{L}(S) = \mathcal{L}(G) \cap pr(K) = pr(K) = \mathcal{L}(G^f) \\ \mathcal{L}_M(G\|S) &= \mathcal{L}_M(G) \cap \mathcal{L}_M(S) = \mathcal{L}_M(G) \cap pr(K) = \mathcal{L}_M(G) \cap \mathcal{L}(G^f) \\ &= \mathcal{L}_M(G^f) \end{aligned}$$

Hence, the closed-loop system $G\|S$ achieves the same control objective as G^f. The representation of a controller as finite automaton has several advantages, e.g., automata representation is more compact than the mapping f which needs to be defined for all strings s in $\mathcal{L}(G)$.

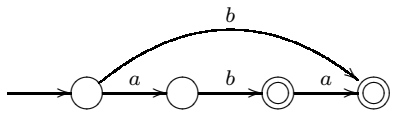
Example 5.6. Consider the previous example, the supervisor f can be realized as a finite automaton

S :



with $\mathcal{L}(S) = \mathcal{L}_M(S) = pr(K) = K = \{\epsilon, b, a, ab, aba\}$. The synchronous composition between S and G can be obtained as:

$G\|S$:



It can be seen that $\mathcal{L}_M(G\|S) = \{b, ab, aba\}$, and $\mathcal{L}(G\|S) = \mathcal{L}(G_f) = K = pr(\mathcal{L}_M(G\|S))$. □

5.1.3 Checking the Existence Condition

Next we illustrate how to check the controllability and relative closure of a given specification language K. Let's assume that K is regular, so there exists a finite automaton $S = (Y, \Sigma, \beta, y_0, Y^M)$ that recognizes K, i.e., $\mathcal{L}_M(S) = K$. Without loss of generality, we assume that $\mathcal{L}(S) = pr(\mathcal{L}_M(S)) = K$.

First, for the *relative closure* condition $K = pr(K) \cap \mathcal{L}_M(G)$, we only need to verify whether $K \supseteq pr(K) \cap \mathcal{L}_M(G)$, since $K \subseteq pr(K) \cap \mathcal{L}_M(G)$ always holds due to the facts that $K \subseteq pr(K)$ and $K \subseteq \mathcal{L}_M(G)$. In order to check whether $K \supseteq pr(K) \cap \mathcal{L}_M(G)$, we consider the product $G\|S$, and it can be seen that the condition $pr(K) \cap \mathcal{L}_M(G) \supseteq K$ if and only if $X^M \times Y \subseteq X^M \times Y^M$. Thus the relative closure of K can be determined in $O(mn)$ time [Cassandras and Lafortune, 2008], where m, n stand for the number of states in G and S respectively.

Next, to test the *controllability* condition, we need to verify whether $pr(K)\Sigma_u \cap \mathcal{L}(G) \subseteq K$. For such a purpose, we compare the active event set of each state of $G\|S$ with the active event set of the corresponding state in G. If there exists an uncontrollable event in the latter that does not appear in the former, then K is not controllable. Thus the controllability of K can be determined in $O(mn)$ time [Ramadge and Wonham, 1989, Cassandras and Lafortune, 2008].

As shown above, given a regular language K, the existence of a supervisor depends on the relative closure and controllability of K. However, many specifications of practical significance may fail to satisfy these conditions. Then, we usually turn to computing the maximum sub-language of K satisfying these properties. Since controllability is closed under union, so there exists a unique supremal controllable sub-language of K, which can be calculated iteratively. Interested readers may refer to [Ramadge and Wonham, 1989, Cassandras and Lafortune, 2008] for more details and more advanced topics such control under partial observations, modular and decentralized supervisory control.

5.2 Timed Language Supervisory Control

In this section, we deal with the supervisory control problem for timed automata, which is known as the timed supervisory control problem. One possible method is to use the finite region transition system for a timed automaton as derived in the previous chapter, based on which the DES supervisory control theory presented in the previous section can be directly applied. Instead, we describe an extension of the Ramadge-Wonham framework of supervisory control over discrete-event systems to timed languages. The concept of controllability and the existence of the maximally permissive supervisor can be suitably generalized by using timed automata as models. The discussion of timed language and supervisory control for timed automata is based on [Wong-Toi and Hoffmann, 1991].

5.2.1 Timed Language

First, we briefly review the formal language theory for timed automata. A time sequence $\bar{\tau} = \tau_1\tau_2\ldots$ is an infinite sequence of time values, $\tau_i \in \mathbb{R}_{\geq 0}$ satisfying:

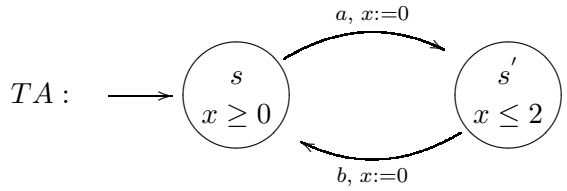
- Monotonicity: $\tau_i \leq \tau_{i+1}, \forall i \geq 1$.
- Progress: $\forall t \geq 0, \exists i \geq 1$ such that $\tau_i > t$.

A timed word over Σ is a pair $(\bar{\sigma}, \bar{\tau})$ where $\bar{\sigma} = \sigma_1\sigma_2...$ is an infinite word over Σ. A timed language T over Σ is a set of timed words over Σ. For example, $T_1 = \{((ab)^\omega, \bar{\tau}) \mid 1 \leq (\tau_{2i} - \tau_{2i-1}) \leq 2, i \geq 1\}$ is a timed language over $\Sigma = \{a, b\}$, and the timed words in T_1 are like $\{(a,\tau_1)(b,\tau_2)(a,\tau_3)(b,\tau_4)\cdots(a,\tau_{2i-1})(b,\tau_{2i})\cdots\}$ with the requirement that once a happens the event b will happen within 1 to 2 time units.

A timed automaton $TA = (Q, Q_0, \Sigma, \mathbb{C}, I, \rightarrow)$ reads a timed word $(\bar{\sigma}, \bar{\tau})$ if the following conditions hold: There is an initial state $q_0 \in Q_0$ such that a run of TA starting from $(q_0, 0)$ and staying at q_0 till τ_1. Then, there exists a transition, $(q_0, \sigma_1, \varphi_1, \lambda_1, q_1) \in \rightarrow$, in TA, with $0 + \tau_1$ satisfying φ_1. After the transition and a clock reset to zero in λ, the state of TA becomes (q_1, x_1), and it stays at q_1 for a du-

ration of length $\tau_2 - \tau_1$, then the event σ_2 occurs and the transition $(q_1, \sigma_2, \varphi_2, \lambda_2, q_2) \in \rightarrow$ is taken with clock reset and so on.

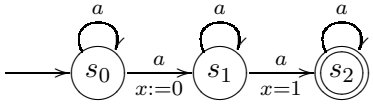
Example 5.7. Consider the timed language T_1. The timed words in T_1 can be accepted by the following timed automaton:



A run corresponding to a timed word $(a, \tau_1)(b, \tau_2)(a, \tau_3)(b, \tau_4) \cdots (a, \tau_{2i-1})\ (b, \tau_{2i}) \cdots$ can be described as below: $(s, 0) \rightarrow^{\tau_1} (s, \tau_1) \dashrightarrow^{a} (s', 0) \rightarrow^{\tau_2 - \tau_1} (s', \tau_2 - \tau_1) \dashrightarrow^{b} (s, 0) \rightarrow \cdots$. Here we use the solid arrow to represent the elapse of time with duration at the superscript, while the dashed arrow is used to represent the transitions caused by an event. □

As we are interested in nonterminating behaviors of timed automata, we hence adopt the Büchi acceptance condition in timed automata, and call them *Büchi timed automata*. A timed word is accepted by a Büchi timed automaton (BTA), if its corresponding run visit the marked states, $F \subseteq Q$, infinitely often. A timed language is timed ω-regular if it is accepted by a Büchi timed automaton. It can be shown that, for a timed ω-regular language T, denoted as T, its un-timed version, denoted as $Untime(T)$ with $Untime(T) = \{\bar{\sigma} \in \Sigma^{\omega} \mid \exists \bar{\tau} \text{ s.t. } (\bar{\sigma}, \bar{\tau}) \in T\}$, is ω-regular. For example, the un-timed version of T_1 is $\{(ab)^{\omega}\}$.

Timed ω-regular language is closed under intersection and union, but it is not closed under complementation. For example, the timed language $T = \{(a^{\omega}, \bar{\tau}) \mid \text{for some } 1 \leq i \leq j,\ \tau_j = \tau_i + 1\}$ is timed ω-regular since it is accepted by a Büchi time automaton given below



However its complement $T^c = \{(a^{\omega}, \bar{\tau}) \mid \forall i, j :\ \tau_j \neq \tau_i + 1\}$ is not timed ω-regular. To see why, we first observe that any timed au-

tomaton accepting T^c needs to remember all the time instances once the event a occurred within the past one time unit since it has to avoid a happening again exactly after one time unit. However, it is possible that a occurs infinite number of times within one time unit, which means that we need infinite number of time clocks to track the occurrence of event a.

5.2.2 Timed Language Supervisory Control

Now we turn to the timed language supervisory control problem, which can be stated as follows. Given a timed automaton $TA = (Q, Q^0, \Sigma, \mathbb{C}, I, \rightarrow, F)$ associated with the accepted timed language $\mathcal{L}(TA)$ and the timed specification $K \subseteq \mathcal{L}(TA)$. Design a supervisor $f : pr(\mathcal{L}(TA)) \times \mathbb{R}_{\geq 0} \rightarrow 2^{\Sigma_c}$, where Σ_c is the set of controllable events, such that $\mathcal{L}(TA^f) = K$, where $\mathcal{L}(TA^f)$, the timed behaviors of the controlled system, is defined as

- $(\epsilon, 0) \in L_0$,
- $(\bar{e}, \bar{\tau})(\sigma, t) \in L_0$, if $(\bar{e}, \bar{\tau}) \in L_0$, $(\bar{e}, \bar{\tau})(\sigma, t) \in pr(\mathcal{L}(TA))$ and $\sigma \notin f((\bar{e}, \bar{\tau}), t)$,
- $\mathcal{L}(TA^f) = \mathcal{L}(TA) \cap L_0$.

Before we establish a necessary and sufficient condition for the existence of a supervisor for a timed automaton, we first extend the concept of the controllability of languages to the case of timed languages.

Definition 5.5. Given a timed automaton TA, a timed language $K \subseteq \Sigma^* \times \mathbb{R}_{\geq 0}$ is said to be *controllable* if

$$pr(K)(\Sigma_u \times \mathbb{R}_{\geq 0}) \cap pr(\mathcal{L}(TA)) \subseteq pr(K).$$

The following theorem states a necessary and sufficient condition for the existence of the supervisor for timed automata, which can be seen as a direct extension of the existence conditions of a supervisor for finite automata given in the previous section.

Theorem 5.3. [Wong-Toi and Hoffmann, 1991] Let TA be a plant, and $K \subseteq \mathcal{L}(TA)$ be the specification. There exists a Σ_u-enabling and

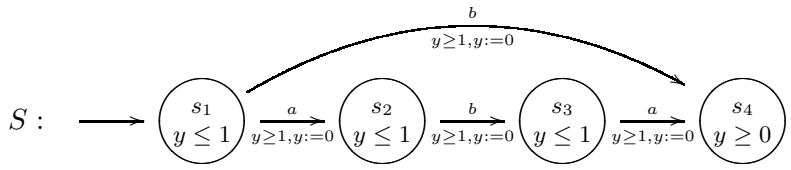
non-marking supervisor f such that $\mathcal{L}(TA^f) = K$ if and only if K is controllable with respect to $\mathcal{L}(TA)$, and K is $\mathcal{L}(TA)$ closed, i. e., $K = pr(K) \cap \mathcal{L}(TA)$.

We illustrate the design of supervisors for timed automata via the following example.

Example 5.8. Consider the previous example. The desired generated behavior

$$K = pr(\{(a^k b a^k, \bar{\tau}) | \tau_i = i, k \leq 1\}) = pr(\{(b,1),(a,1)(b,2)(a,3)\}).$$

It is clear that $K \subseteq \mathcal{L}(TA)$ is $\mathcal{L}(TA)$ closed and controllable. The supervisor can be realized as the following timed automaton:



The supervisory control law $f : \Sigma^* \times \mathbb{R}_{\geq 0} \to 2_c^{\Sigma}$ then is given by

$$f(s,t) = \begin{cases} \{a\} & s = (b,1), t = 2 \\ \{a\} & s = (a,1), t = 2 \\ \{a\} & s = (a,1)(b,2)(a,3) t = 4 \\ \emptyset & otherwise \end{cases}$$

□

In the next section, we will briefly review recent developments in hybrid supervisory control that aims to design hybrid controllers for continuous systems with respect to temporal logic specifications.

5.3 Hybrid Supervisory Control

The supervisory control of hybrid systems using abstracted models has been advocated in the literature since the early 1990's, see e.g., [Lemmon et al., 1999, Koutsoukos et al., 2000] and the references therein. A recent trend is to synthesize hybrid controllers for continuous or hybrid dynamical systems to satisfy complicated temporal

logic specifications. This is also known as the symbolic control problem, and can be formulated as follows. Consider a continuous variable dynamical system

$$\Sigma : \begin{cases} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= g(x(t)) \end{cases} \tag{5.1}$$

where $x(t) \in \mathbb{R}^n$ is the state of the system, $u(t) \in \mathbb{R}^r$ is the control input and $y(t) \in \mathbb{R}^p$ is the observed output of the system.

It is assumed that the initial condition x_0 is restricted to a certain region in the state space $X_0 \subseteq \mathbb{R}^n$. The goal is to design a controller such that the output $y(t)$ generated by the closed-loop system satisfies a given temporal logic specification. For example, one may request that the output $y(t)$ visits certain regions in the output space $\mathbb{R}^p$ with a specific order while avoiding some forbidden regions in $\mathbb{R}^p$.

5.3.1 Temporal Logic over Reals

To capture such requirements in temporal logic, we adopt the temporal logic over reals (RTL) [Reynolds, 2001] and define a set of atomic propositions as $\Pi = \{\pi_0, \pi_1, \cdots, \pi_m\}$, which is a finite set of symbols that label these regions of interest. Furthermore, the denotation $[\![\cdot]\!]$ of each symbol is defined as the region labeled by π, i.e., $[\![\pi_i]\!] \subseteq \mathbb{R}^p$ for any π_i in Π. The atom π_i is true if and only if $y(t)$ is in $[\![\pi_i]\!]$. The symbol π_0 is reserved for initial conditions, $y(0) \in [\![\pi_0]\!]$, and correspondingly the state is started from $x(0) \in X_0 \subseteq \mathbb{R}^n$.

The syntax of the propositional temporal logic over the reals can be formally introduced as below.

Definition 5.6. Let Π be a finite set of atomic propositions, i.e., $\Pi = \{\pi_1, \pi_2, \cdots, \pi_n\}$. The set of all well formed propositional temporal logic (RTL) formulas over the reals are recursively defined from predicates in Π according to the following rules

1. `true`, `false`, and π_i are RTL formulas for all $\pi_i \in \Pi$;
2. if φ_1 and φ_2 are RTL formulas, then $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ and $\neg\varphi_1$ are RTL formulas;

3. if φ_1 and φ_2 are RTL formulas, then $\varphi_1 \mathcal{U} \varphi_2$ and $\varphi_1 \mathcal{R} \varphi_2$ are RTL formulas

Formally, the semantics of RTL formulas are defined over continuous time signals. Given a function $y : \mathbb{R}^+ \to \mathbb{R}^r$, we define $y|_t$ to be the t time shift of y with definition $y|_t(s) = y(s+t)$ for all $s \in \mathbb{R}^+$.

Definition 5.7. Let $y(t)$ be a function $y : \mathbb{R}^+ \to \mathbb{R}^r$, and Π a finite set of atomic propositions, i.e., $\Pi = \{\pi_1, \pi_2, \cdots, \pi_n\}$ with atom mapping $[\![\cdot]\!]$. For $t, s \in \mathbb{R}^+$, the semantics of an RTL formula over Π can be defined as

1. $(y, [\![\cdot]\!]) \models \pi_i$ iff $y(0) \in [\![\pi_i]\!]$;
2. $(y, [\![\cdot]\!]) \models \neg p$ iff $y(0) \notin [\![\pi_i]\!]$;
3. $(y, [\![\cdot]\!]) \models \varphi_1 \wedge \varphi_2$ if $(y, [\![\cdot]\!]) \models \varphi_1$ and $(y, [\![\cdot]\!]) \models \varphi_2$;
4. $(y, [\![\cdot]\!]) \models \varphi_1 \vee \varphi_2$ if $(y, [\![\cdot]\!]) \models \varphi_1$ or $(y, [\![\cdot]\!]) \models \varphi_2$;
5. $(y, [\![\cdot]\!]) \models \varphi_1 \mathcal{U} \varphi_2$ if there exists $t \geq 0$ such that $(y|_t, [\![\cdot]\!]) \models \varphi_2$ and for all s with $0 \leq s \leq t$ we have $(y|_s, [\![\cdot]\!]) \models \varphi_1$;
6. $(y, [\![\cdot]\!]) \models \varphi_1 \mathcal{R} \varphi_2$ if for all $t \geq 0$ it is $(y|_t, [\![\cdot]\!]) \models \varphi_2$ or there exists some s such that $0 \leq s \leq t$ and $(y|_s, [\![\cdot]\!]) \models \varphi_1$.

Intuitively speaking, the formula $\varphi_1 \mathcal{U} \varphi_2$ expresses the property that over the trajectory $y(t)$, φ_1 is true until φ_2 becomes true. On the contrary, the release operator $\varphi_1 \mathcal{R} \varphi_2$ means that φ_2 should hold true and be released when φ_1 becomes true.

Furthermore, we can derive several additional temporal operators such as

- $\Diamond \varphi = \texttt{true}\, \mathcal{U} \varphi$ means that the sub-formula φ *eventually* becomes true for a trajectory $y(t)$;
- $\Box \varphi = \neg \Diamond \neg \varphi$ indicates that φ *always* holds true for $y(t)$;

The following examples from [Fainekos et al., 2009] illustrate some typical control specifications that can be formulated as temporal logic formulas based on the atomic proposition set Π.

- Reachability while avoiding regions: The formula $\neg(\pi_1 \vee \pi_2) \sqcup \pi_3$ represents the requirement that the output finally reaches the region $[\![\pi_3]\!]$ while keeping away from the regions $[\![\pi_1]\!]$ and $[\![\pi_2]\!]$;
- Sequencing: The formula $\Diamond(\pi_1 \wedge \Diamond(\pi_2 \wedge \Diamond\pi_3))$ represents the requirement that the output reaches the regions $[\![\pi_1]\!]$, $[\![\pi_2]\!]$ and $[\![\pi_3]\!]$ in order;
- Coverage: The formula $\Diamond\pi_1 \wedge \Diamond\pi_2 \wedge \Diamond\pi_3$ represents the requirement that the output eventually reaches all the regions $[\![\pi_1]\!]$, $[\![\pi_2]\!]$ and $[\![\pi_3]\!]$ without any particular order;
- Recurrence (Liveness): The formula $\Box(\Diamond\pi_1 \wedge \Diamond\pi_2 \wedge \Diamond\pi_3)$ represents the requirement that the output reaches these regions $[\![\pi_1]\!]$, $[\![\pi_2]\!]$ and $[\![\pi_3]\!]$ infinitely often.

More complicated specifications can be composed from the basic specifications using the logic operators. The goal is to design a hybrid controller such that the trajectories of the closed-loop system satisfy a given specification formula ϕ.

5.3.2 Linear Control Systems

We first consider the case when the dynamical system to be controlled is linear, i.e.,

$$\begin{cases} \dot{x}(t) &= Ax(t) + a + Bu(t) \\ y(t) &= Cx(t) \end{cases} \tag{5.2}$$

where $x(t) \in \mathbb{R}^n$ is the state of the system, $u(t) \in \mathbb{R}^m$ is the control input and $y(t) \in \mathbb{R}^p$ is the observed output of the system, while $A \in \mathbb{R}^{n\times n}$, $a \in \mathbb{R}^n$, $B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{p\times n}$ are state matrices. The results in this sub-section are mainly based on [Habets and van Schuppen, 2004, Kloetzer and Belta, 2008], where it is assumed that all regions of interest are polyhedral sets.

A *polyhedral set* P is a subset of $\mathbb{R}^N$, described by a finite number of linear inequalities. Namely, there exist K non-zero vectors $n_1, \cdots, n_K \in \mathbb{R}^N$ and scalars $\alpha_1, \cdots, \alpha_K \in \mathbb{R}$, such that

$$P = \left\{ x \in \mathbb{R}^N | \forall i = 1, \cdots, K : \; n_i^T x \leq \alpha_i \right\}. \tag{5.3}$$

Each linear inequality $n_i^T x \leq \alpha_i$ forms a half-space, so this is called the *half-space representation* of a polyhedral set P. Correspondingly, the hyperplane formed by $n_i^T x = \alpha_i$ is called a *supporting hyperplane* for P. A bounded polyhedral set is called a *polytope*. A polytope can alternatively be characterized as the convex hull of a finite number of points, $v_1, \cdots, v_M \in \mathbb{R}^N$,

$$P = \left\{ x \in \mathbb{R}^N | x = \sum_{i=1}^{M} \lambda_i v_i, \sum_{i=1}^{M} \lambda_i = 1, \lambda_i \geq 0 \right\} \tag{5.4}$$

where $v_1, \cdots, v_M$ are called vertices of the polytope P. This is called the vertex representation of a polytope.

The half-space representation and the vertex representation for a polytope are equivalent and can be transformed from one to the other [Ziegler, 1995]. Which definition should be employed depends on what is more suitable for the problem at hand. The intersection of a polytope P with one of its supporting hyperplanes

$$F_i = \{ x \in \mathbb{R}^N | n_i^T x = \alpha_i \} \cap P \tag{5.5}$$

is called a facet of P, if the dimension of the intersection is equal to $N-1$. The vector n_i is the normal vector of the facet F_i, $(i = 1, \cdots, K)$, and, by convention, n_i is of unit length and always points out of the polytope P. Also, the corresponding hyperplane $n_i^T x = \alpha_i$ is called a *bounding hyperplane* for P. Later on, we assume that each facet of P corresponds to exactly one n_i, and vice versa. Namely, each n_i corresponds to a bounding hyperplane and is linearly independent of other norm vectors.

The set of vertices of a polytope P is denoted by $\mathcal{V}(P)$. Given a vertex $v \in \mathcal{V}(P)$, we denote by $\mathcal{F}(v)$ the set of all facets containing v. It can be shown that for a polytope there are at least $N+1$ vertices, i.e. $M \geq N+1$. A polytope that has exactly $N+1$ vertices is called a *simplex* in $\mathbb{R}^N$. Any full dimensional polytope can be triangularized [Lee, 1997]. In other words, for any full dimensional polytope P, there exists full dimensional simplex $S_1, \cdots, S_L$ such that: (1) $P = \cup_{i=1,2,\cdots,L} S_i$, (2) $S_i \cap S_j$ is either empty or a common facet of S_i and S_j, for all $i, j = 1, \cdots, L$ with $i \neq j$, (3) The set of vertices of simplex S_i is a subset of $\{v_1, v_2, \cdots, v_M\}$ for all $i = 1, \cdots, L$.

A reason for restricting polytopes is due to the fact that an affine function's value inside a polytope P will be uniquely determined by its values at the vertices of P. An affine function $f : \mathbb{R}^N \to \mathbb{R}^r$ is of the form

$$f(x) = Ax + b \tag{5.6}$$

for constant matrices $A \in \mathbb{R}^{r\times N}$ and vector $b \in \mathbb{R}^r$. Due to convexity and linearity, it is easy to obtain:

Lemma 5.2. Let $w \in \mathbb{R}^r$ and $d \in \mathbb{R}$. Then $w^T f(x) > d$ everywhere in a polytope P, if and only if the inequality holds at all the vertices of P, i.e., $w^T f(v_i) > d$ for $i = 1, \cdots, N+1$.

It is easy to see that the result remains valid if $>$ is replaced by $\geq, =, <, \leq$. Also, the result remains valid if f is restricted to a facet F_i. The following result tells us that the function value of an affine function on polytope P is completely determined by the values of f on the vertices of P.

Lemma 5.3. [Habets and van Schuppen, 2004] Let P be a polytope in $\mathbb{R}^N$ and $f : \mathbb{R}^N \to \mathbb{R}^r$ an affine function. The function value of $x \in P$, $f(x)$ is completely determined by the values of f on the vertices of P, $f(v_i) = g_i$, $i = 1, \cdots, M$, i.e., $x \in P \Rightarrow f(x) = \sum_{v\in\mathcal{V}(P)} \lambda_v f(v), \quad \sum_{v\in\mathcal{V}(P)} \lambda_v = 1,\ \lambda_v \geq 0$.

Moreover, f is unique. Assume not, and there is another affine function $f' : \mathbb{R}^N \to \mathbb{R}^r$ satisfying $f'|_{\mathcal{V}(P)} = g$. Then, consider $f - f'$, which is affine and $(f - f')|_{\mathcal{V}(P)} = g - g = 0$. It then follows from the previous lemma that $f - f'$ is the function identically zero and thus $f = f'$. In particular, we can present an explicit reconstruction for the case of an affine function on a simplex.

Lemma 5.4. [Habets and van Schuppen, 2004] Let S_N be a simplex in $\mathbb{R}^N$ and $f : \mathbb{R}^N \to \mathbb{R}^r$ an affine function. The restriction of f to S_N is a convex combination of its values at the vertices, and is given by

$$f(x) = GW^{-1} \begin{bmatrix} x \\ 1 \end{bmatrix},\ x \in P \tag{5.7}$$

where

$$G = \begin{bmatrix} g_1 & \cdots & g_{N+1} \end{bmatrix}$$
$$W = \begin{bmatrix} v_1 & \cdots & v_{N+1} \\ 1 & \cdots & 1 \end{bmatrix}$$

are $r \times (N+1)$ and $(N+1) \times (N+1)$ real matrices respectively.

For the linear control system (5.2), it is assumed that all the regions are given as polytopes with $[\![\pi_i]\!] = \{y \in \mathbb{R}^p | n_i^T y \leq g_i\}$, where $n_i \in \mathbb{R}^p$ and $g_i \in \mathbb{R}$. Note that $y(t) \in [\![\pi_i]\!]$ holds true if and only if $n_i^T Cx(t) \leq g_i$, which corresponds to a polytope in the state space $P_i = \{x \in \mathbb{R}^n | n_i^T Cx \leq g_i\}$.

We restrict our control $u(t)$ to be in the form of an affine function of state, i.e., $u(t) = kx(t) + v$, where k and v are to be designed. The closed-loop dynamics will be

$$\dot{x}(t) = (A + Bk)x(t) + w, \tag{5.8}$$

which has an affine function on its right hand side. Our task is to design the controller u, i.e., k and v, in such a way that all trajectories of the closed-loop system starting from a polytope P_i can either all stay inside P_i or all transit to a neighboring polytope P_j after a finite period of time without intersecting with another regions during this transition process. The closed-loop system becomes a piecewise affine system with polytopes as the partition of the state space, which represents a special class of a hybrid system.

For each polytope, the following two problems are considered:

- *Invariant Control Problem:* The invariant control problem for the linear control system (5.2) with respect to a polytope P seeks an affine feedback control law $u = kx + v$ such that all trajectories for the closed-loop system $\xi(t)$ starting from P will remain in P forever, i.e., $\xi(0) \in P \Rightarrow \xi(t) \in P$, $\forall t \in \mathbb{R}^+$.

- *Control to Facet Problem:* Consider the linear control system (5.2) on a polytope P, and let F be a facet of P. The control to facet problem is to determine whether there exists an affine feedback

control law $u = kx + v$ such that all trajectories for the closed-loop system starting from P will leave P through F after a finite time τ. Namely, there exists a finite escape time $\tau > 0$ such that the following hold

- $\xi(t) \in P$ for $0 \leq t < \tau$,
- $\xi(\tau) \in F$,
- $\exists \epsilon > 0$ such that $\xi(t) \notin P \cup F$ for $\tau < t < \tau + \epsilon$.

Since the value of an affine function in a polytope can be determined by its values at the vertices of the polytope, the existence of such an affine feedback can be determined by some linear inequalities evaluated at (a finite number of) vertices of the polytope P. The following proposition characterizes all affine vector fields for which the polytope is an invariant.

Theorem 5.4. [Habets and van Schuppen, 2004, Kloetzer and Belta, 2008] The invariant control problem for the linear control system (5.2) with respect to a polytope P is solvable provided that the following sets are non-empty

$$U_P(v_i) = \bigcap_{F \in \mathcal{F}(v_i)} \{u \in \mathbb{R}^m | \ \eta_F^T(Av_i + a + Bu) < 0\}$$

for all $v_i \in \mathcal{V}(P)$, where η_F is the normal vector for the facet F.

It is a very intuitive condition, and basically requests the existence of a control signal to make the vector field point inside to the polytope P for all vertices. The conditions form a collection of linear inequalities and can be easily checked. Moreover, when the sets are nonempty, a multi-affine control law $u = kx+v$ can be constructed, with the control value at vertex v_i being any element in $U_P(v_i)$. Similarly, the control to facet problem also admits a simple solution.

Theorem 5.5. [Habets and van Schuppen, 2004, Kloetzer and Belta, 2008] Let P be a polytope in $\mathbb{R}^n$ with a facet F and $\dot{x} = Ax + a + Bu$ be

a linear control system. The control to facet problem admits a solution if the following sets are non-empty

$$U_P(v_i) = \bigcap_{G \in \mathcal{F}(v_i)} \left\{ u \in \mathbb{R}^m | \eta_G^T (Av_i + a + Bu) < 0 \right\}$$

for all $v_i \in \mathcal{V}(P)$ such that $F \notin \mathcal{F}(v_i)$, and

$$U_P(v_i) = \bigcap_{G \in \mathcal{F}(v_i), G \neq F} \left\{ u \in \mathbb{R}^m \left| \begin{array}{l} \eta_G^T (Av_i + a + Bu) < 0 \\ \eta_F^T (Av_i + a + Bu) > 0 \end{array} \right. \right\}$$

for all $v_i \in \mathcal{V}(P)$ such that $F \in \mathcal{F}(v_i)$.

Intuitively speaking, the sufficient conditions above forces the vector fields of the closed-loop system point inside to P for all vertices except the vertices of the facet F. It therefore guarantees that all trajectory starting from P will exit P through the facet F within finite time durations. Hence, such a control solves the "Control to Facet Problem."

Once the above two control problems are solved, we can deduce a finite labeled transition system that is bisimilar to the closed-loop continuous system (seen as an infinite state transition system with label mapping consistent with $[\![\cdot]\!]$). The construction of the finite abstraction is conceptually simple. Note that the specification regions $[\![\pi_i]\!]$ for the output $y(t)$ are assumed to be polytopes and mutually exclusive and only share facets if adjacent. Correspondingly, $[\![\pi_i]\!]$ will imply a collection of polyhedrons in R^n, denoted as P_i. All states in P_i are labeled with π_i for consistency. The collection of all such P_i forms the set of discrete states, while the initial state is the polytope containing x_0, i.e., $x_0 \in P_0$. There is a transition from P_i to P_j, i.e., $(P_i, P_j) \in \rightarrow_P$ (assume $i \neq j$), if they share a facet F and the control to the facet F is solvable for P_i. There is a self-loop transition for a polytope P_i, i.e., $(P_i, P_i) \in \rightarrow_P$, if the invariant control problem is feasible for the polytope P_i. Hence, we obtain a transition system $T_P = (\{P_i\}, \{P_0\}, \rightarrow_P)$, which has finite states. The symbol π_i is then used to label the discrete state P_i. It is not difficult to see that the deduced labeled transition system is bisimilar to the continuous control system (5.2). Actually,

the relation $R \subseteq \{P_i\} \times \mathbb{R}^n$, defined by $(P_i, x) \in R$ if $x \in P_i$, together with its reverse, forms a bisimulation relation between T_P and (5.2).

After obtaining the finite abstraction model T_P, one can apply the supervisory control design theory in Section 5.1 with respect to T_P, or use model checking methods to design a sequence of region transitions such that the required RTL specifications are satisfied, see e.g., [Kloetzer and Belta, 2008]. Due to bisimulation relation between the linear control system and the abstracted model, the sequence of discrete region transitions can be mimicked by continuous trajectories $x(t)$, i.e., there exist continuous control signals to drive the output $y(t)$ so to satisfy the RTL specifications. Furthermore, continuous control signals can be designed based on Theorem 5.4 and 5.5. For example, if a self-loop transition for region π_i occurs, then the invariant control law for region P_i can be designed based on results in Theorem 5.4. On the other hand, if there is a transition from region π_i to π_i, the control law proposed in Theorem 5.5 can be adopted to achieve the region transition.

5.3.3 Multi-affine Control Systems

Next we consider the symbolic control problem for a class of nonlinear dynamics

$$\dot{x}(t) = f(x(t), u(t)) = g(x) + Bu \tag{5.9}$$

where $g(x) : \mathbb{R}^n \to \mathbb{R}^n$ is assumed to be multi-affine, and $B \in \mathbb{R}^{n \times m}$ is constant.

A map $f : \mathbb{R} \to \mathbb{R}$ is said to be *affine* when for every $x, y \in \mathbb{R}$ and for every $\alpha, \beta \in \mathbb{R}$ satisfying $\alpha + \beta = 1$ the equality $f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$ holds. The number $\alpha x + \beta y$ is said to be an affine combination of x and y when $\alpha + \beta = 1$. The constraint $\alpha + \beta = 1$ can be expressed as a single variable $\lambda \in \mathbb{R}$ to define the affine combination as $\lambda x + (1 - \lambda)y$.

Definition 5.8. [Belta and Habets, 2006] A map $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *multi-affine* when its projection to each x_i, $i = 1, \cdots, n$, is affine.

A map $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be *multi-affine* if for each $i = 1, \cdots, m$ the map $f_i : \mathbb{R}^n \to \mathbb{R}$ is multi-affine.

In other words, when we fix all variables except x_i, the map becomes affine. More specifically, the multi-affine map f consists of the sum of polynomials in the indeterminates $x_1, \cdots, x_n$, with the property that the degree of any of the indeterminates $x_1, \cdots, x_n$ is less than or equal to 1. State differently, f has the form

$$f(x) = \sum_{i_1,\cdots,i_n \in \{0,1\}} c_{i_1,\cdots,i_n} x_1^{i_1} \cdots x_n^{i_n},$$

where $c_{i_1,\cdots,i_n} \in \mathbb{R}$ for all $i_1, \cdots, i_n \in \{0, 1\}$ and using the convention that if $i_k = 0$, then $x_k^{i_k} = 1$.

It is also assumed that the regions of interest are all rectangles.

Definition 5.9. A n-rectangle E in $\mathbb{R}^n$ is a set defined by: $E = \prod_{i=1}^{n}(a_i, b_i)$, where a_i, $b_i \in \mathbb{R}$ satisfying $a_i < b_i$ for $i = 1, \cdots, n$.

The set of vertices of an n-rectangle is denoted by $\mathcal{V}(E)$ and defined by $\mathcal{V}(E) = \{x \in \mathbb{R}^n | x_i \in \{a_i, b_i\}\}$. The facet of an n-rectangle E is the intersection of the closure of E, $\bar{E}$, with the hyperplane defined by $x_i = a_i$ or $x_i = b_i$. Given a vertex $v \in \mathcal{V}(E)$, we denote by $\mathcal{F}(v)$ the set of all facets containing v.

The following theorem tells us that the function value of a multi-affine function on an n-rectangle E is completely determined by the values of f on the vertices of E.

Theorem 5.6. [Belta and Habets, 2006] Let E be an n-rectangle in $\mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^m$ a multi-affine function. The function value of $x \in E$, $f(x)$ is completely determined by the values of f on the vertices of E, i.e.,

$$x \in E \Rightarrow f(x) = \sum_{v \in \mathcal{V}(E)} \lambda_v f(v), \quad \sum_{v \in \mathcal{V}(E)} \lambda_v = 1.$$

Furthermore, the function f constructed is multi-affine and is unique.

The multi-affine control problem is to design (multi-affine state feedback) control laws $u(t)$ such that its closed loop system state trajectories $x(t)$ (with proper initial conditions) satisfies a give RTL formula ϕ. It is assumed that the regions corresponding to the atomic propositions in RTL are all bounded rectangles in $\mathbb{R}^n$. The reason for

restricting the control law $u(t)$ to be a multi-affine function of $x(t)$, i.e., $u(t) = h(x(t))$ with $h(x)$ a multi-affine function, is to make the closed-loop system $\dot{x} = g(x) + Bh(x)$ remain to be multi-affine so to take advantage of the nice properties of multi-affine functions described in Theorem 5.6.

The basic idea for multi-affine control design is very similar to the case of linear control systems over polytopes. It basically relies on the nice properties of multi-affine functions (Theorem 5.6) to obtain symbolic abstractions based on partitions of state space induced by n-rectangles. Similarly, we also try to make an n-rectangle E either invariant or all trajectories leaving E through the same facet. Recall that the facet of E is given by either $x_i = a_i$ or $x_i = b_i$. Moreover, to each facet F defined by $x_i = a_i$ we associate a normal vector η_F defined by $\eta_{Fj} = 0$ for $j \neq i$ and $\eta_{Fj} = -1$ for $j \neq i$. Contrarily, the normal vector η_F for facet defined by $x_i = b_i$ is given as $\eta_{Fj} = 0$ for $j \neq i$ and $\eta_{Fj} = 1$ for $j \neq i$.

For each n-rectangle, the following two problems are considered:

- *Invariant Control Problem:* The invariant control problem for a multi-affine control system $\dot{x} = g(x) + Bu$ on an n-rectangle E seeks a multi-affine feedback control law $k : \mathbb{R}^n \to \mathbb{R}^m$ such that all trajectories for the closed-loop system $\xi(t)$ starting from E will remain in E forever, i.e., $\xi(0) \in E \Rightarrow \xi(t) \in E,\ \forall t \in \mathbb{R}^+$.

- *Control to Facet Problem:*Consider a multi-affine control system $\dot{x} = g(x) + Bu$ on an n-rectangle E, and let F be a facet of E. The control to facet problem is to determine whether there exists a multi-affine feedback control law $k : \mathbb{R}^n \to \mathbb{R}^m$ such that all trajectories for the closed-loop system starting from E will leave E through F after a finite time τ. Namely, there exists a finite escaping time $\tau > 0$ such that the following hold

  - $\xi(t) \in E$ for $0 \leq t < \tau$,
  - $\xi(\tau) \in F$,
  - $\exists \epsilon > 0$ such that $\xi(t) \notin E \cup F$ for $\tau < t < \tau + \epsilon$.

Similar to the linear control systems, we have the following results for multi-affine control systems.

Theorem 5.7. [Belta and Habets, 2006, Tabuada, 2009]Let E be an n-rectangle in $\mathbb{R}^n$ and $\dot{x} = g(x) + Bu$ be a multi-affine control system. The rectangle invariant problem admits a solution if the following sets are non-empty

$$U_E(v) = \bigcap_{F \in \mathcal{F}(v)} \{u \in \mathbb{R}^m | \eta_F^T(g(v) + Bu) < 0\}$$

for all $v \in \mathcal{V}(E)$.

If the sets are non-empty, a multi-affine control law k can be constructed by the theorem and with the control value at vertex v being any element in $U_E(v)$.

Theorem 5.8. [Belta and Habets, 2006, Tabuada, 2009]Let E be an n-rectangle in $\mathbb{R}^n$ with a facet F and $\dot{x} = g(x) + Bu$ be a multi-affine control system. The control to facet problem admits a solution if the following sets are non-empty

$$U_E(v) = \bigcap_{G \in \mathcal{F}(v)} \left\{ u \in \mathbb{R}^m \;\middle|\; \eta_G^T(g(v) + Bu) < 0 \right\}$$

for all $v \in \mathcal{V}(E)$ such that $F \notin \mathcal{F}(v)$, and

$$U_E(v) = \bigcap_{G \in \mathcal{F}(v), G \neq F} \left\{ u \in \mathbb{R}^m \;\middle|\; \begin{array}{l} \eta_G^T(g(v) + Bu) < 0 \\ \eta_F^T(g(v) + Bu) > 0 \end{array} \right\}$$

for all $v \in \mathcal{V}(E)$ such that $F \in \mathcal{F}(v)$.

The above results enable us to construct finite transition systems that are bisimilar abstractions for multi-affine control systems. To obtain the abstraction, we use n-rectangle as states and we place a transition from a rectangle E to itself when the invariant control problem has a solution for E, and a transition from E to E' is added when E and E' shares a common facet F and the control to facet problem is solvable for E with respect to the common facet F.

5.3.4 Nonlinear Control Systems

So far, we have considered special cases when the continuous dynamics are linear or multi-affine control systems. In this subsection, we consider a general nonlinear control system

$$\Sigma : \begin{cases} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= g(x(t)) \end{cases} \tag{5.10}$$

where $x(t) \in \mathbb{R}^n$ is the state of the system, $u(t) \in U \subseteq \mathbb{R}^r$ is the control input and $y(t) \in \mathbb{R}^p$ is the observed output of the system.

Once again, the goal is to design a controller such that the output $y(t)$ generated from the closed-loop system satisfies a given temporal logic specification ϕ. It is also assumed that ϕ is build from atomic propositions, $\Pi = \{\pi_0, \pi_1, \cdots, \pi_m\}$, where each $[\![\pi_i]\!]$ stands for a regions in concern. Also, it is assumed that the region $[\![\pi_i]\!]$ is bounded and convex.

The development here follows the results in [Fainekos et al., 2009], and the basic idea is to introduce a simpler linear model in $\mathbb{R}^p$, for which we design a hybrid controller to achieve a modified version of the specification ϕ. Then, a controller is designed for the original nonlinear system to keep its trajectory always within a neighborhood of the trajectory from the designed linear systems provided that their initial conditions are close enough. Note that the modification on the specification ϕ is made in such a way that once the trajectories are close enough to a trajectory satisfying the modified version of ϕ, then the original specification ϕ holds true for all these nearby trajectories. For such a purpose, we under-approximate the region $[\![\pi_i]\!]$ using a polytope when it needs to be reached by $y(t)$, otherwise over-approximate it using another polytope. Based on these approximations, we introduce new atomic propositions, $\tilde{\pi}_i$, and construct a new version of the specification ϕ, called $\tilde{\phi}$. Then, based on the previous section results of controlling a linear control system over polytopes, we can design controllers and successful runs for proper initial conditions. Finally, we design controllers for the nonlinear system such that its output $y(t)$ tracks the trajectory of the closed-loop linear system with specified bounded errors. Then, The resulting output trajectory $y(t)$ is guaran-

teed to satisfy the initial user specification.

Next, we introduce briefly the tracking problem and modifications on the temporal logic specifications. First, the nonlinear control system (5.10) is abstracted to a linear control system:

$$\Sigma' : \ \dot{z}(t) = Az(t) + Bv(t),\ z(t) \in \mathbb{R}^p,\ z_0 \in [\![\pi_0]\!], v \in V. \tag{5.11}$$

We would like the linear control system approximates the trajectories of the nonlinear control system (5.10) in the following sense.

Definition 5.10. [Girard and Pappas, 2005] A relation $\mathcal{W} \subseteq \mathbb{R}^p \times \mathbb{R}^n$ is an approximate simulation relation of precision δ of Σ' by Σ if for all $(z_0, x_0) \in \mathcal{W}$,

1. $\|z_0 - g(x_0)\| \leq \delta$
2. For all state trajectories $z(t)$ of Σ' such that $z(0) = z_0$ there exists a state trajectory $x(t)$ of Σ such that $x(0) = x_0$ and satisfies $(z(t), x(t)) \in \mathcal{W}$ for all $t \geq 0$.

An interface associated with the approximation simulation relation $\mathcal{W}$ allows us to choose the control inputs for the nonlinear control system (5.10) so that the states in the linear control system (5.11) and the states of the nonlinear control system (5.10) remain in $\mathcal{W}$.

Definition 5.11. [Fainekos et al., 2009] A continuous function $u_{\mathcal{W}} : V \times \mathcal{W} \to U$ is an interface associated with the approximate simulation relation $\mathcal{W}$, if for all $(z_0, x_0) \in \mathcal{W}$, for all trajectories $z(t)$ of Σ' associated to input $v(t)$ such that $z(0) = z_0$, the trajectory of Σ given by

$$\dot{x}(t) = f(x(t), u_{\mathcal{W}}(v(t), z(t), x(t))),\ x(0) = x_0 \tag{5.12}$$

satisfies for all $t \geq 0$, $(z(t), x(t)) \in \mathcal{W}$.

It is clear from the definitions that interconnecting the linear control system (5.11) and the nonlinear control system (5.10) through the interface $u_{\mathcal{W}}$

$$\begin{cases} \dot{x}(t) &= f(x(t), u_{\mathcal{W}}(v(t), z(t), x(t))),\ x(0) = x_0 \\ y(t) &= g(x(t)) \end{cases}$$

satisfies for all $t \geq 0$, $\|y(t) - z(t)\| \leq \delta$ provided $\|g(x_0) - z_0\| \leq \delta$.

The approximate simulation relation can be constructed by the level sets of a simulation function, which is a positive function bounding the distance between the observations and non-increasing under parallel evolution of the systems.

Definition 5.12. [Girard and Pappas, 2005] Let $\mathcal{V} : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}^+$ be a continuous and piecewise differentiable function. Let $u_{\mathcal{V}} : V \times \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}^p$ be continuous function. The function $\mathcal{V}$ is a simulation function of Σ' by Σ, and $u_{\mathcal{V}}$ is an associated interface if for all $(z, x) \in \mathbb{R}^p \times \mathbb{R}^n$,

$$\mathcal{V}(z, x) \geq \|z - g(x)\|^2, \tag{5.13}$$

$$\sup_{v \in \mathcal{V}} \left(\frac{\partial \mathcal{V}(z,x)}{\partial z}(Ax + Bv) + \frac{\partial \mathcal{V}(z,x)}{\partial x} f(x, u_{\mathcal{V}}(v, z, x)) \right) \leq 0 \tag{5.14}$$

Then, the approximate simulation relation can be defined as level sets of the simulation function.

Theorem 5.9. [Girard and Pappas, 2005, Fainekos et al., 2009] Let the relation $\mathcal{W} \subseteq \mathbb{R}^p \times \mathbb{R}^n$ be given by

$$\mathcal{W} = \{(z, x) | \mathcal{V}(z, x) \leq \delta^2\}. \tag{5.15}$$

If for all $v \in V$, for all $(z, x) \in \mathcal{W}$, $u_{\mathcal{V}}(v, z, x) \in U$, then $\mathcal{W}$ is an approximate simulation relation of precision δ of Σ' by Σ and $u_{\mathcal{W}} : V \times \mathcal{W} \to U$ given by $u_{\mathcal{W}}(v, z, x) = u_{\mathcal{V}}(v, z, x)$ is an associated interface.

Usually it is not easy to find such a simulation function except in some special cases. Also, the arguments used by the simulation function is similar to Lyapunov functions, so it is usually very conservative.

In our setup, an RTL formula ϕ is provided as a controller specification for the original nonlinear control system, while we need to deduce a new RTL formula for the auxiliary linear system such that once a trajectory satisfies the modified specification all neighboring trajectories will satisfy the original specification ϕ. Hence, we introduce the notation of δ-contraction so as to capture the robustness of satisfaction for a formula.

Definition 5.13. [Fainekos et al., 2009] Given a radius $\delta \in \mathbb{R}^+ \cup \{+\infty\}$ and a point α in a normed space A, the δ-ball centered at α is defined as $B_\delta(\alpha) = \{\beta \in A |\ \|\alpha - \beta\| \leq \delta\}$. If $\Gamma \subseteq A$, then

$$C_\delta(\Gamma) = \{\alpha \in A | B_\delta(\alpha) \subseteq \Gamma\}$$

is the δ-contraction and $B_\delta(\Gamma) = \{\alpha \in A | B_\delta(\alpha) \cap \Gamma \neq \emptyset\}$ is the δ-expansion.

Now, we define a new set of atomic propositions

$$\tilde{\Pi} = \{\xi_\alpha | \alpha = \pi \text{ or } \neg\pi \text{ for } \pi \in \Pi\}\,.$$

Next, we describe how to translate an RTL ϕ on Π into a new RTL, denoted as $\mathbf{rob}(\phi)$, on $\tilde{\Pi}$. First, we write ϕ into the Negation Normal Form (NNF). Second, replace the occurrence of atomic proposition π and $\neg\pi$ with ξ_π and $\xi_{\neg\pi}$ respectively. Thirdly, we define a new atomic map $[\![\cdot]\!]_\delta$ as follows:

$$\forall \xi \in \tilde{\Pi},\ [\![\xi]\!]_\delta = \begin{cases} C_\delta([\![\pi]\!]^c) & \text{if } \xi = \xi_{\neg\pi} \\ C_\delta([\![\pi]\!]) & \text{if } \xi = \xi_{\neg\pi} \end{cases}, \tag{5.16}$$

where $\delta \in \mathbb{R}^+$ is a given positive scalar, and $[\![\pi]\!]^c$ stands for the complement of a the set $[\![\pi]\!]$.

Intuitively, it means that we expand the region that $y(t)$ must avoid and δ-contract the region that it needs to reach. The following result tells us that the trajectory satisfies the δ-robust specification, then any other trajectories that remain δ-close to the initial one will satisfy ϕ.

Theorem 5.10. [Fainekos et al., 2009] Consider a formula $\phi \in \Phi_\Pi$, which is built on a set of atoms Π, a map $[\![\cdot]\!] : \Pi \to \mathcal{P}(\mathbb{R}^p)$, and a number $\delta \in \mathbb{R}^+$, then for all functions $y(t)$ and $z(t)$ from $\mathbb{R}^+$ to $\mathbb{R}^p$ such that for all $t \geq 0$, $\|z(t) - y(t)\| \leq \delta$, it holds that $(z, [\![\cdot]\!]_\delta) \models \mathbf{rob}(\phi) \Rightarrow (y, [\![\cdot]\!]) \models \phi$.

Then, one can design a hybrid controller for the linear control system to satisfy $\mathbf{rob}(\phi)$ as introduced in Section 5.3.1, i.e., the closed-loop trajectory $z(t)$ satisfies $\mathbf{rob}(\phi)$. Once this is done, the remaining task is to design the interface so that the trajectory $y(t)$ always stay in the δ neighborhood of $z(t)$.

5.4 Notes and Further Reading

The supervisory control theory was developed by Ramadge and Wonham in the 1980's [Ramadge and Wonham, 1987, 1989], and has seen significant growth in 1990's [Cassandras and Lafortune, 2008]. The discussion on the timed language and supervisory control for timed automata is based on [Wong-Toi and Hoffmann, 1991]. Related works on timed language supervisory control include [Brandin and Wonham, 1994, Maler et al., 1995].

Reachability control on simplex for linear and affine dynamics follows the work in [Habets and van Schuppen, 2004]. The results on multi-affine systems are mainly based on [Belta and Habets, 2006]. The concept of approximate bisimulation was proposed in [Girard and Pappas, 2007]. Our treatment of the synthesis of hybrid controllers for nonlinear systems mainly follows the results in [Girard and Pappas, 2009]. Although we only considered the symbolic control for continuous control systems, the extension of the idea to the cases of hybrid systems is not difficult provided that the regions of concern, such as invariant sets and guard sets, are all assumed to be polyhedrons or rectangles (for multi-affine dynamics). For example, the control problems for rectangular multi-affine hybrid systems were investigated in [Habets et al., 2006b] using similar ideas described in Section 5.3.2, while the reachability and control problems for hybrid systems with piecewise affine dynamics defined on simplices were considered in [Habets et al., 2006a] using the techniques discussed in Section 5.3.1. The finite quotient transition systems obtained for corresponding hybrid systems in [Habets et al., 2006b,a] can also be used for more general supervisory control synthesis using the techniques introduced Section 5.1.

The supervisory control of hybrid systems using abstracted models has been advocated in the literature since early 1990's, see e.g., [Lemmon et al., 1999, Cury et al., 1998, Raisch and O'Young, 1998, Koutsoukos et al., 2000] and the references therein. Early work in the area of hybrid supervisory control, e.g., [Cury et al., 1998, Raisch and O'Young, 1998, Koutsoukos et al., 2000], mainly performed the supervisor synthesis with respect to regular language

specifications (see Section 5.1) based on language equivalent or approximating quotient systems. However, language equivalence does not guarantee branching logic, such as CTL, specifications, so we adopt a stronger equivalence condition, namely bisimulation, instead.

Symbolic control methods using simulation or approximate simulation based quotient transition systems have been exploited in the literature and the trend is to design controllers in an automatic way, see e.g., [Tabuada, 2008, Kloetzer and Belta, 2008]. As an application of the theory introduced here, symbolic motion planning for robots has been considered in the literature, see e.g., [Belta et al., 2007, Fainekos et al., 2009]. Readers who are interested in symbolic control may refer to the book [Tabuada, 2009] for more comprehensive and detailed discussions on this topic. The design methods introduced in [Tabuada, 2009] also include reactive synthesis based on similarity games and fixed-point computation, which are rooted in the computer science literature, see e.g., [Thomas, 1995], and form a parallel approach to the supervisory control theory discussed here.

6

Concluding Remarks

This paper gives an overview of the theory on hybrid dynamical systems. This is a very ambitious goal as hybrid systems literature is very dynamic with a wide spectrum of approaches and applications ranging from computer science, mathematics and control theory. It is perfectly possible, and perhaps probable, that we have missed important results in spite of our best efforts. If this has happened, we do apologize.

At the same time, this paper aims to serve as an introductory article to the field of hybrid systems for readers with various backgrounds. Hybrid system theory offers a critical piece of foundation and a unified theoretical framework for cyber-physical systems as it helps to better understand and design the interaction of discrete logic (arising from cyber part) and continuous physical dynamics. For such a purpose, a variety of basic materials are included to help readers understand the key ideas in hybrid systems.

Finally, we would like to draw attention to the similarity between hybrid theory literature and a box of tools. There are many kinds of models for hybrid systems and their corresponding analysis and design methods. Hybrid automata is very general and convenient

to model a wide variety of dynamical systems. Unfortunately, many problems become undecidable and hence untractable if one uses exclusively with the hybrid automata framework. Some of the models may be very specific to certain classes of dynamics, but efficient analytical or computational tools exist. So, which modeling framework should be followed depends on the problem at hand.

Acknowledgements

We highly appreciate the constructive and helpful comments from four anonymous reviewers. The financial supports of NSF-CNS-1239222 and NSF-CAREER-1253488 for this work are greatly acknowledged.

References

R. Alur and D. Dill. The theory of timed automata. *Theoretical Computer Science*, 126:193–235, 1994.

R. Alur and T. Henzinger. Modularity for timed and hybrid systems. In A. Mazurkiewicz and J. Winkowski, editors, *CONCUR '97: Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, pages 74–88. Springer Berlin Heidelberg, 1997. ISBN 978-3-540-63141-5.

R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Logic in Computer Science, 1990. LICS '90*, pages 414–425, Jun 1990.

R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

R. Alur, T. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. In P. J. Antsaklis, editor, *Proceedings of the IEEE: Special issue on hybrid systems*, volume 88, pages 971–984. IEEE Press, 2000a.

Rajeev Alur, Costas Courcoubetis, ThomasA. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In RobertL. Grossman, Anil Nerode, AndersP. Ravn, and Hans Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer Berlin Heidelberg, 1993. ISBN 978-3-540-57318-0.

Rajeev Alur, Radu Grosu, Yerang Hur, Vijay Kumar, and Insup Lee. Modular specification of hybrid systems in charon. In *Hybrid Systems: Computation and Control*, pages 6–19. Springer, 2000b.

P. J. Antsaklis and A. Nerode. Hybrid control systems: An introductory discussion to the special issue. *IEEE Trans. Automat. Contr.*, 43(4):457–460, 1998.

Panos J Antsaklis. A brief introduction to the theory and applications of hybrid systems. In *Proc IEEE, Special Issue on Hybrid Systems: Theory and Applications*, 2000.

Eugene Asarin and Oded Maler. As soon as possible: Time optimal control for timed automata. In *Hybrid Systems: Computation and Control*, pages 19–30. Springer, 1999.

Eugene Asarin, Thao Dang, and Oded Maler. The d/dt tool for verification of hybrid systems. In *Computer Aided Verification*, pages 365–370. Springer, 2002.

J. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube. Impulse differential inclusions: A viability approach to hybrid systems. *IEEE Trans. Automat. Contr.*, 47(1):2–20, 2002.

R. Baheti and H. Gill. Cyber-physical systems. *The Impact of Control Technology*, pages 161–166, 2011.

C. Baier and J. P. Katoen. *Principles of model checking*. MIT press Cambridge, 2008.

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.

Gerd Behrmann, Alexandre David, and Kim G Larsen. A tutorial on uppaal. In *Formal methods for the design of real-time systems*, pages 200–236. Springer, 2004.

C. Belta and L. C G J M Habets. Controlling a class of nonlinear systems on rectangles. *Automatic Control, IEEE Transactions on*, 51(11):1749–1759, Nov 2006.

C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G.J. Pappas. Symbolic planning and control of robot motion. *Robotics and Automation Magazine, IEEE*, 14(1):61–70, March 2007.

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

A. Bemporad, F. Borrelli, and M. Morari. On the optimal control law for linear discrete time hybrid systems. In *Hybrid systems: Computation and control*, volume 2289 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2002.

Alberto Bemporad, Giancarlo Ferrari-Trecate, and Manfred Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE transactions on automatic control*, 45(10):1864–1876, 2000.

S. C. Bengea and R. A. DeCarlo. Optimal control of switching systems. *Automatica*, 41(1):11–27, January 2005.

Béatrice Bérard, Michel Bidoit, Alain Finkel, François Laroussinie, Antoine Petit, Laure Petrucci, and Philippe Schnoebelen. *Systems and software verification: model-checking techniques and tools*. Springer Publishing Company, Incorporated, 2010.

A. Bhaya and F. Mota. Equivalence of stability concepts for discrete time-varying systems. *Int. J. Robust and Nonlin. Contr.*, 4:725–740, 1994.

F. Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, 2003.

Francesco Borrelli, Mato Baotić, Alberto Bemporad, and Manfred Morari. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41(10):1709–1721, 2005.

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. SIAM, 1994.

B.A. Brandin and W.M. Wonham. Supervisory control of timed discrete-event systems. *Automatic Control, IEEE Transactions on*, 39(2):329–342, 1994.

M. S. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Trans. Automat. Contr.*, 43(4):475–482, 1998.

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

M.S. Branicky, M.M. Curtiss, J. Levine, and S. Morgan. Sampling-based planning, control and verification of hybrid systems. *IEE Proceedings - Control Theory and Applications*, 153:575–590(15), September 2006.

R. W. Brockett. Asymptotic stability and feedback stabilization. In R. W. Brockett, R. S. Millman, and H. J. Sussmann, editors, *Differential Geometric Control Theory*, pages 181–191. Boston, MA: Birkhuser, 1983.

C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems (2nd Edition)*. Springer-Verlag, 2008.

C.G. Cassandras, D.L. Pepyne, and Y. Wardi. Optimal control of a class of hybrid systems. *IEEE Trans. Automat. Contr.*, 46(3):398–415, 2001.

Christos G Cassandras and John Lygeros. *Stochastic hybrid systems*. CRC Press, 2010.

A. Chutinan and B. H. Krogh. Computational techniques for hybrid system verification. *IEEE Trans. Automat. Contr.*, 48(1):64–75, 2003.

Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *Computer Aided Verification*, pages 359–364. Springer, 2002.

E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT press, 1999.

Edmund M. Clarke, E Allen Emerson, and A Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.

José ER Cury, Bruce H Krogh, and Toshihiko Niinomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *Automatic Control, IEEE Transactions on*, 43(4):564–568, 1998.

J. Daafouz, R. Riedinger, and C. Iung. Stability analysis and control synthesis for switched systems: a switched lyapunov function approach. *IEEE Trans. Automat. Contr.*, 47(11):1883–1887, 2002.

Thao Dang, Alexandre Donzé, Oded Maler, and Noa Shalev. Sensitive state-space exploration. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4049–4054. IEEE, 2008.

Tuhin Das and Ranjan Mukherjee. Optimally switched linear systems. *Automatica*, 44(5):1437–1441, 2008.

H. De Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of computational biology*, 9(1):67–103, 2002.

R. A. DeCarlo, M. S. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. In P. J. Antsaklis, editor, *Proceedings of the IEEE: Special issue on hybrid systems*, volume 88, pages 1069–1082. IEEE Press, 2000.

Akash Deshpande, Aleks GÃűllÃij, and Pravin Varaiya. Shift: A formalism and a programming language for dynamic networks of hybrid automata. In Panos Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*, pages 113–133. Springer Berlin Heidelberg, 1997.

Stephen Edwards, Luciano Lavagno, Edward A Lee, and Alberto Sangiovanni-Vincentelli. Design of embedded systems: Formal models, validation, and synthesis. *Readings in hardware/software co-design*, page 86, 2001.

Magnus Egerstedt. Behavior based robotics using hybrid automata. In Nancy Lynch and BruceH. Krogh, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 103–116. Springer Berlin Heidelberg, 2000.

Magnus Egerstedt, Yorai Wardi, and Henrik Axelsson. Transition-time optimization for switched-mode dynamical systems. *Automatic Control, IEEE Transactions on*, 51(1):110–115, 2006.

E Allen Emerson. Temporal and modal logic. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, 995:1072, 1990.

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.

G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009. ISSN 0005-1098.

E. Feron. Quadratic stabilizability of switched systems via state and output feedback. Technical Report CICS-P-468, MIT, 1996.

Giancarlo Ferrari-Trecate, Marco Muselli, Diego Liberati, and Manfred Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003.

Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In *Computer Aided Verification*, pages 379–395. Springer, 2011.

Paul Gastin and Denis Oddoux. Fast ltl to büchi automata translation. In *Computer Aided Verification*, pages 53–65. Springer, 2001.

A. Girard and G. J. Pappas. Approximation metrics for discrete and continuous systems. *Automatic Control, IEEE Transactions on*, 52(5):782–798, 2007.

A. Girard and G. J. Pappas. Hierarchical control system design using approximate simulation. *Automatica*, 45(2):566–571, 2009. ISSN 0005-1098.

Antoine Girard and George J Pappas. Approximation metrics for discrete and continuous systems. 2005.

R. Goebel, R. Sanfelice, and A. R. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28ÍC93, 2009.

R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid Dynamical Systems: modeling, stability, and robustness*. Princeton University Press, 2012.

Humberto Gonzalez, Ramanarayan Vasudevan, Maryam Kamgarpour, S Shankar Sastry, Ruzena Bajcsy, and Claire Tomlin. A numerical method for the optimal control of switched systems. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 7519–7526. IEEE, 2010.

L. Habets and Jan H. van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40(1):21–35, 2004.

LCGJM Habets, Pieter J Collins, and Jan H van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *Automatic Control, IEEE Transactions on*, 51(6):938–948, 2006a.

LCGJM Habets, M Kloetzer, and Calin Belta. Control of rectangular multi-affine hybrid systems. In *Decision and Control, 2006 45th IEEE Conference on*, pages 2619–2624. IEEE, 2006b.

S. Hedlund and A. Rantzer. Cdp tool: A matlab tool for optimal control of hybrid systems. department of automatic control. In *Lund Institute of Technology*, 1999.

S. Hedlund and A. Rantzer. Convex dynamic programming for hybrid systems. *Automatic Control, IEEE Transactions on*, 47(9):1536–1540, Sep 2002.

W. PMH Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001.

T. Henzinger. Hybrid automata with finite bisimulations. In Z. Füllöp and G. Gécgeg, editors, *ICALP'95: Automata, Languages, and Programming*. Springer-Verlag, 1995.

T. Henzinger. The theory of hybrid automata. In M.K. Inan and R.P. Kurshan, editors, *Verification of Digital and Hybrid Systems*, volume 170 of *NATO ASI Series*, pages 265–292. Springer Berlin Heidelberg, 2000.

T. Henzinger, P. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. In *Computer aided verification*, pages 460–463. Springer, 1997.

T. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

Thomas A Henzinger, Peter W Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 373–382. ACM, 1995.

J. P. Hespanha. Stabilization through hybrid control. In H. Unbehauen, editor, *Encyclopedia of Life Support Systems (EOLSS)*, volume Control Systems, Robotics, and Automation. Oxford, UK, 2004a.

J. P. Hespanha and A. S. Morse. Stability of switched systems with average dwell-time. In *Proc. 38th IEEE Conf. Decision Control*, pages 2655–2660, 1999.

J. P. Hespanha, D. Liberzon, and A. S. Morse. Logic-based switching control of a nonholonomic system with parametric modeling uncertainty. *Systems & Control Letters*, 38(3):167–177, 1999.

J. P. Hespanha, D. Liberzon, D. Angeli, and E. D. Sontag. Nonlinear norm-observability notions and stability of switched systems. *IEEE Trans. Automat. Contr.*, 52(2):154–168, 2005.

Joao P Hespanha. Stochastic hybrid systems: Application to communication networks. In *Hybrid systems: computation and control*, pages 387–401. Springer, 2004b.

Gerard J Holzmann. The model checker spin. *IEEE Transactions on software engineering*, 23(5):279–295, 1997.

J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Prentice Hall, third edition, 2006.

Jianghai Hu, John Lygeros, and Shankar Sastry. Towards a theory of stochastic hybrid systems. In *Hybrid Systems: Computation and Control*, pages 160–173. Springer, 2000.

J. Imura and A. van der Schaft. Characterization of well-posedness of piecewise-linear systems. *IEEE Trans. Automat. Contr.*, 45(9):1600–1619, 2000.

Karl Henrik Johansson, Magnus Egerstedt, John Lygeros, and Shankar Sastry. On the regularization of zeno hybrid automata. *Systems & Control Letters*, 38(3):141–150, 1999.

M. Johansson. *Piecewise Linear Control Systems: A Computational Approach*, volume 284 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, 2003a.

M. K.-J. Johansson, editor. *Piecewise Linear Control Systems: A Computational Approach*, volume 284 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, 2003b.

Aleksandar Lj Juloski, Siep Weiland, and WPMH Heemels. A bayesian approach to identification of hybrid systems. *Automatic Control, IEEE Transactions on*, 50(10):1520–1533, 2005.

Matt Kaufmann, J Strother Moore, and Panagiotis Manolios. *Computer-aided reasoning: an approach*. Kluwer Academic Publishers, 2000.

Hassan K Khalil and JW Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, 2002.

C. King and R. Shorten. A singularity test for the existence of common quadratic lyapunov functions for pairs of stable LTI systems. In *Proc. 2004 American Contr. Conf.*, pages 3881–3884, 2004.

M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *Automatic Control, IEEE Transactions on*, 53(1):287–297, 2008.

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

Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: Probabilistic symbolic model checker. In *Computer performance evaluation: modelling techniques and tools*, pages 200–204. Springer, 2002.

G. Lafferriere, G. J. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematics of control, signals and systems*, 13(1):1–21, 2000.

Carl W Lee. Subdivisions and triangulations of polytopes. In *Handbook of discrete and computational geometry*, pages 271–290. CRC Press, Inc., 1997.

E.A. Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369, May 2008.

Domine MW Leenaerts and Wim MG Van Bokhoven. *Piecewise linear modeling and analysis*. Springer, 1998.

M. D. Lemmon, K. He, and I. Markovsky. Supervisory hybrid systems. *Control Systems, IEEE*, 19(4):42–55, 1999.

D. Liberzon. *Switching in Systems and Control*. Birkhauser, Boston, 2003.

D. Liberzon and A. S. Morse. Basic problems in stability and design of switched systems. *IEEE Contr. Syst. Magazine*, 19(5):59–70, 1999.

D. Liberzon, J. P. Hespanha, and A. S. Morse. Stability of switched linear systems: A lie-algebraic condition. *Syst. Contr. Lett.*, 37(3):117–122, 1999.

H. Lin and P. J. Antsaklis. Switching stabilizability for continuous-time uncertain switched linear systems. *IEEE Trans. Automat. Contr.*, 52(4):633–646, 2007.

H. Lin and P.J. Antsaklis. Stability and stabilizability of switched linear systems: A survey of recent results. *Automatic Control, IEEE Transactions on*, 54(2):308–322, 2009.

Hai Lin and Panos J Antsaklis. Hybrid state feedback stabilization with l 2 performance for discrete-time switched linear systems. *International Journal of Control*, 81(7):1114–1124, 2008.

Ji-Nan Lin and Rolf Unbehauen. Canonical piecewise-linear approximations. *IEEE Transactions on circuits and systems. I: Fundamental theory and applications*, 39(8):697–699, 1992.

Jie Liu, Xiaojun Liu, Tak-Kuen J.Koo, B. Sinopoli, S. Sastry, and E.A. Lee. A hierarchical hybrid system model and its simulation. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 4, pages 3508–3513 vol.4, 1999.

J. Lygeros, S. Sastry, and C. Tomlin. *The Art of Hybrid Systems*. 2001. URL `http://robotics.eecs.berkeley.edu/~sastry/ee291e/book.pdf`.

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

N. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):103–157, 2003.

Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS 95*, pages 229–242. Springer, 1995.

Michael Margaliot and Daniel Liberzon. Lie-algebraic stability conditions for nonlinear switched systems and differential inclusions. *Systems & control letters*, 55(1):8–16, 2006.

A. N. Michel. Recent trends in the stability analysis of hybrid dynamical systems. *IEEE Trans. Circuits Syst. I*, 46(1):120–134, 1999.

R. Milner. *Communication and Concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

Eduardo Mojica-Nava, Nicanor Quijano, and Naly Rakoto-Ravalontsalama. A polynomial approach for optimal control of switched nonlinear systems. *International Journal of Robust and Nonlinear Control*, 2013.

A. P. Molchanov and D. Liu. Robust absolute stability of time-varying nonlinear discrete-time systems. *IEEE Trans. Circuits Syst. I*, 49(8):1129–1137, 2002.

A. P. Molchanov and E. Pyatnitskiy. Criteria of asymptotic stability of differential and difference inclusions encountered in control theory. *Systems & Control Letters*, 13:59–64, 1989.

A. S. Morse. Supervisory control of families of linear set-point controllers - part 1: Exact matching. *IEEE Trans. Automat. Contr.*, 41(10):1413–1431, 1996.

K. S. Narendra and J. Balakrishnan. A common lyapunov function for stable LTI systems with commuting a-matrices. *IEEE Trans. Automat. Contr.*, 39 (12):2469–2471, 1994.

Truong Nghiem, Sriram Sankaranarayanan, Georgios Fainekos, Franjo Ivančić, Aarti Gupta, and George J Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pages 211–220. ACM, 2010.

A. Papachristodoulou and S. Prajna. A tutorial on sum of squares techniques for systems analysis. In *Proc. 2005 American Control Conf.*, 2005.

Antonis Papchristodoulou and Stephen Prajna. Robust stability analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 54(5): 1034–1041, 2009.

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

S. Pettersson. Synthesis of switched linear systems. In *Proc. 42nd IEEE Conf. Decision Control*, pages 5283–5288, 2003.

S. Pettersson and B. Lennartson. Stabilization of hybrid systems using a min-projection strategy. In *Proc. 2001 American Contr. Conf.*, pages 223–228, 2001.

S. Pettersson and B. Lennartson. Hybrid system stability and robustness verification using linear matrix inequalities. *Inter. J. Contr.*, 75(16-17):1335–1355, 2002.

Erion Plaku, Lydia E Kavraki, and Moshe Y Vardi. Falsification of ltl safety properties in hybrid systems. *International Journal on Software Tools for Technology Transfer*, 15(4):305–320, 2013.

Amir Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57, Oct 1977.

Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 179–190. ACM, 1989.

G Pola, ML Bujorianu, J Lygeros, and MDD Benedetto. Stochastic hybrid models: An overview. In *Proc. IFAC Conf. Anal. Design Hybrid Syst*, pages 45–50, 2003.

S. Prajna and A. Papachristodoulou. Analysis of switched and hybrid systems - beyond piecewise quadratic methods. In *Proc. 42nd IEEE Conf. Decision Control*, pages 2779–2784, 2003.

Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.

Michael Melholt Quottrup, Thomas Bak, and RI Zamanabadi. Multi-robot planning: A timed automata approach. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 4417–4422. IEEE, 2004.

Jörg Raisch and Siu D O'Young. Discrete approximation and supervisory control of continuous systems. *Automatic Control, IEEE Transactions on*, 43 (4):569–573, 1998.

P. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

Peter J Ramadge and W Murray Wonham. Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization*, 25(1): 206–230, 1987.

A. Rantzer and M. Johansson. Piecewise linear quadratic optimal control. *IEEE Trans. Automat. Contr.*, 45(4):629–637, 2000.

Mark Reynolds. Continuous temporal models. In *AI 2001: Advances in Artificial Intelligence*, pages 414–425. Springer, 2001.

Jacob Roll, Alberto Bemporad, and Lennart Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, 2004.

A. van der Schaft and H. Schumacher. *An Introduction to Hybrid Dynamical Systems*, volume 251 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, London, 2000.

M Shahid Shaikh and Peter E Caines. On the hybrid optimal control problem: theory and algorithms. *Automatic Control, IEEE Transactions on*, 52(9):1587–1603, 2007.

Robert Shorten, Fabian Wirth, Oliver Mason, Kai Wulff, and Christopher King. Stability criteria for switched and hybrid systems. *SIAM review*, 49(4):545–592, 2007.

B Izaias Silva, Keith Richeson, Bruce Krogh, and Alongkrit Chutinan. Modeling and verifying hybrid dynamic systems using checkmate. In *Proceedings of 4th International Conference on Automation of Mixed Processes*, pages 323–328, 2000.

E. Skafidas, R. J. Evans, A. V. Savkin, and I. R. Petersen. Stability results for switched controller systems. *Automatica*, 35(4):553–564, 1999.

Eduardo Sontag. Nonlinear regulation: The piecewise linear approach. *Automatic Control, IEEE Transactions on*, 26(2):346–358, 1981.

Eduardo D Sontag. *Mathematical control theory: deterministic finite dimensional systems*, volume 6. Springer, 1998.

Zhendong Sun, Shuzhi Sam Ge, and Tong Heng Lee. Controllability and reachability criteria for switched linear systems. *Automatica*, 38(5):775–786, 2002.

H.J. Sussmann. A maximum principle for hybrid optimal control problems. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 1, pages 425–430 vol.1, 1999.

P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, Springer New York, 2009.

Paulo Tabuada. An approximate simulation approach to symbolic control. *Automatic Control, IEEE Transactions on*, 53(6):1406–1418, 2008.

Wolfgang Thomas. On the synthesis of strategies in infinite games. In *STACS 95*, pages 1–13. Springer, 1995.

C. Tomlin, G.J. Pappas, and S. Sastry. Conflict resolution for air traffic management: a study in multiagent hybrid systems. *Automatic Control, IEEE Transactions on*, 43(4):509–521, Apr 1998.

Claire J Tomlin, Ian Mitchell, Alexandre M Bayen, and Meeko Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, 2003.

Fabio Danilo Torrisi and Alberto Bemporad. Hysdel-a tool for generating computational hybrid models for analysis and synthesis problems. *Control Systems Technology, IEEE Transactions on*, 12(2):235–249, 2004.

Moshe Y Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for concurrency*, pages 238–266. Springer, 1996.

Yorai Wardi and Magnus Egerstedt. Algorithm for optimal mode scheduling in switched systems. In *American Control Conference (ACC), 2012*, pages 4546–4551. IEEE, 2012.

M. A. Wicks and R. A. DeCarlo. Solution of coupled lyapunov equations for the stabilization of multi-modal linear systems. In *Proc. 1997 American Contr. Conf.*, pages 1709–1713, 1997.

M. A. Wicks, P. Peleties, and R. A. DeCarlo. Switched controller design for the quadratic stabilization of a pair of unstable linear systems. *European J. Control*, 4:140–147, 1998.

H. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *Automatic Control, IEEE Transactions on*, 11(2):161–167, Apr 1966.

H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, pages 1527–1528. IEEE, 1991.

Guangming Xie and Long Wang. Controllability and stabilizability of switched linear-systems. *Systems & Control Letters*, 48(2):135–155, 2003.

X. Xu and P. J. Antsaklis. Results and perspectives on computational methods for optimal control of switched systems. In *Hybrid Systems: Computation and Control*, pages 540–555. Springer, 2003.

X. Xu and P. J. Antsaklis. Optimal control of switched systems based on parameterization of the switching instants. *IEEE Trans. Automat. Contr.*, 49 (1):2–16, 2004.

H. Ye, A. N. Michel, and L. Hou. Stability theory for hybrid dynamical systems. *IEEE Trans. Automat. Contr.*, 43(4):461–474, 1998.

Sergio Yovine. Kronos: A verification tool for real-time systems. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1):123–133, 1997.

G. Zhai, B. Hu, K. Yasuda, and A. N. Michel. Qualitative analysis of discrete-time switched systems. In *Proc. 2002 American Contr. Conf.*, volume 3, pages 1880–1885, 2002.

G. Zhai, H. Lin, and P. J. Antsaklis. Quadratic stabilizability of switched linear systems with polytopic uncertainties. *Inter. J. Contr.*, 76(7):747–753, 2003.

F. Zhu and P. J. Antsaklis. Optimal control of switched hybrid systems: a brief survey. *ISIS Tech. Report (to appera in* Discrete Event Dynamic Systems*)*, 3, 2011.

Günter M Ziegler. *Lectures on polytopes*, volume 152. Springer, 1995.