Chapter 20
Trajectory-Based Theory for Hybrid Systems

A. Agung Julius

Abstract This chapter presents a trajectory-based perspective in solving safety/reachability analysis and synthesis problems and fault diagnosability analysis in hybrid systems. The main tool used in obtaining the results presented in this chapter is the concept of trajectory robustness, which is derived from the theory of approximate bisimulation. Trajectory robustness essentially provides a guarantee on how far the system's state trajectories can deviate (in L_∞ norm) as a result of initial state variations. It further leads to the possibility of approximating the set of the system's trajectories, which is infinite, with a finite set of trajectories. This fact, in turns, allows us to pose the above problems as finitely many finite problems that can be practically solved. In addition, these finite problems can be solved in parallel.

20.1 Note from the Author

This chapter is dedicated to Arjan van der Schaft in the occasion of his 60th birthday. The work presented here germinated from seed ideas that I developed while working as a doctoral student under Arjan's mentorship. I was a graduate student at the Department of Applied Mathematics at the University of Twente in the period of 1999–2005. During this period, I had the fortune of being introduced to the (then) nascent field of hybrid systems, and exposed to the elegance of the behavioral systems theory. Under Arjan's guidance, I wrote my doctoral dissertation on essentially the intersection of these two fields. In later years, while being a postdoctoral researcher at the University of Pennsylvania, I was introduced to the seminal work of Antoine Girard and George Pappas on the approximate bisimulation theory. The trajectory-based perspective of the behavioral systems theory and the notion of invariance and metrics in the space of trajectories from the approximate bisimulation theory are largely the impetus of the results presented in this paper.

A.A. Julius (✉)
Department of Electrical, Computer, and Systems Engineering,
Rensselaer Polytechnic Institute, 110 Eighth Street, NY, Troy 12180, USA
e-mail: agung@ecse.rpi.edu

M.K. Camlibel et al. (eds.), *Mathematical Control Theory I*,
Lecture Notes in Control and Information Sciences 461,
DOI 10.1007/978-3-319-20988-3_20

20.2 Introduction

Hybrid systems are dynamical systems with interacting discrete and continuous dynamics [1]. Intuitively, one way to describe a hybrid system is to think of it as a multimodal dynamical system, where the dynamics of the continuous states depends on the discrete state of the system, which is also called the *mode* or the *location*. Because of their modeling expressivity, hybrid systems have been used in modeling of embedded systems [2–8], air traffic systems [9–14], automotive systems [15–17], electronic circuits [18–20], genetic regulatory networks [21–23], computational morphodynamics [24], and other fields.

In this chapter, we consider two types of hybrid systems, *autonomous* hybrid systems and *control* hybrid systems. More formal definitions of these systems will follow in Sect. 20.3.1. Intuitively, the autonomous hybrid systems do not admit any input. They are the hybrid systems analog of $\dot{x} = f(x)$. Hybrid control systems, on the other hand, admit both continuous and discrete control inputs. They are the hybrid systems analog of $\dot{x} = f(x, u)$. In a sense, for autonomous hybrid systems, the evolution of the states is completely determined by its initial state.[1]

Research involving autonomous hybrid systems is typically of the analysis type, i.e., they are concerned with proving whether the systems have certain properties. One of the most important analysis problems in hybrid systems is the *reachability/safety analysis*, where the question of interest is whether the system can enter an undesirable state during its execution. Reachability/safety analysis has a lot of important practical applications, for example, in the safety analysis of air traffic systems [11, 12, 14, 25, 26], design verification for electronic circuits [18–20], design verification for synthetic biology [21, 27], and model analysis for biochemical processes [28].

Another type of analysis problems that is also studied a lot is the *observability analysis* (see e.g., the editorial [29]). Here, the question of interest is whether we can infer certain properties of the state trajectories by observing certain aspects thereof. An important problem of this type is *fault diagnosis*. The central question in fault diagnosis is whether we can infer that the state trajectory is faulty (e.g., it involves a directly unobservable fault event) from partial observation (e.g., by observing only the a part of the events in the system). Fault diagnosis for hybrid systems is an active research area, with applications in embedded control systems [30], process control [31], and others.

For hybrid control systems, there is a strong research interest involving synthesis. The *synthesis* part of the safety/reachability issue deals with the construction of control laws/algorithms for systems with input and controllable events, in order to achieve executions with desired properties (e.g., safety) despite uncertainties.

In this chapter, we review some results on reachability/safety analysis and synthesis and fault diagnosis for hybrid systems. The underlying theme of the results is

[1] For simplicity, in this chapter we do not consider nondeterminism and stochasticity in the hybrid system dynamics.

that they are all trajectory based. That is, they make use of trajectories to represent the systems, and they are based on reasoning at the trajectory level, instead of at the system representation level.

20.3 Review of the Fundamentals

20.3.1 Hybrid Automata

Following [1], we define hybrid systems as hybrid automata. A hybrid automaton is expressed as an octuple $\mathcal{H} = (L, \mathcal{X}, Init, A, \mathcal{U}, E, Inv, \Sigma)$, where:

- L is a finite set of discrete states, which are also called modes or locations.
- $\mathcal{X}$ is the continuos state space.
- $Init \subset \mathcal{X} \times L$ is the set of initial states.
- A is a finite set of transition symbols.
- $\mathcal{U}$ is the space of continuous input.
- E is the set of transitions.
- $Inv : L \to 2^{\mathcal{X}}$ defines the invariant sets of each location. For an $\ell \in L$, $Inv(\ell)$ is the set in which the continuous states must remain as long as the discrete state is ℓ.
- Σ assigns each location to its continuous dynamics. For each location $\ell \in L$, we define

$$\Sigma(\ell) : \dot{x} = F_\ell(x),\ x \in Inv(\ell), \tag{20.1}$$

  if the hybrid system is autonomous, or

$$\Sigma(\ell) : \dot{x} = F_\ell(x, u),\ x \in Inv(\ell),\ u \in \mathcal{U}, \tag{20.2}$$

  if the hybrid system admits control inputs. Here we assume that for each location F_ℓ satisfies some conditions that guarantee well-posedness of the continuous dynamics.

Each transition in E is a pentuple $e = (\ell, \ell', Guard, R, a) \in E$, where $\ell \in L$ is the origin of the transition, $\ell' \in L$ is the target location, $Guard \subset Inv(\ell)$ is the guard set of the transition, and $R : Inv(\ell) \to Inv(\ell')$ is the reset map. The symbol $a \in A$ is the symbol associated with the transition. The semantics of the execution of a hybrid automaton can be explained as follows: (see illustration in Fig. 20.1a). An execution trajectory of $\mathcal{H}$ is a sequence

$$(\ell_0, x_0, u_0, e_0, \Delta_0), (\ell_1, x_1, u_1, e_1, \Delta_1), \ldots, (\ell_N, x_N, u_N, \varnothing, \Delta_N), \tag{20.3}$$

where for all values of i that appear here $\Delta_i \in [0, \infty)$, $e_i \in E$, and $u_i : [0, \Delta_i] \to \mathcal{U}$ is the input signal, if the hybrid system admits input. If the system does not admit

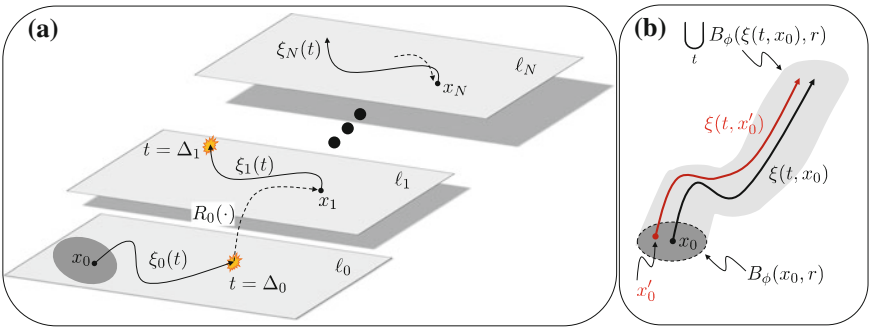

Fig. 20.1 An illustration of **(a)** the execution trajectory of a hybrid automaton, **b** the concept of trajectory robustness

any input, the execution trajectory is a sequence

$$(\ell_0, x_0, e_0, \Delta_0), (\ell_1, x_1, e_1, \Delta_1), \ldots, (\ell_N, x_N, \varnothing, \Delta_N). \tag{20.4}$$

The initial state $(x_0, \ell_0) \in Init$. Each element of the sequence is essentially an interval of execution within which the discrete state is constant. These execution intervals can be characterized recursively as follows. For the ith interval, the value of the continuous state $x(t)$ is given by $\xi_i(t)$, which satisfies $\xi_i(0) = x_i$ and the ODE given by $\Sigma(\ell_i)$. Within the time interval $[0, \Delta_i]$, $\xi_i(t) \in Inv(\ell_i)$. At time $t = \Delta_i$, the transition $e_i = (\ell_i, \ell_{i+1}, Guard_i, R_i, a)$ occurs. That means $\xi_i(\Delta_i) \in Guard_i$ and the continuous state is reset for the next interval of execution. That is, for the $(i+1)$-st interval, the continuous state is initialized at $\xi_{i+1}(0) = x_{i+1} = R_i(\xi_i(\Delta_i))$. Further, the symbol $a \in A$ is associated to the transition. If the transition is triggered externally, for example, a can be considered the discrete command that is given to the system. For the discussion in this chapter, we limit our attention to execution trajectories with finitely many intervals, and that the last interval does not terminate with a transition. Physically, the amount of time that elapses during the execution trajectory above is $\sum_{i=0}^{N} \Delta_i$. Also, we only stipulate that the transitions occur when the continuous state is in the guard set of the transition. We do not stipulate (yet) whether the transitions happen spontaneously, i.e., triggered by the system's own dynamics (example: a falling object bouncing off the floor), or they are triggered externally (example: switching gear in manual transmission).

20.3.2 Trajectory Robustness

The key ingredient in our framework is the notion of *trajectory robustness*. With the notion of trajectory robustness, we provide a guarantee on how far the system's state trajectories can deviate (in L_∞ norm) as a result of initial state variations. This concept

is easily extensible to treat system parameter variation, for example, by embedding the parameters as static states in the system. Therefore, although not explicitly stated, the following discussion also applies to variations in system parameter.

We construct trajectory robustness using the theory of *approximate bisimulation*, which was developed by Girard and Pappas [32–34]. This theory was subsequently extended to stochastic hybrid systems and trajectory-based analysis of hybrid systems [35–39]. In the following, we review the application of approximate bisimulation in establishing state trajectory robustness with respect to initial condition variation for a nonlinear dynamical system

$$\Sigma : \dot{x} = F(x),\ x \in \mathcal{X}, \tag{20.5}$$

where x is the state of the system, and $\mathcal{X}$ is the state space. Suppose that we can construct a differentiable function $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that

$$\phi(x, x') \geq \left\| x - x' \right\|, \forall x, x' \in \mathcal{X}, \tag{20.6a}$$

$$\frac{d\phi}{dt} = \frac{\partial \phi}{\partial x} F(x) + \frac{\partial \phi}{\partial x'} F(x') \leq 0, \forall x, x' \in \mathcal{X}. \tag{20.6b}$$

Such a function $\phi(x, x')$ is called a *autobisimulation function* [32–34].

Notation We denote the solution of (20.5) with initial state x_0 as $\xi(t, x_0)$ and we define the ball

$$B_\phi(x, r) \triangleq \{y \in \mathcal{X} \mid \phi(x, y) \leq r\},\ x \in \mathcal{X},\ r > 0. \tag{20.7}$$

From (20.6b), we can easily conclude that the value of ϕ is nondecreasing along any two state trajectories of the system. From (20.6a), we can see that $\phi(x, x')$ is an upper bound for the Euclidean distance between the two states. Then by combining these two properties, we can conclude that for all $t \geq 0$,

$$\xi(t, x_0') \in B_\phi(\xi(t, x_0), \phi(x_0, x_0')), \tag{20.8}$$

$$\left\| \xi(t, x_0) - \xi(t, x_0') \right\| \leq \phi(x_0, x_0'), \tag{20.9}$$

for any initial state $x_0' \in \mathcal{X}$. Please refer to Fig. 20.1b for an illustration of this concept.

Remark 20.1 The concept of trajectory robustness is very related to the concept of contraction metric developed by Lohmiller and Slotine [40]. In general, there are some differences between the two concepts. For example, autobisimulation function can also be defined as a pseudometric if we are only concerned about the divergence of the state trajectories in a certain subspace. Also, as the name suggests, bisimulation functions are originally defined to bound the divergence between the state trajectories of two different systems [33]. However, as defined in this chapter, if we also require that ϕ is a metric in $\mathcal{X}$, then it can be considered as a contraction metric.

The autobisimulation function ϕ plays an essential role in establishing trajectory robustness. If (20.5) defines a stable linear affine dynamics

$$\Sigma : \dot{x} = Ax + b,\ x, b \in \mathbb{R}^n,\ A \in \mathbb{R}^{n \times n}, \tag{20.10}$$

and A is Hurwitz, then ϕ can be using a quadratic Lyapunov function as follows [33, 38].

$$\phi(x, x') = \sqrt{(x - x')^T P (x - x')}, \tag{20.11}$$

where P is a symmetric positive definite matrix satisfying the Lyapunov Linear Matrix Inequality

$$A^T P + PA \preceq 0. \tag{20.12}$$

If (20.5) defines a special class of nonlinear dynamics, the procedure above can be extended. For example, if $F(x)$ in (20.5) is polynomial, we can search for a polynomial autobisimulation function. We refer the reader for more details on this to [39], where sum-of-squares optimization technique [41] is used for this purpose. For the discussion in this chapter, it suffices to consider the linear affine case above.

20.4 Approximation with Finite Behavior

Trajectory robustness gained from approximate bisimulation theory is a very useful tool. It allows us to approximate a dynamical system or a hybrid system with a representation that has finitely many trajectories. The main idea can be explained as follows. Consider the dynamics given by (20.5), and suppose that the system is known to have an initial state in a compact set $Init \subset \mathcal{X}$. This means, any trajectory of the system can be written as $\xi(t, x_0')$, for some $x_0' \in Init$. Suppose that we have a bisimulation function ϕ that satisfies (20.6a), (20.6b). See the illustration in Fig. 20.2.

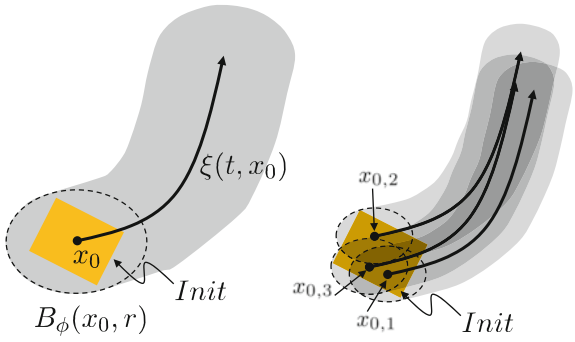

Fig. 20.2 Approximation of the sytem's trajectories with a set of one trajectory (*left*) or finitely many trajectories (*right*)

If $x_0 \in \mathcal{X}$ and $r > 0$ are such that

$$Init \subset B_\phi(x_0, r), \tag{20.13}$$

then for any trajectory of the system $\xi(t, x_0')$, we have

$$\xi(t, x_0') \in B_\phi(\xi(t, x_0), r). \tag{20.14}$$

Thus, we can think of a single trajectory $\xi(t, x_0)$ as an approximation of the entire set of the system's trajectories. Equation (20.14) essentially means that the accuracy of this approximation is given by r.

The idea above can be further generalized and stated as follows.

Theorem 20.2 *Consider a family of initial states $x_{0,1}, x_{0,2}, \ldots, x_{0,M} \in \mathcal{X}$ and positive numbers $r_1, r_2, \ldots, r_M$ such that*

$$Init \subset \bigcup_{k=1}^{M} B_\phi(x_{0,k}, r_k). \tag{20.15}$$

For any $x_0' \in Init$ there exists $k \in \{1, \ldots, M\}$ such that

$$\xi(t, x_0') \in B_\phi(\xi(t, x_{0,k}), r_k), \forall t \geq 0. \tag{20.16}$$

The main point of this theorem is that the entire set of the system's trajectories can be approximated by a finite set of trajectories. Again, the numbers $r_1, r_2, \ldots, r_M$ essentially define the accuracy of this approximation. The smaller they are, the approximation is more accurate but we can expect to need more balls to cover $Init$.

In the remainder of this section, we will discuss the extension of this idea to hybrid systems. We limit our discussion in this section to autonomous hybrid systems and leave control hybrid systems for Sect. 20.7.3. Consider a hybrid automaton $\mathcal{H}$ as defined in Sect. 20.3.1. Suppose that $L = \{\ell_0, \ell_1, \ldots, \ell_{|L|}\}$ and that for each discrete state $\ell_i \in L$ the continuous state dynamics

$$\Sigma(\ell_i) : \dot{x} = F_{\ell_i}(x)$$

admits an autobisimulation function ϕ_i. Further, we assume that the guard sets of the transitions define the boundary of the invariant sets of the locations. Also, we assume the transitions in this system occur as soon as the continuous state hits a guard of a transition. Consider an execution trajectory of such hybrid automaton, as exemplified in (20.4). For simplicity of the discussion, let us assume that there are only two intervals, i.e., the trajectory is $(\ell_0, x_0, e_0, \Delta_0), (\ell_1, x_1, \varnothing, \Delta_1)$. This is illustrated in Fig. 20.3. The transition $e_0 = (\ell_0, \ell_1, Guard_0, R_0, a_0)$. We define $\overline{Guard}$ as the union of the guards of all transitions other than e_0 that start in ℓ_0,

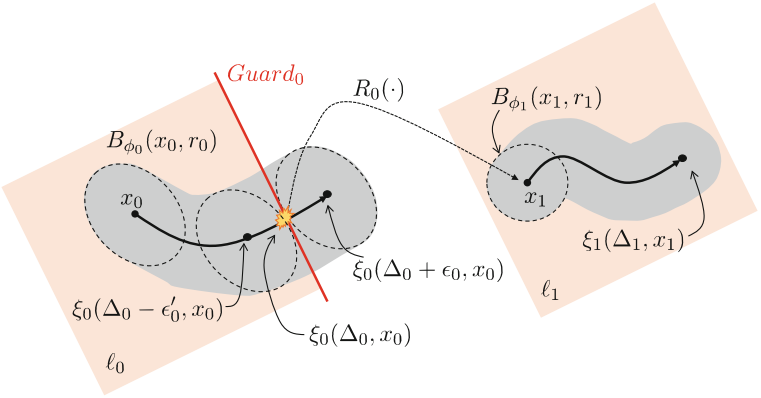

Fig. 20.3 An illustration of an execution trajectory of a hybrid automaton and the concept of trajectory robustness

$$R_0^{-1}\left(B_{\phi_1}(x_1, r_1)\right) \triangleq \left\{x \in Guard_0 \mid R_0(x) \in B_{\phi_1}(x_1, r_1)\right\},$$
$$G_0 \triangleq Guard_0 \backslash R_0^{-1}\left(B_{\phi_1}(x_1, r_1)\right).$$

We can formulate the following theorem (adapted from [38]).

Theorem 20.3 *Suppose that $r_0, r_1, \varepsilon_0, \varepsilon_0' > 0$ are such that the following are true.*

$$B_{\phi_0}(x_0, r_0) \subset Init, \tag{20.17a}$$
$$B_{\phi_0}(\xi_0(t, x_0), r_0) \not\cap (\overline{Guard} \cup G_0), \ \forall t \in [0, \Delta_0 + \varepsilon_0], \tag{20.17b}$$
$$B_{\phi_0}(\xi_0(t, x_0), r_0) \not\cap Inv(\ell_0), \tag{20.17c}$$
$$B_{\phi_0}(\xi_0(t, x_0), r_0) \not\cap Guard_0, \ \forall t \in [0, \Delta_0 - \varepsilon_0'] \tag{20.17d}$$

Then, for any $x_0' \in B_{\phi_0}(x_0, r_0)$ the following are also true:

- *The execution trajectory starting from (x_0', ℓ_0) also exits ℓ_0 through transition e_0.*
- *The transition occurs at time Δ_0', where $\Delta_0' \in [\Delta_0 - \varepsilon_0', \Delta_0 + \varepsilon_0]$.*
- *In the first interval, for all $t \in [0, \Delta_0']$, $\xi_0(t, x_0') \in B_{\phi_0}(\xi_0(t, x_0), r_0)$.*
- *After the transition e_0, the continuous state is reset into $B_{\phi_1}(x_1, r_1)$.*

This theorem can be generalized to trajectories with more intervals. Essentially, it shows that the trajectory starting at (ℓ_0, x_0) is an approximation of those starting in the location ℓ_0 with initial continuous state in the neighborhood of x_0 in the sense that (i) the divergence of the continuous state trajectory is bounded in the sense of Theorem 20.2, (ii) the sequence of transitions are preserved, and (iii) the divergence of the transition times is bounded. Following the same idea as in Theorem 20.2, if the set of initial states is compact, we can use Theorem 20.3 to approximate the set of trajectories of a hybrid automaton with a finite set of trajectories.

20.5 Safety/Reachability Analysis

Safety/reachability analysis is concerned with the question whether any of the system's state trajectories enters a predefined set of unsafe states. For a dynamical system as in (20.5) with a set of initial states $Init$, we define a set $Unsafe \subset \mathcal{X}$ and ask whether there is an initial state $x_0' \in Init$ such that $\xi(t', x_0) \in Unsafe$ for some $t' \in [0, T]$. If such initial state does not exist then the system is safe.[2] This is illustrated in Fig. 20.4. The question described above is called bounded-time safety/reachability analysis, because of the specified upper bound T. If the dynamics (20.5) admits an autobisimulation function ϕ, then the following result can be stated.

Proposition 20.4 *See the illustration in Fig. 20.4. If $r > 0$ is such that*

$$B_\phi(\xi(t, x_0), r) \not\cap Unsafe, \ \forall t \in [0, T],$$

then there exists no initial state $x_0' \in B_\phi(x_0, r)$ from which the state trajectory enters $Unsafe$ in the time interval $[0, T]$.

This proposition allows us to generalize the safety property of the trajectory initialized at x_0 to other trajectories initialized at other states in its neighborhood. Further, the safety of the entire system can be proved by analyzing the safety of finitely many trajectories, as stated below.

Theorem 20.5 *Consider a family of initial states $x_{0,1}, x_{0,2}, \ldots, x_{0,M} \in \mathcal{X}$ and positive numbers $r_1, r_2, \ldots, r_M$ such that*

$$Init \subset \bigcup_{k=1}^{M} B_\phi(x_{0,k}, r_k). \tag{20.18}$$

If for each $k \in \{1, \ldots, M\}$

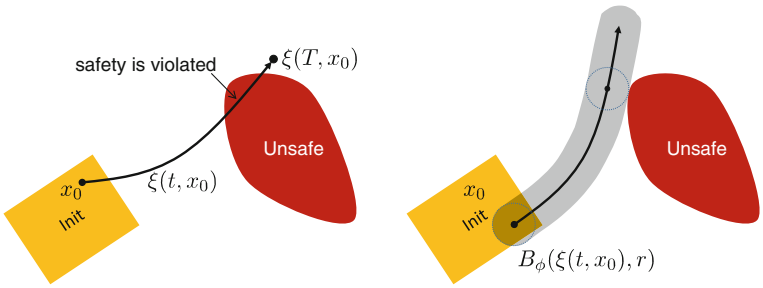

Fig. 20.4 *Left* An illustration of the safety property. *Right* How trajectory robustness can be used in safety/reachability analysis

[2] We assume that $Init$ and $Unsafe$ do not intersect. Otherwise, the problem is trivial.

$$B_\phi(\xi(t, x_{0,k}), r_k) \not\cap Unsafe, \ \forall t \in [0, T],$$

then the system is safe.

For an autonomous hybrid automaton $\mathcal{H}$, safety/reachability analyis can be setup by defining a set of unsafe states $Unsafe \subset \mathcal{X} \times L$. Again, the system is deemed safe if for any state trajectory initialized in $Init \subset \mathcal{X} \times L$, the resulting state trajectory does not enter $Unsafe$. Next, we formulate the analog of Proposition 20.4 for autonomous hybrid automaton. Consider the hybrid automaton discussed in Sect. 20.4, and its execution trajectory that is discussed in Theorem 20.3. The trajectory is $(\ell_0, x_0, e_0, \Delta_0), (\ell_1, x_1, \varnothing, \Delta_1)$.

Proposition 20.6 *Suppose that $r_0, r_1, \varepsilon_0, \varepsilon_0' > 0$ satisfy the conditions (20.17a), (20.17d) in Theorem 20.3. In addition, suppose that*

$$B_{\phi_0}(\xi_0(t, x_0), r_0) \not\cap Unsafe, \ \forall t \in [0, \Delta_0 + \varepsilon_0],$$
$$B_{\phi_1}(\xi_1(t, x_1), r_1) \not\cap Unsafe, \ \forall t \in [0, \Delta_1].$$

Then, for any $x_0' \in B_{\phi_0}(x_0, r_0)$ the following are also true:

- *The execution trajectory starting from (x_0', ℓ_0) is safe until transition e_0 that happens at time Δ_0', i.e.,*

$$\xi_0(t, x_0') \notin Unsafe, \ \forall t \in [0, \Delta_0'].$$

- *Afterwards, in location ℓ_1, the execution trajectory is safe for Δ_1 time units, i.e.,*

$$\xi_1(t, R_0\left(\xi_0(\Delta_0', x_0')\right)) \notin Unsafe, \ \forall t \in [0, \Delta_1].$$

This proposition can be generalized to the case where the execution trajectory has more intervals. Also, if the set of initial states $Init$ can be covered by the union of balls as in Theorem 20.5, then we can prove that the system is safe. More details on this idea is reported in [38, 42, 43].

20.6 Observability and Fault Diagnosability

Observability analysis can be intuitively explained as follows. Suppose that the set $\mathfrak{B}$ contains all trajectories of the system, and $\pi_1 : \mathfrak{B} \to \mathfrak{P}_1$ and $\pi_2 : \mathfrak{B} \to \mathfrak{P}_2$ are surjective maps with co-domains $\mathfrak{P}_1$ and $\mathfrak{P}_2$ respectively. The map π_1 can be considered as observation map that extracts information from the trajectories in $\mathfrak{B}$. If this map is bijective, then all information from the trajectories in $\mathfrak{B}$ is retained. Otherwise, multiple distinct trajectories in $\mathfrak{B}$ are mapped to the same element in $\mathfrak{P}_1$, representing the idea that some information (that distinguishes these trajectories) is lost in the observation. The map π_2 represents another aspect of the trajectories in $\mathfrak{B}$. We say that $\mathfrak{P}_2$ is *observable* from $\mathfrak{P}_1$ if the composite map $\pi_1^{-1} \circ \pi_2$ is

injective, where π_1^{-1} is the set-theoretic inverse map of π_1. This means the observed information from π_1 can uniquely determine the output of π_2.

In classical linear systems theory, this (behavioral) definition of observability coincides with the notion of observability for state-space systems [44]. That is, if we define the state-space system as

$$\Sigma_{\text{lin}} : \begin{cases} \dot{x} = Ax + Bu, \ x \in \mathbb{R}^n, u \in \mathbb{R}^m, \\ y = Cx + Du, \ y \in \mathbb{R}^p, \end{cases} \tag{20.19}$$

we define $\mathfrak{B}$ to be the set of (x, u, y) trajectories that are compatible with this system description. The map π_1 takes such trajectories and retains only the x components. The map π_2 is the identity map. The characterization of observability as discussed in the previous paragraph coincides with the well-known Kalman rank condition

$$\text{rank}[C^T \ A^T C^T \ \ldots \left(A^T\right)^n C^T] = n. \tag{20.20}$$

For hybrid automata as in Sect. 20.3.1, the notion of observation can be more general. In particular, in this chapter, we consider the observation that simply retains only the discrete aspect of the trajectories. This can be made precise using the following definition.

Definition 20.7 For a hybrid automaton $\mathcal{H}$ as defined in Sect. 20.3.1, we define the function $\Gamma : E \to A$ to map any transition $e \in E$ to its transition symbol. For an execution trajectory

$$\omega \triangleq (\ell_0, x_0, e_0, \Delta_0), (\ell_1, x_1, e_1, \Delta_1), \ldots, (\ell_N, x_N, \varnothing, \Delta_N), \tag{20.21}$$

we define the map

$$\pi_{\text{discrete}}(\omega) \triangleq (\Gamma(e_0), \Delta_0), (\Gamma(e_1), \Delta_1), \ldots, (\varnothing, \Delta_N). \tag{20.22}$$

The map π_{discrete} essentially takes the execution trajectory and returns only the symbols of the transitions and the intervals between the occurrence of the symbols. Note that Γ is not necessarily injective, which implies that the transitions are not necessarily distinguishable one from another. As an extreme case, A is a singleton. That means we can only observe when a transition has occurred, but not which transition. Observability analysis for this kind of observation map has been considered, for example by Di Benedetto et al. in [45] where the question is whether the discrete state can be uniquely determined there from.

Fault diagnosability analysis is related to observability analysis. Suppose that $\mathfrak{B}$, the set of trajectories of the system, can be divided into two disjoint partitions, $\mathfrak{B}_{\text{nom}}$ and $\mathfrak{B}_{\text{fault}}$. $\mathfrak{B}_{\text{nom}}$ represents the normal behavior of the system, while $\mathfrak{B}_{\text{fault}}$ represents the faulty behavior of the system. That is, $\mathfrak{B}_{\text{fault}}$ consists of trajectories where a fault occurs. If we again define the observation map $\pi_1 : \mathfrak{B} \to \mathfrak{P}_1$ as above, then the system is *fault diagnosable* from the observation map π_1 if for any $p \in \mathfrak{P}_1$,

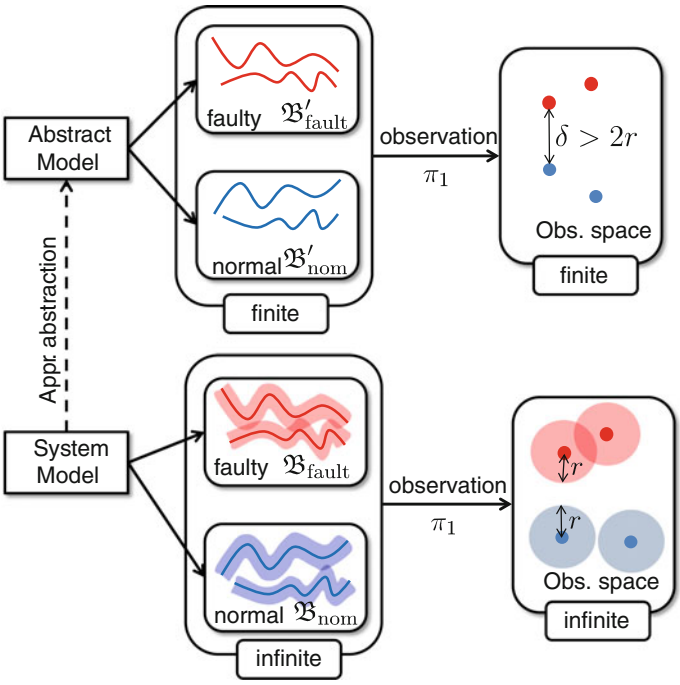

Fig. 20.5 By approximating all the system's trajectories with a set of finitely many trajectories, we can reduce fault diagnosability analysis to a finite problem

$\pi_1^{-1}(p)$ is either strictly in $\mathfrak{B}_{\text{nom}}$ or strictly in $\mathfrak{B}_{\text{fault}}$. In other words, based on the observation defined by π_1, we can always conclude whether the trajectory is normal or faulty.

Although the basic concept is easy to understand, in practice verifying fault diagnosabilty is difficult because the system typically has infinitely many trajectories. However, if we can approximate the system with another one with finitely many trajectories, as explained in Sect. 20.4, then the analysis is much simpler. This is illustrated in Fig. 20.5.

Suppose that $\mathfrak{B}$ can be approximated with $\mathfrak{B}'$ that only has finitely many trajectories. That is, there exists an injective map $\alpha : \mathfrak{B} \to \mathfrak{B}'$ such that for any trajectory $\omega \in \mathfrak{B}$, $\alpha(\omega) \in \mathfrak{B}'$ is an approximation of ω. Intuitively, ω and $\alpha(\omega)$ are close to each other. To be precise, suppose that $\mathfrak{P}_1$ is equipped with a metric $\|\cdot\|_p$ and

$$\|\pi_1(\omega) - \pi_1(\alpha(\omega))\|_p \leq r, \ \forall \omega \in \mathfrak{B}. \tag{20.23}$$

Then, if we define

$$\mathfrak{B}'_{\text{nom}} \triangleq \alpha(\mathfrak{B}_{\text{nom}}), \ \mathfrak{B}'_{\text{fault}} \triangleq \alpha(\mathfrak{B}_{\text{fault}}),$$

we have the following result.

Theorem 20.8 *If for any* $\omega_1 \in \mathfrak{B}'_{\text{nom}}$ *and* $\omega_2 \in \mathfrak{B}'_{\text{fault}}$

$$\|\pi_1(\omega_1) - \pi_1(\omega_2)\| > 2r \tag{20.24}$$

then the system is fault diagnosable.

Note that checking the condition (20.24) in this theorem is practically possible because both $\mathfrak{B}'_{\text{nom}}$ and $\mathfrak{B}'_{\text{fault}}$ have finitely many trajectories.

For an autonomous hybrid automaton, suppose that ω is an execution trajectory given in (20.21) and $\alpha(\omega)$ is the approximate trajectory in the sense of Theorem 20.3. From the theorem, we know that both trajectories have the same sequence of transitions, and the timing of transitions are close. That is, if the observation map is π_{discrete} and $\pi_{\text{discrete}}(\omega)$ is as given in (20.22), then

$$\pi_{\text{discrete}}(\alpha(\omega)) = \left(\Gamma(e_0), \Delta'_0\right), (\Gamma(e_1), \Delta'_1), \ldots, (\varnothing, \Delta_N), \tag{20.25}$$

$$\Delta'_k \in [\Delta_k - \varepsilon, \Delta_k + \varepsilon], \forall k \in \{0, 1, \ldots, N-1\}, \tag{20.26}$$

for some $\varepsilon > 0$. The distance between $\pi_{\text{discrete}}(\omega)$ and $\pi_{\text{discrete}}(\alpha(\omega))$ can be defined in terms of the timing difference, and hence the distance can be bounded as in (20.23). Therefore, we can use Theorem 20.8 to verify fault diagnosability for autonomous hybrid automata based on observing the timing of the transitions and the respective transition symbols. This is the underlying idea behind some recent work on fault diagnosability of some classes of hybrid systems [46] and probabilistic hybrid systems [47].

20.7 Controller Synthesis

In this section, we discuss the controller synthesis problem related to safety/ reachability properties. For a dynamical system given by

$$\Sigma : \dot{x} = F(x, u), \ x \in \mathcal{X}, \ u \in \mathcal{U}, \tag{20.27}$$

where F is well-posed, the problem can be posed as follows. Given a compact set of initial condition $Init \subset \mathcal{X}$, and a set of goal states $Goal \subset \mathcal{X}$, we want to steer the state starting from any initial state $x_0 \in Init$ such that the state trajectories enter $Goal$ at time $t = T$ and in the time interval $[0, T]$ the state remains safe (does not enter a set of states termed $Unsafe$).

The notion of trajectory robustness discussed in Sect. 20.3.2 can also be used in trajectory-based controller synthesis. The key concept in this approach is the *control autobisimulation function (CAF)* [48, 49]. A continuously differentiable function $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **control autobisimulation function** of (20.5) if for any

$x, x' \in \mathcal{X}$, $\psi(x, x') \geq \|x - x'\|$, and there exists a function $k : \mathcal{X} \to \mathcal{U}$ such that

$$\frac{d\psi}{dt} = \frac{\partial \psi}{\partial x} f(x, k(x)) + \frac{\partial \psi}{\partial x'} f(x', k(x')) \leq 0. \tag{20.28}$$

Remark 20.9 The control autobisimulation function is an analog of the control Lyapunov function (CLF) [50], for trajectory robustness [33, 38]. While control Lyapunov function has been used to construct control laws that guarantee stability (e.g., [51]), we shall use the control autobisimulation function to construct control laws that guarantee trajectory robustness.

A consequence of the existence of a CAF is the existence of a feedback control law $u = k(x)$, such that the closed-loop system

$$\dot{x} = F(x, k(x)), x \in \mathcal{X}, \tag{20.29}$$

has $\psi(\cdot, \cdot)$ as a autobisimulation function (see Sect. 20.3.2). For a given dynamical system Σ in (20.27) and a control autobisimulation function ψ, the class of all feedback control laws $k(\cdot)$ that satisfy (20.28) is called the *class of admissible feedback laws*.

Notation For a given dynamical system Σ in (20.27) and a feedback control law $u = k(x)$, the closed-loop trajectory with initial condition $x(0) = x_0$ is denoted by $\xi_k(t, x_0)$. For a control autobisimulation function ψ, $x \in \mathcal{X}$, $r > 0$, we define the ball

$$B_\psi(x, r) \triangleq \{y \in \mathcal{X} \mid \psi(x, y) \leq r\}.$$

The trajectory-based controller synthesis paradigm can be stated as follows. We first construct feedback controllers from the class of feasible feedback laws. By definition, the closed-loop system will then admit a predefined autobisimulation function. This means that the trajectory robustness property discussed in Sect. 20.3.2 is guaranteed to hold. Please refer to Fig. 20.6 for an illustration.

Theorem 20.10 *Suppose that for a given initial state $x_0 \in Init$, we can design an admissible feedback law $u = k_0(x)$ that results in a closed-loop execution trajectory $\xi_{k_0}(t, x_0)$ satisfying*

$$B_\psi(\xi_{k_0}(t, x_0), r_0) \cap Unsafe, \ \forall t \in [0, T], \tag{20.30}$$

$$B_\psi(\xi_{k_0}(t, x_0), r_0) \subset Goal. \tag{20.31}$$

Then, for any initial state $x_0' \in B_\psi(\xi_{k_0}(t, x_0), r_0)$, the closed-loop trajectory $\xi_{k_0}(t, x_0)$ is also safe for $t \in [0, T]$ and is in the Goal set at $t = T$.

Therefore, the admissible feedback law $u = k_0(x)$ is applicable not only for the initial state x_0 but also to other initial states in its neighborhood. The controller synthesis procedure can be performed in two steps:

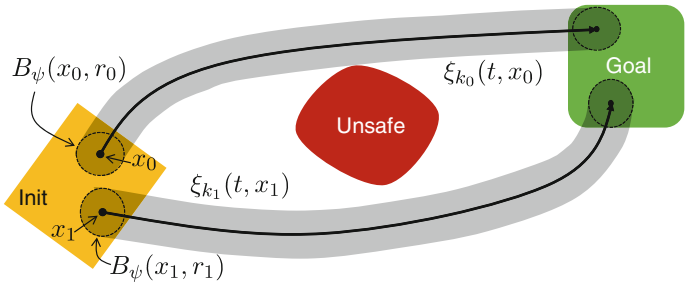

Fig. 20.6 An illustration for trajectory-based controller synthesis

Step 1 For a given initial state, synthesize an innerloop controller that endows the system with the trajectory robustness property.

Step 2 Obtain finitely many trajectories resulting from Step 1 that have the desired qualitative properties to cover $Init$. Note that the controller in Step 1 can depend on the initial state.

Effectively, the trajectory robustness approach allows us to reduce the problem of finding a control law that works for infinitely many initial states in $Init$ to a problem where this has to be done for finitely many initial states. Moreover, the control law can depend on the initial state, and the control law for each initial state can be designed independently of the others'. Steering the system from a particular initial state, is arguably an easier task than finding a control law that works for the entire $Init$ set. Thus, we break down a hard problem into a finite number of simpler and parallelizable problems.

20.7.1 Controller Synthesis for Linear Affine Dynamics

The synthesis of the CAF and the controllers for systems with linear affine dynamics is discussed below. In this case, $F(x, u)$ in (20.27) takes the form

$$F(x, u) = Ax + f + Bu, \ x \in \mathbb{R}^n, u \in \mathbb{R}^m, \tag{20.32}$$

where $A \in \mathbb{R}^{n\times n}$, $f \in \mathbb{R}^n$, and $B \in \mathbb{R}^{n\times m}$. For such systems, we again construct CAF as quadratic functions [48, 49]. That is, we assume that

$$\psi(x, x') = \sqrt{(x - x')^T P (x - x')}, \tag{20.33}$$

where $P \in \mathbb{R}^{n\times n}$ is a positive definite matrix. In this case, the inequality (20.28) becomes

$$(x - x')^T P \left(A(x - x') + B(k(x) - k(x'))\right) \leq 0. \tag{20.34}$$

We then construct a feedback law of the form $u(t) = k(x) = Kx + v(t)$, where $K \in \mathbb{R}^{m \times n}$, and $v(t) \in \mathbb{R}^m$ is a time-varying function, both to be determined later. With this controller, (20.34) becomes

$$(x - x')^T P (A + BK)(x - x') \leq 0. \tag{20.35}$$

A well-known result in control theory (see e.g., [52, 53]) states that there exist P and K such that (20.35) holds if and only if (A, B) is stabilizable. In this case, there are well-known methods to synthesize the suitable P and K. For example, we can pose (20.24) as a linear matrix inequality (LMI) [54], which can be solved efficiently using existing semidefinite programming software tools, such as SeDuMi or SDPT3. With some modification, this method can also be used to handle magnitude constraint on the input signal, $\|u\|_{L_\infty} \leq M$, for some $M > 0$ [48, 49].

Notice that given P and K that satisfy (20.35), we are still free to design $v(t)$. Whatever $v(t)$ is, the control law is admissible. The remaining task in the controller design is therefore to find $v(t)$ that steers the trajectories of the closed-loop system from $Init$ to the $Goal$ set, without entering the $Unsafe$ set. This corresponds to Step 2 in the previous section. The problem of finding such $v(t)$ for a given initial state is easier to solve than the original problem, because the control input is only required to work for that particular initial state. We can use a variety of methods for this, for example, using path planning methods from robotics [48], or by using human inputs [49].

In this chapter, we use $v(t)$ as a feedforward control input that depends on the initial state $x(0)$. It is actually possible to define $v(t)$ through a feedback control law, i.e., as a function of $x(t)$. Such feedback law can be learned from the feedforward control input, and is guaranteed to have the same safety property as the feedforward controller above. For further discussion on this topic, the reader is referred to [55].

20.7.2 Controller Synthesis for Nonlinear Dynamics

The results from the previous section can be generalized to some classes of systems with nonlinear dynamics [56]. We consider systems of the form:

$$\Sigma : \begin{cases} \frac{dx}{dt} = f(x) + g(x)u, \ x \in \mathbb{R}^n, u \in \mathbb{R}^m, \\ y = h(x), \ y \in \mathbb{R}^m, \end{cases} \tag{20.36}$$

where y is the output of the system. We assume that the safety and goal reaching properties of the system can be expressed in terms of y (instead of x).

20.7.2.1 Feedback Linearizable Systems

If (20.36) is feedback linearizable (for a comprehensive discussion, the reader is referred to standard textbooks on Nonlinear Control Systems such as [57, 58]), there exists a feedback law

$$u(t) = \kappa(x) + \lambda(x)w(t), \ w(\cdot) \in \mathbb{R}^m, \tag{20.37}$$

such that the closed-loop system, with new input $w(t)$ and output $y(t)$, is a linear system. The necessary and sufficient conditions for feedback linearizability and the design procedure for $\kappa(x)$ and $\lambda(x)$ are covered in the above-mentioned books. In the context of our discussion, the linearizing feedback can be implemented as an inner feedback loop. Once the system is linear, we can apply the results from the previous section for controller synthesis. This method has been applied in designing a controller for fully actuated flexible robot arms [59], whose dynamics are feedback linearizable.

20.7.2.2 Differentially Flat Systems

Differentially flat systems are widely encountered in mechanical and robotics systems. For examples and comprehensive discussion, the reader is referred to [60–62]. The system in (20.37) is differentially flat if it has flat outputs. The outputs $y = (y_1, \ldots, y_m)$ are *flat outputs* if x and u can be written as functions of y and its time derivatives,

$$x = \Xi(y, \dot{y}, \ldots, y^{(\ell)}), \quad u = \Upsilon(y, \dot{y}, \ldots, y^{(\ell+1)}), \tag{20.38}$$

for some integer ℓ, and $(y, \dot{y}, \ldots, y^{(\ell)})$ are not constrained to satisfy a differential equation by themselves. In other words, any sufficiently smooth trajectory y is admissible as an output trajectory of the system.

A differentially flat system is related to a linear system, namely an ℓth order integrator chain, through the transformation in (20.38). In the context of our discussion, we can apply the results from the previous section for controller synthesis for the integrator chain. The controller for the nonlinear system (20.37) can then be obtained using the transformation in (20.38).

20.7.3 Controller Synthesis for Hybrid Systems

Consider the control hybrid automata defined in Sect. 20.3.1. For simplicity of the discussion, let us assume that the continuous state dynamics in each location is linear affine. That is, suppose that $L = \{\ell_0, \ell_1, \ldots, \ell_{|L|}\}$ and that for each discrete state $\ell_i \in L$ the continuous state dynamics is

$$\Sigma(\ell_i) : \dot{x} = A_{\ell_i} x + f_{\ell_i} + B_{\ell_i} u,$$

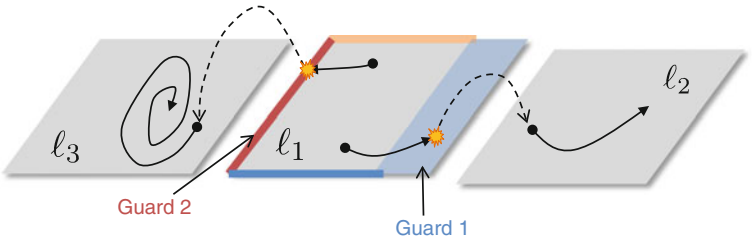

Fig. 20.7 Illustration of forcing and nonforcing guards of hybrid automata. From different initial states in location ℓ_1, the trajectories can undergo different evolution and transition to different locations. The set Guard 1 represents a nonforcing transition, while Guard 2 represents a forcing transition

where A_{ℓ_i}, B_{ℓ_i}, and f_{ℓ_i} are matrices with appropriate dimensions, and the pair (A_{ℓ_i}, B_{ℓ_i}) is stabilizable. Therefore, the continuous state dynamics in each location admits a control autobisimulation function ψ_i.

We assume there are two types of transitions, *forcing* and *nonforcing*. A forcing transition occurs immediately when the continuous state hits the guard, which is the case for autonomous hybrid automata in Sect. 20.4. See Fig. 20.7, where the guards of forcing transitions are illustrated as lines on the boundary of the invariant set of location ℓ_1. A nonforcing transition can happen at any time while the continuous state is in its guard. In Fig. 20.7, this is illustrated by the transition from location ℓ_1 to ℓ_2. In this case, the guard set is "thick," indicating that the transition can happen, but not necessarily as soon as the guard is hit. The occurrence of a nonforcing transition can be user-triggered (corresponding to a discrete control input), or externally triggered. Nonforcing transitions are useful to model events whose occurrence is not predetermined (uncertain) because it is to be triggered by the user/controller, or it is triggered externally at an a priori unknown time.

In defining the control specification, we define a set of initial state $Init \subset \mathcal{X} \times L$. We assume that there is a subset $Unsafe \subset \mathcal{X} \times L$ of unsafe states. A trajectory of the hybrid system corresponds to an unsafe execution if it enters the unsafe set. We also define the set $Goal \subset \mathcal{X} \times L$, which must be entered by the state trajectory. Again, the control problem is defined as finding the feedback control strategy that is guaranteed to bring any initial state in $Init$ to the $Goal$ set without entering the $Unsafe$ set.

Without any loss of generality, we can assume that the set $Init$ is contained in (the invariant set of) one location, called $\ell_{\text{init}} \in L$. If this is not the case, we can divide the problem into several subproblems, each with an $Init$ set contained in a specific location. Similarly, we can assume the $Goal$ is also entirely contained in one location, called $\ell_{\text{goal}} \in L$.

Controller synthesis for hybrid systems can be done using a hierarchical approach, which can be described in the following steps:

Step 1: Discrete Synthesis. We compute a discrete trajectory that starts in ℓ_{init} and ends in ℓ_{goal}. By discrete trajectory, we mean an alternating sequence of locations and transitions

$$\ell_{\text{init}} = \ell_0 \overset{e_0}{\to} \ell_1 \overset{e_1}{\to} \ell_2 \overset{e_2}{\to} \cdots \overset{e_{N-1}}{\to} \ell_N = \ell_{\text{goal}}. \tag{20.39}$$

Each transition e_i, $i \in \{0, \ldots, N-1\}$, is an element of E, originating in ℓ_i, and targeting ℓ_{i+1}. We require that each transition here is either forcing or user-triggered. Such a discrete trajectory is not necessarily unique, but at this step we only need one. The computation of such a discrete trajectory is a standard procedure in formal verification of discrete event systems [63]. For this purpose, there are many good algorithms and computational tools that can be used, such as STRIPS and PDLL [64].

Step 2: Continuous Synthesis. In this step, we synthesize the continuous controller for each of the visited locations ($\ell_{0,1,\ldots,N}$) in order to implement the computed discrete trajectory. In each location ℓ_i,, we define an initial set based on how ℓ_i is reached from ℓ_{i-1}. We then formulate the control problem of bringing the continuous state from this initial set to the interim goal set, which is the guard of transition e_i that will bring the state to location ℓ_{i+1} without entering the forbidden set. The forbidden set is defined as the union of *Unsafe* and the guards of other forcing transitions from ℓ_i. This is thus an instance of the control problem discussed in Sect. 20.7.1. If we are able to construct a continuous controller that implements the discrete trajectory, then the hybrid control problem is solved. Otherwise, we go back to Step 1, and compute another discrete trajectory.

Remark 20.11 The control problem that we discuss in this chapter is only concerned with the safety/reachability property. In addition, it is possible to formulate an optimal control problem in which a performance objective needs to be optimized while maintaining the safety/reachability property. For further discussion on the trajectory-based approach to this problem, the reader is referred to [65].

20.8 Concluding Remarks

We review some results that allow us to use trajectory-level reasoning in solving some problems in safety/reachability analysis of hybrid systems, controller synthesis for safety/reachability, and fault diagnosability of hybrid systems. The main feature of this approach is the possibility to break down a problem involving infinitely many trajectories of the system into one that only involves finitely many of them.

While we focus solely on safety/reachability property in this chapter, the discussion is actually generalizable to verification of and controller synthesis for other properties, such as those that can be described with temporal logics. In addition, there have also been extension work that consider stochasticity in the dynamics.

Acknowledgments The author wishes to acknowledge the support from the National Science Foundation through the CAREER grant CNS-0953976 and the grant CNS-1218109 for the research leading to results presented here. The results are summarized from the author's earlier work in collaboration with George Pappas, Antoine Girard, Georgios Fainekos, Alessandro D'Innocenzo, and graduate students Sina Afshari, Andrew Winn, and Yi Deng.

References

1. A.J. van der Schaft, J.M. Schumacher, *An Introduction to Hybrid Dynamical Systems* (Springer, London, 2000)
2. P. Tabuada, G.J. Pappas, P. Lima, Compositional abstractions of hybrid control systems. Discrete Event Dyn. Syst. **14**(2), 203–238 (2005). April
3. R. Alur, R. Grosu, I. Lee, O. Sokolsky, Compositional modeling for refinement for hierarchical hybrid systems. J. Logic Algebraic Program. **68**(1–2), 105–128 (2006)
4. J. Hu, Application of Stochastic Hybrid Systems in Power Management of Streaming Data, in *Proceedings of American Control Conference*, Minneapolis, USA (2006)
5. C. Kossentini, P. Caspi, Approximation, Sampling and Voting in Hybrid Computing Systems, in *HSCC 2006*, vol. 3927, LNCS, ed. by J.P. Hespanha, A. Tiwari (Springer, Heidelberg, 2006), pp. 363–376
6. J. Kapinski, A. Donzé, F. Lerda, H. Maka, S. Wagner, B.H. Krogh, Control Software Model Checking Using Bisimulation Functions for Nonlinear Systems, in *Proceedings of IEEE Conference Decision and Control*, Cancun, Mexico (2008)
7. R. Alur, G. Weiss, Regular Specifications of Resource Requirements for Embedded Control Software, in *Proceedings of 14th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 159–168 (2008)
8. R. Alur, A. D'Innocenzo, K. Johansson, G. Pappas, G. Weiss, Modeling and Analysis of Multihop Control Networks, in *Proceedings of 15th IEEE Real-Time and Embedded Technology and Applications Symposium* (2009)
9. C. Tomlin, G.J. Pappas, S. Sastry, Conflict resolution for air traffic management: a study in multi-agent hybrid systems. IEEE Trans. Autom. Control **43**(4), 509–521 (1998)
10. N. Lynch, High-level Modeling Andanalysis of An Air-traffic Management System, in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 1589. Springer, p. 3 (1999)
11. M. Prandini, J. Hu, J. Lygeros, S. Sastry, A probabilistic approach to aircraft conflict detection. IEEE Trans. Intell. Transp. Syst. **1**(4), 199–220 (2000)
12. J. Hu, M. Prandini, S. Sastry, Probabilistic Safety Analysis in Three Dimensional Aircraft Flight, in *Proceedings of 42nd IEEE Conference Decision and Control*, Maui, USA, pp. 5335–5340 (2003)
13. A. Bayen, P. Grieder, G. Meyer, C. Tomlin, Lagrangian delay predictive model for sector-based air traffic flow. AIAA J. Guidance Control. Dyn. **28**(5), 1015–1026 (2005)
14. H.A.P. Blom, J. Krystul, G.J. Bakker, A Particle System for Safety Verification of Free Flight in Air Traffic, in *Proceedings of IEEE Conference Decision and Control*, San Diego, USA (2006)
15. J. Lygeros, N. Lynch, Strings of Vehicles: Modeling and Safety Conditions, in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 1386. Springer, pp. 273–288 (1998)
16. A. Fehnker, Automotive Control Revisited: Linear Inequalities as Approximation of Reachable Sets, in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 1386. Springer, pp. 110–125, (1998)
17. A. Balluchi, F.D. Natale, A.L. Sangiovanni-Vincentelli, J.H. van Schuppen, Synthesis for Idle Speed Control of an Automotive Engine, in *HSCC 2004*, vol. 2993, LNCS, ed. by R. Alur, G.J. Pappas (Springer, Heidelberg, 2004), pp. 80–94

18. T. Dang, A. Donzé, O. Maler, Verification of Analog and Mixed-Signal Circuits Using Hybrid System Techniques, in *FMCAD 2004*, vol. 3312, LNCS, ed. by A.J. Hu, A.K. Martin (Springer, Heidelberg, 2004), pp. 21–36
19. G. Frehse, PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech, in *HSCC 2005*, vol. 3414, LNCS, ed. by M. Morari, L. Thiele (Springer, Heidelberg, 2005), pp. 258–273
20. G. Frehse, B.H. Krogh, R.A. Rutenbar, O. Maler, Time domain verification of oscillator circuit properties. Electron. Notes Theoret. Comput. Sci. **153**(3), 9–22 (2006)
21. G. Batt, B. Yordanov, R. Weiss, C. Belta, Robustness analysis and tuning of synthetic gene networks. Bioinformatics **23**(18), 2415–2422 (2007)
22. S. Drulhe, G. Ferrari-Trecate, H. de Jong, The switching threshold reconstruction problem for piecewise affine models of genetic regulatory networks. IEEE Trans. Autom. Control **53**(1), 153–165 (2008)
23. E. Cinquemani, A. Milias-Argeitis, S. Summers, J. Lygeros, Local Identification of Piecewise Deterministic Models of Genetic Networks, in *HSCC 2009*, vol. 5469, LNCS, ed. by R. Majumdar, P. Tabuada (Springer, Heidelberg, 2009), pp. 105–119
24. K. Amonlirdviman, N.A. Khare, D.R.P. Tree, W.-S. Chen, J.D. Axelrod, C.J. Tomlin, Mathematical modeling of planar cell polarity to understand domineering nonautonomy. Science **307**(5708), 423–426 (2005)
25. M.L. Bujorianu, J. Lygeros, Reachability Questions in Piecewise Deterministic Markov Processes, in *HSCC 2003*, vol. 2623, LNCS, ed. by O. Maler, A. Pnueli (Springer, Heidelberg, 2003), pp. 126–140
26. A. Abate, S. Amin, M. Prandini, J. Lygeros, S.S. Sastry, Computational Approaches to Reachability Analysis of Stochastic Hybrid Systems, in *HSCC 2007*, vol. 4416, LNCS, ed. by A. Bemporad, A. Bicchi, G. Buttazzo (Springer, Heidelberg, 2007), pp. 4–17
27. G. Batt, C. Belta, R. Weiss, Temporal logic analysis of gene networks under parameter uncertainty. IEEE Trans. Autom. Control **53**(1), 215–229 (2008)
28. D. Riley, X. Koutsoukos, K. Riley, Modeling and analysis of the sugar cataract development process using stochastic hybrid systems. IET Syst. Biol. **3**(3), 137–154 (2009)
29. E. De Santis, M.D. Di Benedetto, Editorial: observability and observer-based control of hybrid systems. Int. J. Robust. Nonlinear Control **19**, 1519–1520 (2009)
30. F. Zhao, X. Koutsoukos, H. Haussecker, J. Reich, P. Cheung, Monitoring and fault diagnosis of hybrid systems. IEEE Trans. Syst. Man Cybern. Part B **35**(6), 1225–1240 (2005)
31. N. Olivier-Maget, G. Hetreux, J.M. Le Lann, M.V. Le Lann, Model-based fault diagnosis for hybrid systems: application on chemical processes. Comput. Chem. Eng. **33**(10), 1617–1630 (2009)
32. A. Girard, G.J. Pappas, Approximate Bisimulation for Constrained Linear Systems, in *Proceedings of the IEEE Conference Decision and Control*, Seville, Spain (2005)
33. A. Girard, G.J. Pappas, Approximation metrics for discrete and continuous systems. IEEE Trans. Autom. Control **52**(5), 782–798 (2007)
34. A. Girard, A.A. Julius, G.J. Pappas, Approximate simulation relations for hybrid systems. Int. J. Discrete Event Dyn. Syst. **18**, 163–179 (2008)
35. A.A. Julius, Approximate Abstraction of Stochastic Hybrid Automata, in *HSCC 2006*, vol. 3927, LNCS, ed. by J.P. Hespanha, A. Tiwari (Springer, Heidelberg, 2006), pp. 318–332
36. A.A. Julius, A. Girard, G.J. Pappas, Approximate Bisimulation for a Class of Stochastic Hybrid Systems, in *Proceedings of American Control Conference*, Minneapolis, USA (2006)
37. A.A. Julius, G.J. Pappas, Approximate abstraction of stochastic hybrid systems. IEEE Trans. Autom. Control **54**(6), 1193–1203 (2009)
38. A.A. Julius, G.E. Fainekos, M. Anand, I. Lee, G.J. Pappas, Robust Test Generation and Coverage for Hybrid Systems, in *HSCC 2007*, vol. 4416, LNCS, ed. by A. Bemporad, A. Bicchi, G. Buttazzo (Springer, Heidelberg, 2007), pp. 329–342
39. A.A. Julius, G.J. Pappas, Trajectory Based Verification Using Local Finite-Time Invariance, in *HSCC 2009*, vol. 5469, LNCS, ed. by R. Majumdar, P. Tabuada (Springer, Heidelberg, 2009), pp. 223–236

40. W. Lohmiller, J.J.E. Slotine, On contraction analysis for nonlinear systems. Automatica **34**(6), 683–696 (1998)
41. S. Prajna, A. Papachristodoulou, P. Seiler, P.A. Parillo, SOSTOOLS and its Control Application, in *Positive Polynomials In Control*. Springer (2005)
42. Y. Deng, A. Rajhans, A.A. Julius, STRONG: A Trajectory-Based Verification Toolbox for Hybrid Systems, in *QEST 2013*, vol. 8054, LNCS, ed. by K. Joshi, M. Siegle, M. Stoelinga, P.R. D'Argenio (Springer, Heidelberg, 2013), pp. 165–168
43. Y. Deng, A.A. Julius, Safe Neighborhood Computation for Hybrid System Verification, in *Proceedings of 4th Workshop on Hybrid Autonomous Systems, ser. Electronic Proceedings in Theoretical Computer Science*, vol. 174. Springer, pp. 1–12 (2014)
44. J.W. Polderman, J.C. Willems, *Introduction to Mathematical Systems Theory: a Behavioral Approach* (Springer, New York, 1998)
45. M.D. Di Benedetto, S. Di Gennaro, A. D'Innocenzo, Discrete state observability of hybrid systems. Int. J. Robust. Nonlinear Control **19**, 1564–1580 (2009)
46. Y. Deng, A.D'Innocenzo, S. Di Gennaro, M.D. Di Benedetto, A.A. Julius, Verification of hybrid automata diagnosability with measurement uncertainty, provisionally accepted to the IEEE Trans. Autom. Control (2015)
47. Y. Deng, A. D'Innocenzo, A.A. Julius, Probabilistic Diagnosability of Hybrid Systems, in *Proceedings of ACM 18th International Conference Hybrid Systems: Computation and Control*, Seattle, WA, pp. 88–97 (2015)
48. A.A. Julius, Trajectory-based Controller Design for Hybrid Systems with Affine Continuous Dynamics, in *Proceedings of IEEE Conference Automation Science and Engineering*, Toronto, Canada, pp. 1007–1012 (2010)
49. A.A. Julius, S. Afshari, Using Computer Games for Hybrid Systems Controller Synthesis, in *Proceedings of 49th IEEE Conference Decision and Control*. Atlanta, Georgia, pp. 5887–5892 (2010)
50. Z. Artstein, Stabilization with relaxed controls. Nonlinear Anal. **15**(11), 1163–1170 (1983)
51. E.D. Sontag, A 'universal' construction of Artstein's theorem on nonlinear stabilization. Syst. Control Lett. **13**(2), 117–123 (1989)
52. W.L. Brogan, *Modern Control Theory* (Prentice Hall International, New Jersey, 1991)
53. B. Friedland, *Control System Design: an Introduction to State-Space Methods*. Dover (2005)
54. S. Boyd, L. El Ghaoui, E. Feron, V. Balakrishnan, *Linear Matrix Inequalities in Systems and Control Theory* (SIAM, Philadelphia, 1994)
55. A.K. Winn, A.A. Julius, Feedback Control Law Generation for Safety Controller Synthesis, in *Proceedings of IEEE Conference Decision and Control*, Florence, Italy, pp. 3912–3917 (2013)
56. A.A. Julius, A.K. Winn, Safety Controller Synthesis Using Human Generated Trajectories: Nonlinear Dynamics with Feedback Linearization and Differential Flatness, in *Proceedings of American Control Conference*, Montreal, Canada, pp. 709–714 (2012)
57. H. Nijmeijer, A.J. van der Schaft, *Nonlinear Dynamical Control Systems* (Springer Verlag, New York, 1990)
58. H.K. Khalil, *Nonlinear Systems, 3rd edn.* (Prentice Hall, 2002)
59. S. Saha, A.A. Julius, Trajectory-based Formal Controller Synthesis for Multi-link Robots with Elastic Joints, in *Proceedings of IEEE Conference Decision and Control*, Los Angeles, CA, pp. 830–835 (2014)
60. M.J. van Nieuwstadt, R.M. Murray, Real-time trajectory generation for differentially flat systems. Int. J. Robust Nonlinear Control **8**(11), 995–1020 (1998)
61. J. Levine, *Analysis and Control of Nonlinear Systems: a Flatness Based Approach*. (Springer, 2009)
62. H. Sira-Ramirez, S. Agrawal, *Differentially Flat Systems*. (Marcel Dekker Inc., New york, 2004)
63. E.M. Clarke, O. Grumberg, D.A. Peled, *Model Checking*. (MIT Press, 1999)
64. S.J. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*. (Prentice Hall, 2003)
65. A.K. Winn, A.A. Julius, Optimization of Human Generated Trajectories for Safety Controller Synthesis, in Proceedings of American Control Conference, Washington DC, pp. 4374–4379 (2013)