Hybrid Dynamical Systems

Aaron M. Johnson

June 30, 2026

A *hybrid dynamical system* is a system that contains both continuous and discrete states [1–5]. These are sometimes called *hybrid automata* [6,7] *differential automata* [8], or simplified as just *hybrid systems*. In robotics, the continuous states are positions and velocities while the discrete states include different contact conditions, hard stops, and switching behaviors. But hybrid systems are also commonly used to model electrical circuits with diodes or transistors, HVAC or industrial control systems, neural circuits, and more.

The general concept of a hybrid dynamical system is quite simple, though the careful mathematical definition can get tricky. Consider a dynamical system like the one shown on the right, where the system state lies in one of two *discrete modes*, I or J. When the system is in mode I, the continuous portion of the system state lies in the *domain* D_I and follows the *flow* (vector field) F_I. At some point, the state reaches a *guard set* $G_{\mathrm{I,J}}$ that triggers a *reset map* transition $R_{\mathrm{I,J}}$ into a new mode J with domain D_J. The system then follows the flow F_J until reaching a different guard $G_{\mathrm{J,I}}$ that triggers a reset $R_{\mathrm{J,I}}$ back to mode I.

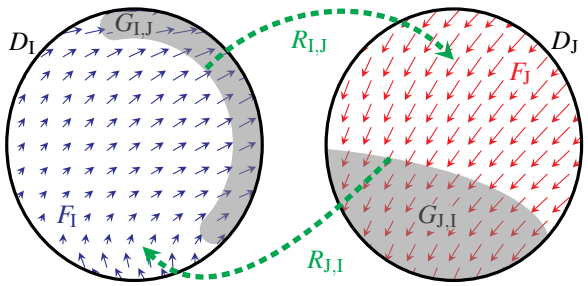


Figure 1: Example hybrid system.

The difficulty of hybrid dynamical systems modeling is that there are a lot of separate pieces to define and keep track of, so the hybrid system definition requires a lot of notation and bookkeeping. Furthermore, we would like to define the system in a way that is self-consistent and avoids algebraic or numerical challenges. Common questions that arise include: Do the domains all have to be the same dimension or subsets of a common domain? Do the guards only exist on the boundary of the domains or on the interior as well? What happens if you reach the boundary of a domain that is not part of a guard set, or is part of two guard sets?

In this chapter we develop a general framework for hybrid systems, based on [1], that addresses many of these issues and results in a well behaved system. Then, in the next chapter, we will fill in the details of the hybrid system for robotic systems that follow rigid body dynamics with contact.

A note on notation in this chapter: Scripted capital letters ($\mathcal{D}, \mathcal{F}, \mathcal{T}$, etc) represent the "hybrid" version of functions or sets, while regular italicized letters (D, F, T, etc) the non-hybrid versions, typically indexed by a mode name in non-italicized letters (I, J, etc) or an index variable i.

1 Hybrid Dynamical System Definition

A *hybrid dynamical system* is a tuple (i.e. a collection of components),

$$\begin{aligned}
&\mathcal{H} := (\mathcal{J}, \Gamma, \mathcal{D}, \mathcal{F}, \mathcal{G}, \mathcal{R}) \quad \textit{where,} \\
&\qquad \mathcal{J} := \{\mathrm{I}, \mathrm{J}, \ldots, \mathrm{K}\} \\
&\qquad \Gamma \subset \mathcal{J} \times \mathcal{J} \\
&\qquad \mathcal{D} := \coprod_{\mathrm{I} \in \mathcal{J}} D_{\mathrm{I}} \\
&\qquad \mathcal{F} : \mathcal{D} \to T\mathcal{D}, \qquad F_{\mathrm{I}} := \mathcal{F}\big|_{D_{\mathrm{I}}} : D_{\mathrm{I}} \to TD_{\mathrm{I}} \\
&\qquad \mathcal{G} := \coprod_{(\mathrm{I},\mathrm{J}) \in \Gamma} G_{\mathrm{I},\mathrm{J}}, \qquad G_{\mathrm{I},\mathrm{J}} \subseteq D_{\mathrm{I}} \\
&\qquad \mathcal{R} : \mathcal{G} \to \mathcal{D}, \qquad R_{\mathrm{I},\mathrm{J}} := \mathcal{R}\big|_{G_{\mathrm{I},\mathrm{J}}} : G_{\mathrm{I},\mathrm{J}} \to D_{\mathrm{J}}
\end{aligned} \qquad \text{Hybrid Dynamical System} \quad (1)$$

where $\mathcal{J}$ are the discrete modes, Γ are the discrete transitions, $\mathcal{D}$ are the continuous domains, $\mathcal{F}$ are the flows, $\mathcal{G}$ are the guard sets, and $\mathcal{R}$ are the rest maps, all of which are further defined below. We can impose smoothness on this structure, which we call a C^r *hybrid dynamical system* for smoothness class $r \in \mathbb{N} \cup \{\infty, \omega\}$, for example a C^2 hybrid dynamical system would be one the components are twice differentiable.

At a high level, the hybrid system considers the discrete state $\mathcal{J}$ and dynamics Γ, the continous state $\mathcal{D}$ and dynamics $\mathcal{F}$, as well as where $\mathcal{G}$ and how $\mathcal{R}$ these interact (i.e. when the continuous state triggers a change in the discrete state).

Note that many of these terms use the *disjoint union* operator $\coprod$, a way of collecting different sets without mixing their elements. Formally, it is defined as,

$$\boxed{D_{\mathrm{I}} \coprod D_{\mathrm{J}} := \{(x, \mathrm{I}) : x \in D_{\mathrm{I}}\} \cup \{(x, \mathrm{J}) : x \in D_{\mathrm{J}}\}} \qquad \text{Disjoint Union} \quad (2)$$

(recall, in contrast, that $D_{\mathrm{I}} \cup D_{\mathrm{J}} := \{x : x \in D_{\mathrm{I}}\} \cup \{x : x \in D_{\mathrm{J}}\}$ and that $D_{\mathrm{I}} \times D_{\mathrm{J}} := \{(x, y) : x \in D_{\mathrm{I}}, y \in D_{\mathrm{J}}\}$). Each element of the combined set is "labeled" with which original set it came from, so that the elements are kept separate.

For example, if both domains were the positive natural numbers ($D_{\mathrm{I}} = D_{\mathrm{J}} = \{1, 2, 3, \ldots\}$), then,

$$\begin{aligned}
D_{\mathrm{I}} \coprod D_{\mathrm{J}} &= \{(1, \mathrm{I}), (1, \mathrm{J}), (2, \mathrm{I}), (2, \mathrm{J}), \ldots\} \\
D_{\mathrm{I}} \cup D_{\mathrm{J}} &= \{1, 2, 3, \ldots\} \\
D_{\mathrm{I}} \times D_{\mathrm{J}} &= \{(1, 1), (1, 2), (2, 1), (2, 2), \ldots\}
\end{aligned}$$

Here, we have defined a different configuration space for each domain, and we want to keep those states separate even if they have the same value. Thus, the full state of the system includes the continuous state x and the discrete mode I that the system is in at that time.

1.1 Modes

$\mathcal{J} := \{\mathrm{I}, \mathrm{J}, \ldots, \mathrm{K}\} \subset \mathbb{N}$ is the finite set of *discrete modes* or *discrete states*. The system state will be in one of the modes $\mathrm{I} \in \mathcal{J}$ at all times. These modes define the different modes of operation of the system, differentiating states that follow different dynamics (e.g. what contact forces are applied) or reside in different state spaces (e.g. which state constraints are active).

$$\mathcal{J} = \{\mathrm{I}, \mathrm{J}\}$$

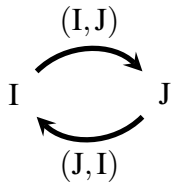
Figure 2: The set of discrete modes from Figure 1.

1.2 Transitions

$\Gamma \subset \mathcal{J} \times \mathcal{J}$ is the set of *discrete transitions*, i.e. possible transitions between discrete modes. This forms a directed graph structure over $\mathcal{J}$, with edges $(\mathrm{I},\mathrm{J}) \in \Gamma$ representing a transition from mode I to mode J. This graph is generally not a tree however since, e.g., both (I,J) and (J,I) can be included.

Just because a transition is included in Γ does not necessarily mean that it can be achieved by the system. Instead, we can define the subset of *achievable transitions* $\tilde{\Gamma}$, based on the existence of a nonempty guard set (defined below),

$$\tilde{\Gamma} := \{(\mathrm{I},\mathrm{J}) \in \Gamma : G_{\mathrm{I},\mathrm{J}} \neq \varnothing\} \qquad \text{Achievable Transitions} \tag{3}$$



$$\Gamma = \{(\mathrm{I},\mathrm{J}), (\mathrm{J},\mathrm{I})\}$$

Figure 3: The discrete transitions from Figure 1.

1.3 Domains

$\mathcal{D} := \coprod_{\mathrm{I} \in \mathcal{J}} D_{\mathrm{I}}$ is the collection of *domains*, i.e. continuous state spaces. We define a separate domain D_{I} for each discrete mode $\mathrm{I} \in \mathcal{J}$, so that when the system is in mode I, the continuous state is an element of D_{I}. Each domain is a state space, e.g. $\mathbb{R}^n$ or $SE(3)$, that captures the full state of the system in that discrete mode (generally including position and velocity states for mechanical systems). Note that the domains do not have to have the same topology or be the same dimension as each other.

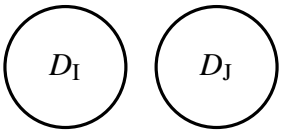


Figure 4: The domains from Figure 1.

The domain definitions may include inequalities, and as such each D_{I} is a *manifold with corners* [9,10] (i.e. a set that includes smooth boundaries and intersections of boundaries). The smoothness of each domain matches that of the overall hybrid system, so each D_{I} is a C^r *manifold with corners*. The combined collection of domains $\mathcal{D}$ is a *hybrid* C^r *manifold with corners* [1, App. D] [11].

1.4 Flows

$\mathcal{F} : \mathcal{D} \to T\mathcal{D}$ is the *flow* or *vector field*, that restricts down to $F_{\mathrm{I}} := \mathcal{F}|_{D_{\mathrm{I}}} : D_{\mathrm{I}} \to TD_{\mathrm{I}}$ for each discrete mode $\mathrm{I} \in \mathcal{J}$. The flow captures the continuous dynamics of the system, such that when the system is in mode I at state $x \in D_{\mathrm{I}}$, then the evolution of the state follows $\dot{x} = F_{\mathrm{I}}(x)$.

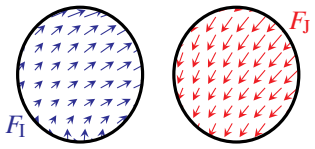


Figure 5: The flows or vector fields from Figure 1.

Depending on the system, we may want to extend this definition to allow for time-varying dynamics, in which case $\mathcal{F} : \mathcal{T} \times \mathcal{D} \to T\mathcal{D}$ and $\dot{x} = F_{\mathrm{I}}(t,x)$, where the hybrid time domain $\mathcal{T}$ is defined below. This is mostly a notational convenience, as we can always include an extra dimension in the state for time with the dynamics $\dot{t} = 1$. Similarly, it may be convenient to make explicit the dependence on control input $u \in U_I$, with the space of control inputs across all modes $\mathcal{U} := \coprod_{\mathrm{I} \in \mathcal{J}} U_I$, in which case $\mathcal{F} : \mathcal{T} \times \mathcal{D} \times \mathcal{U} \to T\mathcal{D}$ and $\dot{x} = F_{\mathrm{I}}(t,x,u)$ (though this is a trivial extension if we know the control policy as a function of time and state, $u(t,x)$, and thus $\dot{x} = F_{\mathrm{I}}(t,x,u(t,x))$).

The flow in each domain F_{I} should follow the same smoothness as the overall system, and thus should be a C^r smooth function. Note that this generally requires the control policy $u(t,x)$ to also be C^r in order for the resulting time-varying vector field to be. The combined function $\mathcal{F}$ is then a C^r *hybrid vector field* [1, App. D].

1.5 Guards

$\mathcal{G} := \coprod_{(\mathrm{I},\mathrm{J}) \in \Gamma} G_{\mathrm{I},\mathrm{J}}$ is a collection of *guard sets* or *jump sets*, one for each transition $(\mathrm{I},\mathrm{J}) \in \Gamma$, where each $G_{\mathrm{I},\mathrm{J}} \subseteq D_{\mathrm{I}}$. The guard is the set of points in the domain that trigger a discrete transition to a different mode, J. As such, it is a subset of the domain of the mode that is the origin of that transition (so guard set $G_{\mathrm{I},\mathrm{J}}$ is a subset of domain D_{I} regardless of which J is indicated). Depending on the system, we may want the guard set to be time varying, $G_{\mathrm{I},\mathrm{J}}(t)$, and dependent on the control input, $G_{\mathrm{I},\mathrm{J}}(t,u)$.

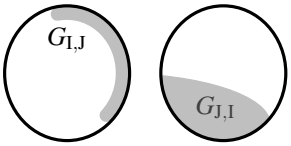


Figure 6: The guard sets from Figure 1.

It is also common to define the guard set as a regular sublevel set of a guard function $g_{\mathrm{I},\mathrm{J}}$, such that $G_{\mathrm{I},\mathrm{J}}(t) := \{x \in D_{\mathrm{I}} : g_{\mathrm{I},\mathrm{J}}(t,x) \leq 0\}$. In that case, we require the

function $g_{\mathrm{I,J}}$ to be C^r and nonsingular on the boundary of the set, i.e. if $g_{\mathrm{I,J}}(t,x) = 0$ then $\mathrm{D}_x g_{\mathrm{I,J}}(t,x) \neq 0$. However, not every hybrid system has guard sets that can satisfy this guard function definition.

It is also useful to define the *outlet set* of a given domain, $G_{\mathrm{I}} \subseteq D_{\mathrm{I}}$, which is the set of all points that trigger *some* transition to another mode, $G_{\mathrm{I}} := \cup_{\mathrm{J}} G_{\mathrm{I,J}}$.

1.6 Resets

$\mathcal{R} : \mathcal{G} \to \mathcal{D}$ is the *reset map* or *transition function* or *jump map*, which restricts down to the individual reset function $R_{\mathrm{I,J}} := \mathcal{R}|_{G_{\mathrm{I,J}}} : G_{\mathrm{I,J}} \to D_{\mathrm{J}}$ for each transition $(\mathrm{I},\mathrm{J}) \in \Gamma$. Once the continuous state x has entered the guard function $G_{\mathrm{I,J}}$, that triggers the corresponding reset map $R_{\mathrm{I,J}}(x)$ to transitions the continuous state $x \in D_{\mathrm{I}}$ into the new domain D_{J} while the discrete state changes to mode J.

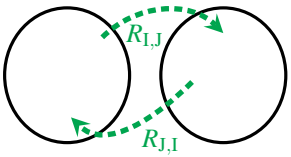


Figure 7: The set of reset maps from Figure 1.

For time-varying systems, the reset map may also be a function of time, $\mathcal{R} : \mathcal{T} \times \mathcal{G} \to \mathcal{D}$ with components $R_{\mathrm{I,J}}(t,x)$. For smooth hybrid systems, we require the functions $R_{\mathrm{I,J}}$ to be C^r maps and the combined $\mathcal{R}$ to be a C^r hybrid map.

1.7 Other Definitions

There is a special class of hybrid systems called *switched systems* or *controlled switching systems* where the transition is dictated by the control input instead of the state [3]. Here, we focus on *event-driven hybrid systems* or *autonomous switching* systems, with state-dependent guards (transitions). When the state reaches a guard set, the system must transition right away – there is no decision to make.

In many formulations of a hybrid system, the continuous state is taken from a general set of states D. Indeed, typically we have a general state space for a robot and any objects and each individual domain is a subset of this space[1]. Then, individual domains for a given mode are defined as subsets of the general domain, $D_{\mathrm{I}} \subset D$, or as a function mapping from the discrete state to the continuous domain, $\mathrm{I} \mapsto D_{\mathrm{I}}$ [6,7]. These restrictions on the general domain are sometimes called *invariants*.

Some definitions also include a restricted set of possible initial conditions $\mathrm{Init} \subseteq \mathcal{D}$, e.g. [6,7]. Here we assume any state is a possible initial condition.

2 Execution of a Hybrid System

A *hybrid execution* $\mathcal{X}$, or *simulation*, of the hybrid system determines the evolution of the state over time. But in order to describe a hybrid dynamical system execution, we must first define the *hybrid time domain* over which it evolves [1,5,7,12,13],

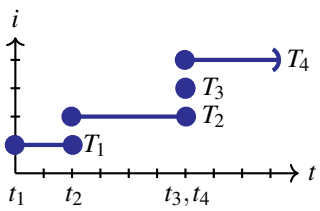
$$\boxed{\mathcal{T} := \coprod_{i=1}^{n} T_i, \quad T_i = [t_i, t_{i+1}] \subset \mathbb{R}, \quad T_i \cap T_{i+1} = t_{i+1}} \qquad \text{Hybrid Time Domain} \tag{4}$$

where n is the (possibly infinite) number of time intervals, $T_i \subset \mathbb{R}$ is a closed interval, and $T_i \cap T_{i+1}$ consists of a single time, t_{i+1}. That is, the hybrid time domain consists of several consecutive time intervals, or *epochs*, where each epoch ends at a time when the next one begins. The final epoch may continue until $t = \infty$.

These epochs represent the time period over which the hybrid system stays in a single mode, and the transition times $t_{i+1} = T_i \cap T_{i+1}$ are the *event times* when a reset map is applied. Note that in this formulation, these event times are included in both the previous and next epochs, allowing the trajectory on each domain to include both endpoints. This is acceptable because we assume that the transition takes no time. It is common to refer to this time as t_{i+1}^- in the previous domain and t_{i+1}^+ in the next domain, though these are the same numerical value of time.

[1] Of course, this can always be achieved trivially through a cross product, $D = D_1 \times D_2 \times \ldots$.

Consider the example hybrid time domain at right. We have $n = 4$ epochs (time intervals). The first $T_1 = [0,2]$ and the second $T_2 = [2,6]$ intersect at the event time $t_2 = 2$. We say that the first epoch ends at $t_2^- = 2$ and the second epoch begins at $t_2^+ = 2$ to clarify these overlapping times. Note that the third epoch consists of only a single point in time – when the system transitioned from the second mode to the third mode, the reset map resulted in the state being in the guard for another transition. That is, $R_{2,3}(t_3^-, x(t_3^-)) \in G_{3,4}(t_3^+)$. Finally, the fourth epoch starts at that same time, $t_4 = t_3$, and extends to infinity, as the system stays in the fourth mode forever.



$T_1 = [0,2], \quad T_2 = [2,6],$
$T_3 = \{6\}, \quad T_4 = [6,\infty)$

Figure 8: A hybrid time domain.

The hybrid execution generally involves following the flow in some domain until reaching a guard, then applying a reset map to a new domain, and finally repeating the cycle by continuing on with the dynamics in that new domain. Formally, the execution is defined as,

$$\begin{aligned}
&\mathcal{X} : \mathcal{T} \to \mathcal{D},\ s.t. \quad \forall T_i \in \mathcal{T},\ t_{i+1} = T_i \cap T_{i+1}, \\
&\qquad 1. \quad \mathcal{X}\big|_{T_i} \in D_i \\
&\qquad 2. \quad \tfrac{d}{dt}\mathcal{X}\big|_{T_i}(t) = \mathcal{F}(\mathcal{X}(t)) = F_i(\mathcal{X}(t)) \\
&\qquad 3. \quad \mathcal{X}\big|_{T_i}(t_{i+1}) \in G_{i,i+1} \\
&\qquad 4. \quad \mathcal{X}\big|_{T_{i+1}}(t_{i+1}) = R_{i,i+1}(\mathcal{X}\big|_{T_i}(t_{i+1})) \in D_{i+1} \\
&\qquad 5. \quad \mathcal{X}\big|_{T_i}(t) \notin G_i\ \forall t \neq t_{i+1}
\end{aligned}$$

Hybrid Execution (5)

The function $\mathcal{X}$ is a smooth (C^r) hybrid map from the hybrid time domain $\mathcal{T}$ to the hybrid state domain $\mathcal{D}$. There are then five conditions on this function:

1. The first condition states that within a single epoch T_i, the state of the system lies in a single domain D_i. Thus, each epoch represents a time interval when the hybrid system is in a single discrete mode i.

2. The second condition is that within an epoch, the dynamics of the system follows the flow or vector field for that mode, F_i. Thus, within each epoch, the system behaves as a conventional (non-hybrid) system. (There is some ambiguity as to whether this time derivative exists for epochs that are only a single time instant, however we consider the system as following the dynamics of that mode even if no time elapses in that mode.)

3. The third condition is that the state at the end of an epoch lies in the guard set $G_{i,i+1}$ for the transition to the next mode. This is the end of the time period where the state resides in domain D_i.

4. The fourth condition is that the state at the start of the next epoch is determined by applying the reset map $R_{i,i+1}$ for that transition to the state at the end of the prior epoch. Note that this new state is in the new domain D_{i+1}, thus beginning the next epoch in the new mode $i+1$. This enforces that the discrete dynamics are applied correctly in the execution.

5. Finally, the fifth condition is that the state is not in any guard at times that are not the transition times. Without this, the system would have to change modes in the middle of an epoch.

3 Properties of Hybrid Systems

There are many properties that people have studied about the structure and consistency of a particular hybrid system or hybrid execution. Here, we briefly summarize these properties and discuss their implications. For a more formal treatment of these properties, see [1,5,7].

3.1 Disjoint Guards

We would like the guard sets to be disjoint. If they are not, then we could reach a state that is an element of multiple guards and it would not be clear what the correct next state should be, e.g. [5]. Thus, we generally require

$$G_{\mathrm{I,J}} \cap G_{\mathrm{I,K}} = \varnothing \quad \forall (\mathrm{I},\mathrm{J}),(\mathrm{I},\mathrm{K}) \in \Gamma \qquad \text{Disjoint Guards} \tag{6}$$

3.2 Guards on Boundaries

Some formulations of hybrid systems impose a stricter condition on the guards than simply $G_{\mathrm{I,J}} \subseteq D_{\mathrm{I}}$. Instead, we could require that the guard lies on the boundary of the domain, $G_{\mathrm{I,J}} \subseteq \partial D_{\mathrm{I}}$ [2,11]. For example, in Figures 1 and 6, $G_{\mathrm{I,J}}$ lies only on the boundary while $G_{\mathrm{J,I}}$ does not. There may not be a reason to define the domain beyond the start of the guard set, and we could instead redefine the domain in a way that removes the interior of the guard set, leaving only the remaining guard as the newly formed edge of the domain. However, this does not work in all cases[2] and we find the restriction unnecessary. Furthermore, there are many cases where a reset map will take values in the interior of a guard set, necessitating guards that are not simply on the boundary of the domain. See the section on reset into guards, below.

The reverse condition, however, is more helpful – should the hybrid system be required to have every point on the boundary of a domain be a member of a guard set, $\partial D_{\mathrm{I}} \subseteq G_{\mathrm{I}}$? Reaching a boundary point that is not a member of a guard will cause problems with the nonblocking property, below, as it is not defined how to continue in this case. However, to avoid this problem we only need boundary points whose flow points "outward" to be a member of a guard. In that case, every point on the boundary either continues into the interior of that domain or is a member of a guard that will reset the state into a new domain. Domains D_{I} and D_{J} from Figure 1 have guards that cover all of the boundary points whose flows point out of the domain.

Formally, if the domain has a boundary function $D_{\mathrm{I}} = \{x : a(x) \geq 0\}$, then we require,

$$\{x \in D_{\mathrm{I}} : a(x) \prec 0\} \subseteq G_{\mathrm{I}} \qquad \text{Guards on Boundaries} \tag{7}$$

Recall from the complementarity chapter that $\prec$ is the trending negative condition, consisting of all points whose first non-zero derivative is negative (and therefore whose value is about to become negative). This condition ensures that all points whose flow will take us out of the domain must be a member of a guard set. A stricter condition, that may be easier to verify, is simply that all points on the boundary must be a member of a guard, $\{x \in D_{\mathrm{I}} : a(x) = 0\} \subseteq G_{\mathrm{I}}$.

3.3 Maximal Execution

An execution $\mathcal{X}$ is *maximal* if it cannot be extended to an execution over a longer hybrid time domain. Ideally, this means that the execution can continue indefinitely. However, it may also mean that we have reached a point where we cannot continue the execution of the hybrid system, in which case we say that the system is *blocking*. This brings us to our next property, nonblocking.

3.4 Nonblocking

A hybrid system is *nonblocking* or *complete* if, for every initial condition $x \in \mathcal{D}$, the hybrid time domain $\mathcal{T}$ for any maximal execution $\mathcal{X}$ from that starting point $\mathcal{X}|_{\mathcal{T}_1}(0) = x$ will either have $N = \infty$ or $t_{N+1} = \infty$. That is, we can continue the execution for either an infinite number of time intervals (see Zeno, below) or an infinite amount of time. Put another way, nonblocking requires the *existance* of an execution at every point. To prove that our hybrid system is nonblocking, we simply need to ensure that we never reach a point where we would leave the current domain that is not in a guard set, i.e. (7), and that the reset map will put us into a valid domain of the next discrete mode.

[2]For example, a domain of $\mathbb{R}^2$ with a guard set consisting of the interval $[0,1]$ on the x axis cannot be changed to exist only on the boundary of the domain. However, the presence of such interior codimension-1 guard sets may cause other problems, e.g. numerical detection.

3.5 Deterministic

A hybrid system is *deterministic* if, for every initial condition $x \in \mathcal{D}$, there exists a unique maximal execution $\mathcal{X}$ from that starting point $\mathcal{X}|_{T_1}(0) = x$. The key word here is *unique*, so from any given state there is only one possible trajectory to continue from there. This requires our continuous and discrete dynamics functions, $\mathcal{F}$ and $\mathcal{R}$, to be deterministic (i.e. provide a single value). But for the whole system to be determinisitc, we must also never reach multiple guards simultaneously, (6).

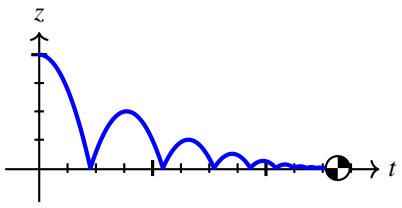
Here, we are focusing on event-driven hybrid systems, where the state reaching a guard set requires a transition. This is required to ensure our system is deterministic – there is no choice to be made about resetting into a new mode.

Note that there are times when we may want to model a nondeterministic hybrid system, e.g. [12], however in this book we will focus on developing a deterministic hybrid system for rigid body dynamics with impact.

3.6 Zeno

An execution of a hybrid dynamical system over a hybrid time trajectory $\mathcal{T}$ is *Zeno* if it undergoes an infinite number of transitions in finite time [7, Def. II.3], i.e. $n = \infty$ and $\sum_{i=1}^{n} |T_i| < \infty$. This behavior is named for the ancient Greek philosopher Zeno of Elea, who noted that before an object can travel a certain distance, it must first travel half that distance. But then it must first travel half of *that* distance, so a quarter of the original distance, and so on. With an infinite number of places to visit in finite time, motion must be impossible. For general motion, calculus tells us that infinitesimal motion over infinitesimal time can result in a finite velocity, and therefore motion is indeed possible. However, in hybrid systems, resolving a hybrid transition requires a finite amount of computations no matter how short each continuous phase was, and so we cannot compute an infinite number of these transitions. A full treatment of Zeno phenomena is quite difficult and beyond the scope of this chapter, see for example [1,7,14–16], but we will give a brief overview of the issues and approaches here.

Consider the case of a bouncing ball, as shown at right. Here, the ball starts at rest at a height of $h_0 = 1$ m, gravity is set to $g = 9.8$ m/s^2, and the ball loses half of its energy on each bounce ($e = 1/\sqrt{2}$). However, that means that no matter how small the impacting velocity is, it will always bounce up again. After 20 bounces, each aerial phase lasts less than one millisecond. Even after an infinite number of transitions, only about 2.633 seconds will have elapsed.

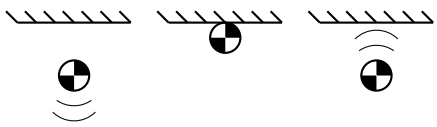


There are a few ways to address Zeno behavior in hybrid systems. First, if we can detect that we are in such an execution, we can *complete* the event sequence by computing the state and the event time that the execution is converging to and simply jump to that point [15]. Alternatively, we can *truncate* the execution by modifying the dynamics to cut off Zeno solutions. For the bouncing ball, one way to do this would be to change the impact law to only rebound with a sufficiently large velocity [1]. However, it is difficult to prove that Zeno trajectories are completely impossible for a given set of hybrid dynamics.

3.7 Reset into Guards

At first it might appear that we would not want a reset map to put the system state in a new guard set, $R_{\mathrm{I,J}}(x) \in G_{\mathrm{J,K}}$, thus immediately triggering another discrete transition. After all, couldn't we have just skipped this middle step and reset into the final mode directly. This property can also take the form of requiring a minimal "dwell time" in a given mode. However, in practice there is generally no reason to prohibit this and the seemingly extra discrete transition can be quite helpful for modeling. The presence of these double transitions is why the definition of the hybrid time domain allows for an interval to consist of a single time, e.g. T_3 in Figure 8.

For example, consider the case of a ball thrown at the ceiling. Assuming plastic impact, any vertical velocity will be zeroed out by the initial collision. However, once the ball is touching the ceiling with zero velocity, it will immediately start to fall away from the ceiling due to gravity. Thus, it will make a second transition back to the no-contact mode. Allowing the first reset map to place the state into the interior of the guard for the second reset makes modeling this



transition easier as it decouples the impact and liftoff conditions. Furthermore, eliminating the middle step of resetting into the constrained mode would require the overall reset map to take the state from the unconstrained mode back to the unconstrained mode while experiencing an impulse from an external constraint (though this looks a lot like elastic impact).

However, allowing a reset map into a guard set brings up the possibility of a potential issue of *zero time Zeno* or *chattering Zeno*, where an execution undergoes an infinite number of transitions while covering zero time (not simply finite time, as in regular Zeno). This can happen if a reset map takes the state into another guard set, whose reset map brings the state back to where it started. To ensure this does not happen, in [1] the hybrid system definition was checked to prove that no more than two transitions could happen sequentially.

4 Practice Problems

1) Consider a system consisting of a planar point particle with mass m described by state $q = [x,z]^T$ and two constraints, $a_G(q) = z \geq 0$ and $a_H(q) = -x\sin(\theta) + z\cos(\theta) \geq 0$ (see the Complementarity chapter). What are the set of modes, $\mathcal{J}$, and discrete transitions, Γ, for this system? What are the achievable transitions, $\tilde{\Gamma}$? How does the parameter θ affect the structure of the hybrid system?

2) Write out a full hybrid system definition, $\mathcal{H} = (\mathcal{J},\Gamma,\mathcal{D},\mathcal{F},\mathcal{G},\mathcal{R})$, for a basic HVAC thermostat. Assume the user provides a desired temperature setpoint and that the thermostat turns on the heat or air conditioning when the room temperature deviates from the set point by more than 4°, and turns them off when the temperature is within 2°. If you would like to simplify the thermal dynamics of the system, you may model them as a linear function of time in each mode. Which of the properties in Section 3 holds for your system?

3) Consider the bouncing ball example from Section 3.6. If we wanted to apply the completion approach, what is the exact time, state, and mode of the system at the end of the Zeno trajectory? *(Hint: The time period of each aerial phase is equal to the coefficient of restitution times the period of the previous phase, $|T_{i+1}| = e|T_i|$.)* If we wanted to apply the truncation approach, propose a change to one or more components of its hybrid system definition that will result in only a finite number of transitions before reaching the post-Zeno state and mode.

References

[1] A. M. Johnson, S. E. Burden, and D. E. Koditschek, "A hybrid systems model for simple manipulation and self-manipulation systems," *International Journal of Robotics Research*, vol. 35, no. 11, pp. 1354–1392, September 2016.

[2] A. Back, J. Guckenheimer, and M. Myers, "A dynamical simulation facility for hybrid systems," in *International Hybrid Systems Workshop*. Springer, 1991, pp. 255–267.

[3] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: Model and optimal control theory," *IEEE Transactions on Automatic Control*, vol. 43, no. 1, pp. 31–45, 2002.

[4] S. N. Simic, K. H. Johansson, J. Lygeros, and S. Sastry, "Towards a geometric theory of hybrid systems," *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications and Algorithms*, vol. 12, no. 5–6, pp. 649–687, 2005.

[5] R. Goebel, R. G. Sanfelice, and A. R. Teel, "Hybrid dynamical systems," *IEEE Control Systems Magazine*, vol. 29, no. 2, pp. 28–93, 2009.

[6] T. A. Henzinger, "The theory of hybrid automata," in *IEEE Symposium on Logic in Computer Science*, 1996, pp. 278–292.

[7] J. Lygeros, K. H. Johansson, S. N. Simic *et al.*, "Dynamical properties of hybrid automata," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 2–17, 2003.

[8] L. Tavernini, "Differential automata and their discrete simulators," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 11, no. 6, pp. 665–683, 1987.

[9] J. M. Lee, *Introduction to smooth manifolds*. New York: Springer–Verlag, 2012.

[10] D. Joyce, "On manifolds with corners," in *Advances in Geometric Analysis*, ser. Advanced Lectures in Mathematics, J. Li and D. H. Phong, Eds. International Press of Boston, Inc., 2012, vol. 21, pp. 225–258.

[11] S. A. Burden, S. Revzen, and S. S. Sastry, "Model reduction near periodic orbits of hybrid dynamical systems," *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2626–2639, 2015.

[12] P. Collins, "A trajectory-space approach to hybrid systems," in *Proceedings of the 16th International Symposium on the Mathematical Theory of Networks and Systems*, 2004, p. 12.

[13] R. Goebel and A. R. Teel, "Solutions to hybrid inclusions via set and graphical convergence with stability theory applications," *Automatica*, vol. 42, no. 4, pp. 573–587, 2006.

[14] P. Ballard, "The dynamics of discrete mechanical systems with perfect unilateral constraints," *Archive for Rational Mechanics and Analysis*, vol. 154, no. 3, pp. 199–274, 2000.

[15] A. Ames, H. Zheng, R. Gregg, and S. Sastry, "Is there life after Zeno? Taking executions past the breaking (Zeno) point," in *American Control Conference, 2006*, 2006, pp. 2652–2657.

[16] Y. Or and A. Ames, "Stability and completion of Zeno equilibria in lagrangian hybrid systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1322–1336, June 2011.