Simulating Zeno Hybrid Systems Beyond Their Zeno Points

Haiyang Zheng

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-114
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-114.html

September 8, 2006

Simulating Zeno Hybrid Systems Beyond Their Zeno Points

by Haiyang Zheng

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Edward A. Lee

Research Advisor

Date

* * * * * *

Professor Shankar Sastry

Second Reader

Date

Contents

Contents ii

Acknowledgements v

Abstract vi

1 Introduction 1

1.1 Bouncing Ball . . . 3

1.2 Water Tank . . . 5

2 Completing Hybrid Systems 9

2.1 Hybrid System Completion . . . 9

2.2 Examples . . . 18

2.2.1 Example 1: Bouncing Ball. . . . 18

2.2.2 Example 2: Water Tank. . . . 21

2.2.3 Example 3: A Non-Zeno Hybrid System. . . . 23

2.2.4 Example 4: A Zeno Hybrid System. . . . 25

2.2.5 Example 5: Another Zeno Hybrid System. . . . 26

3 Approximate Simulation 29

3.1 Numerical Errors . . . 30

3.2 Computation Difficulties . . . 31

3.3 Approximating Zeno Behaviors . . . 33

3.3.1 Issue 1: Relaxing Guard Expressions. . . . 34

3.3.2 Issue 2: Reinitialization. . . . 35

4 Conclusion **37**

bibliography **38**

References 38

Acknowledgements

This work was supported in part by the Ptolemy project, the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley, which receives support from the National Science Foundation (NSF award #CCR-0225610), the State of California Micro Program, and the following companies: Agilent, DGIST, General Motors, Hewlett Packard, Infineon, Microsoft, and Toyota.

I would like to thank Aaron D. Ames. Much of the work presented in this report is collaborated with Aaron. He gives me a lot of useful and constructive comments and discussions on Zeno hybrid systems. His passion on Zeno hybrid systems greatly motivates the work presented in this report. Any errors appearing in this report are of my sole responsibility.

I would like to thank Adam Cataldo for helping me on the concepts of compact and closed sets. I would like to thank Jonathan Sprinkle for his kindness and patience on helping me about Latex questions. I would also like to thank the whole Ptolemy group for their feedback.

This report is dedicated to my parents and extended family. This report is in memory of my grandpa.

Abstract

In this report a technique is proposed to extend the simulation of a Zeno hybrid system beyond its Zeno time point. A Zeno hybrid system model is a hybrid system with an execution that takes an infinite number of discrete transitions during a finite time interval. Some classical Zeno models that incompletely describe the dynamics of the system being modeled are revisited to show that the presence of Zeno behavior indicates that the hybrid system model is incomplete. This motivates the systematic development of a method for completing hybrid system models through the introduction of new *post-Zeno* states, where the completed hybrid system transitions to these post-Zeno states at the Zeno time point.

In practice, simulating a Zeno hybrid system is challenging in that simulation effectively halts near the Zeno time point. Moreover, due to unavoidable numerical errors, it is not practical to exactly simulate a Zeno hybrid system. Therefore, a method for constructing approximations of Zeno models by leveraging the completed hybrid system model is proposed. Using these approximation, a Zeno hybrid system model can be simulated beyond its Zeno point and so the complete dynamics of the system being modeled is revealed.

Chapter 1

Introduction

The dynamics of physical systems at the macro scale level (not considering effects at the quantum level) are continuous in general. Even in a digital computer that performs computation in a discrete fashion, its fundamental computing elements (transistors) have continuous dynamics. Therefore, it is a natural choice to model the dynamics of physical systems with ordinary differential equations (ODEs) or partial differential equations (PDEs). However, modeling a physical system with only continuous dynamics may generate a stiff model, because the system dynamics might have several time scales of different magnitudes. Simulating such stiff models in general is difficult in that it takes a lot of computation time to get a reasonably accurate simulation result.

Hybrid system modeling offers one way to resolve the above problem by introducing abstractions on dynamics. In particular, slow dynamics are modeled as piecewise constants (at different phases) while fast dynamics are modeled as instantaneous changes, i.e., discretely. In this way, the remaining dynamics will have time scales of about the same magnitude and the efficiency of simulation, especially the simulation

speed, is greatly improved. However, special attention must be devoted to hybrid system models because Zeno hybrid system models may arise from the abstractions.

An execution of a Zeno hybrid system has an infinite number of discrete transitions during a finite time interval. The limit of the set of switching time points of a Zeno execution is called the *Zeno time*. The states of the model at the Zeno time point are called the *Zeno states*. Because each discrete transition takes a non-zero and finite computation time, the simulation of a Zeno hybrid system inevitably halts near the Zeno time point.

Some researchers have treated Zeno hybrid system models as over abstractions of the physical systems and tried to rule them out by developing theories to detect Zeno models [1]–[3]. However, because of the intrinsic complexity of interactions between continuous and discrete dynamics of hybrid systems, a general theory, which can give the sufficient and necessary conditions for the existence of Zeno behaviors of hybrid system models with nontrivial dynamics, is still not available (and does not appear to be anywhere on the horizon).

Some researchers have tried to extend the simulation of Zeno systems beyond the Zeno point by regularizing the original system [4], [5] or by using a sliding mode simulation algorithm [6]. The regularization method requires modification of the model structure by introducing some lower bound of the interval between consecutive discrete transitions. However, the newly introduced lower bound invalidates the abstractions and assumptions of the instantaneity of discrete transitions. Consequently, the simulation performance might suffer from the resulting stiff models. Furthermore, different behaviors after the Zeno time may be generated depending on the choices of regularizations. This may not be desirable because the physical system being modeled typically has a unique behavior. The sliding mode algorithm tends to be more

promising in simulation efficiency and uniqueness of behaviors, but it only applies to special classes of hybrid system models.

A new technique to extend simulations beyond the Zeno time point is presented in [7], where a special class of hybrid systems called *Lagrangian* hybrid systems are considered. Rather than using regularizations or a sliding mode algorithm, the dynamics of a Lagrangian hybrid system before and after the Zeno time point are derived under different constraints. In this report, we extend the results in [7] to more general hybrid system models.

Before we get into the details of the algorithm on extending simulation beyond Zeno time points, we would like to investigate some classical Zeno hybrid system models including the bouncing ball model [8] and the water tank model [4], and show that they do not completely describe the behavior of the original physical systems.

1.1 Bouncing Ball

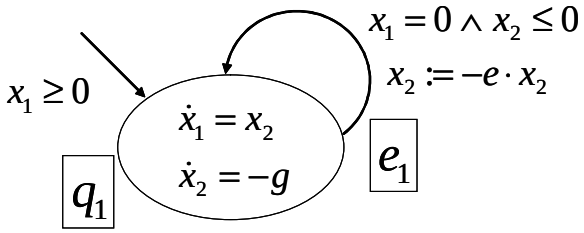

Figure 1.1. A hybrid system model of a simple bouncing ball.

Considering a ball bouncing on the ground, where bounces happen instantaneously with a restitution coefficient $e \in [0, 1]$. A hybrid system model for this system is shown in Fig. 1.1. This model has only one state q_1 associated with a second-order differential equation modeling the continuous dynamics, where the variables x_1 and x_2 represent the ball's position and velocity respectively, and $\dot{x_1} = x_2$, $\dot{x_2} = -g$.

From this one state, there is a transition e_1 that goes back to itself. The transition has a guard expression, $x_1 = 0 \wedge x_2 \leq 0$, and a reset map, $x_2 := -e \cdot x_2$.[1]

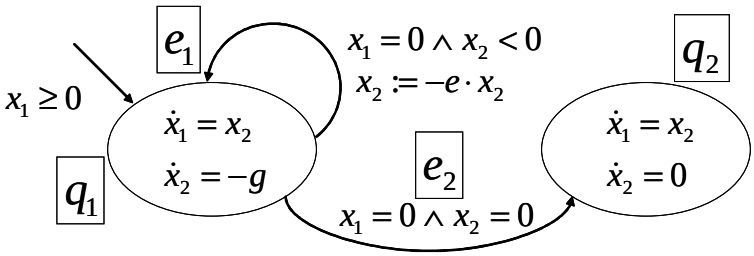

Figure 1.2. A more complete hybrid system model of the bouncing ball.

Note that the above guard expression declares that a bounce happens when the ball touches the ground and its velocity x_2 is non-positive, meaning either it is still or it is moving towards the ground. However, further analysis of the model reveals that when the following condition holds, $x_1 = 0 \wedge x_2 = 0$, meaning that the ball is at rest on the ground, the supporting force from the ground cancels out the gravity force. Therefore, the acceleration of the ball should be 0 rather than the acceleration of gravity. Under this circumstance, the ball in fact has a rather different dynamics given by $\dot{x_1} = x_2,\ \dot{x_2} = 0$.

This suggests that new dynamics might be necessary to describe the model's behavior. Consequently, the complete description of the dynamics of the bouncing ball system should include both an extra state associated with the new dynamics and a transition that drives the model into that state. One design of such a hybrid system model is shown in Fig. 1.2, where q_2 and e_2 are the new state and transition.

[1] Identity reset maps, such as $x_1 := x_1$, are not explicitly shown.

1.2 Water Tank

The second model that we will consider is the water tank system consisting of two tanks. We use x_1 and x_2 for the water levels, r_1 and r_2 for the critical water level thresholds, and v_1 and v_2 for the constant flow of water out of the tanks. There is a constant input flow of water w, which goes through a pipe and into either tank at any particular time point. We assume that $(v_1 + v_2) > w$, meaning that the sum of the output flow in both tanks is greater than the input flow. Therefore, the water levels of both tanks keep dropping. If the water level of any tank drops below its critical threshold, the input water gets delivered into that tank. The process of switching the pipe from one tank to the other takes zero time.

One hybrid system model that describes such system is shown in Fig. 1.3. This model has two states q_1 and q_2 corresponding to the different dynamics of the system when the input water flows into either of the two tanks. Transitions e_1 and e_2 specify switching conditions between states.

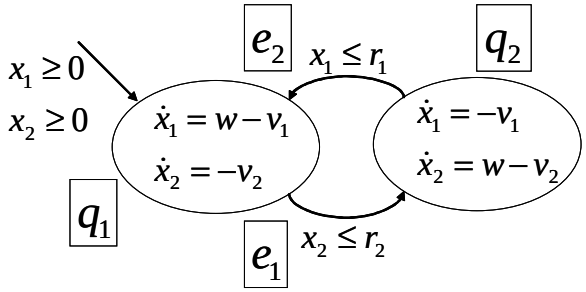

Figure 1.3. A hybrid system model of a water tank system.

Note that the guard expressions between those two states are not mutually exclusive, meaning that the guards $x_1 \leq r_1$ and $x_2 \leq r_2$ may be enabled at the same time. A trivial example will be that the two tanks have initial water levels $x_1 = r_1$ and $x_2 = r_2$. If the two tanks have initial water levels $x_1 > r_1$ and $x_2 > r_2$, then the

water levels of both tanks will drop and the water pipe will switch between the two tanks. As more and more water flows out of tanks, we will see that the frequency of the pipe switching becomes higher and higher. In the limit, when this frequency reaches infinity, both guards become enabled at the same time.

When both guards are enabled, the water tank system will have a different dynamics. Recall the assumption that the switching speed of the water pipe is infinitely fast, the pipe should inject water into both tanks at the same time. In other words, there are virtually two *identical* pipes injecting water into both tanks. Also note that the input water flow is a constant and the pipe cannot hold water, therefore one possible scenario will be that each tank gets *half* of the input water. Therefore, at this time point, the whole system will have a rather different dynamics given by

$$\dot{x_1} = w/2 - v_1, \quad \dot{x_2} = w/2 - v_2. \tag{1.1}$$

We introduce a new state associated with the above dynamics and complete the transitions going from the existing states to the newly added state. The new design of the complete hybrid system model for the water tank system is shown in Fig. 1.4, where q_3 is the new state, e_3 and e_4 are the newly added transitions. Note that for simplicity we allow the water levels to have negative values. Otherwise, we will need some other discrete states to show that once a tank is empty, it is always empty.

The hybrid system model in Fig. 1.4 is similar to the *temporal* regularization results proposed in [4]. One of the key differences is that the temporal regularization solution requires the process of switching pipe to take some positive time ϵ. The amount of this ϵ affects the resulting behaviors. In fact, when ϵ goes to 0, the temporal regularization result is the same as what we have derived in (1.1).

In the next chapter, we will propose a systematic way to complete the specification of hybrid system models. In particular, we will discuss how to introduce new states, to modify the existing transitions, and to construct new transitions to these new

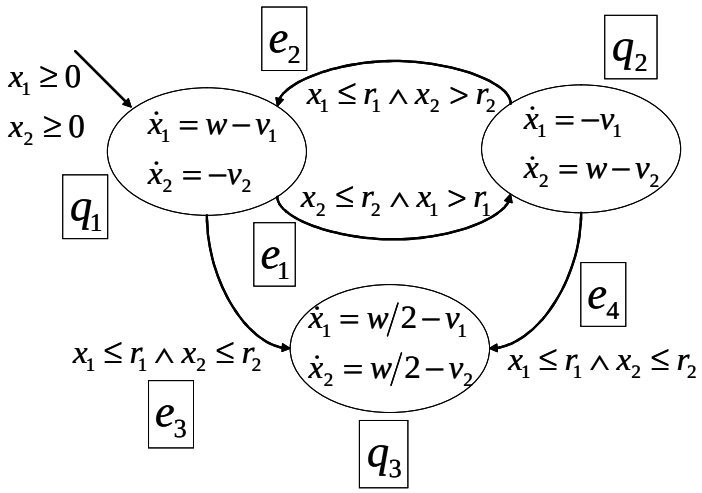

Figure 1.4. A more complete hybrid system model of the water tank system.

states for model behaviors before and after potential Zeno time points. In Chapter 3, we will develop a feasible simulation algorithm to approximate the exact behaviors of Zeno hybrid system models. Conclusions will be given in Chapter 4.

Chapter 2

Completing Hybrid Systems

The purpose of this chapter is to introduce an algorithm for completing hybrid system models with the goal of carrying executions past the Zeno point. This algorithm can be thought of as a combination of the currently known conditions for the existence (or nonexistence) of Zeno behavior in hybrid systems. Of course, the characterization of Zeno behavior in the literature is by no means complete, so we cannot claim that the procedure outlined here is the only way to complete a hybrid system, nor that the resulting hybrid system is the canonical completed hybrid system. We only claim that, given the current understanding of Zeno behavior, this method provides a reasonably satisfying method for completing hybrid systems. We dedicate the latter half of this chapter to examples, where we carry out the completion process.

2.1 Hybrid System Completion

Define a *hybrid system* as a tuple,

$$\mathscr{H} = (\Gamma, D, G, R, F),$$

where

- $\Gamma = (Q, E)$ is a finite directed graph, where Q represents the set of discrete states and E represents the set of edges connecting these states. There are two maps $\mathfrak{s} : E \to Q$ and $\mathfrak{t} : E \to Q$, which are the source and target maps respectively. That is $\mathfrak{s}(e)$ is the source of the edge e and $\mathfrak{t}(e)$ is its target.

- $D = \{D_q \subseteq \mathbb{R}^n_\infty \mid q \in Q\}$ is a set of *domains*, one for each state $q \in Q$. While the hybrid system is in state q, the dynamics of the hybrid system is a trajectory in D_q. Note that $\mathbb{R}^n_\infty = \mathbb{R}^n \cup \{\infty\}$ and $\mathbb{R}^n_\infty$ is compact.

- $G = \{G_e \subseteq D_{\mathfrak{s}(e)} \mid e \in E\}$ is a set of *guards*, where G_e is a set associated with the edge e and determines the switching behavior of the hybrid system at state $\mathfrak{s}(e)$. When the trajectory intersects with the guard set G_e, a transition is triggered and the discrete state of the hybrid system changes to $\mathfrak{t}(e)$. $G_q = \bigcup_{\mathfrak{s}(e)=q} G_e$ is the union of the guards associated with the outgoing edges from the same state q. We assume that G_q is closed, i.e., that every Cauchy sequence converges to an element in G_q.

- $R = \{R_e : G_e \to D_{\mathfrak{t}(e)} \mid e \in E\}$ is a set of *reset maps*. We write the image of R_e as $R_e(G_e) \subseteq D_{\mathfrak{t}(e)}$. These reset maps specify the initial continuous states of trajectories in the target discrete states.

- $F = \{f_q : D_q \to \mathbb{R}^n_\infty \mid q \in Q\}$ is a set of *vector fields*, which specify the dynamics of the hybrid system when it is in a discrete state q. We assume f_q is Lipschitz when restricted to D_q.

In this report, we will not explicitly define hybrid system behavior and Zeno behavior, as these definitions are well-known and can be found in a number of references (cf. [1], [2], [4], [9]).

The goal of this section is to complete a hybrid system $\mathscr{H}$, i.e., we want to form a new hybrid system $\overline{\mathscr{H}}$ in which executions are carried beyond the Zeno point. We

begin by constructing this system theoretically and then discuss how to implement it practically. The theoretical completion of a hybrid system is carried out using the following process:

1. Augment the graph Γ of $\mathscr{H}$, based on the existence of *higher order cycles*, to include *post-Zeno* states, and edges to these post-Zeno states.

2. Specify the domains of the post-Zeno states.

3. Specify the guards on the edges to the post-Zeno states.

4. Specify the reset maps on the edges to the post-Zeno states.

5. Specify the vector fields on the post-Zeno states, based on the vector fields on the pre-Zeno states.

6. Check the existence of higher order cycles again on the completed hybrid systems. If there is any, go the step 1 and repeat the completing process. Otherwise, the process terminates.

Before carrying out this process, it is necessary to introduce the notion of *higher order cycles* in Γ. We call a finite string consisting of alternating states and edges in Γ a *finite path*,

$$q_1 \xrightarrow{e_1} q_2 \xrightarrow{e_2} q_3 \xrightarrow{e_3} \cdots \xrightarrow{e_{k-1}} q_k,$$

with $e_i \in E$ and $q_i \in Q$, s.t., $s(e_i) = q_i$ and $t(e_i) = q_{i+1}$. We denote such a path by $\langle q_1; e_1, e_2, \ldots, e_{k-1}; q_k \rangle$.

For simplicity, we only consider paths with distinct edges. We could have considered paths with repeated edges, but that will result in an unbounded number of paths, each of which is arbitrarily long. This makes the problem intractable. The number and length of paths with distinct edges are finite. In the worst case scenario,

suppose there are N edges in the graph, the number of paths is $\sum_{m=1}^{N}(P_m^N)$, where P_m^N gives the number of permutations of choosing m edges from a set of N edges.

Although we only consider paths with distinct edges, we do not require a path to contain distinct states. In particular, if the starting state is the same as the ending state, such as $\langle q_1; e_1, e_2, \ldots, e_{k-1}; q_1\rangle$, we call such a path a *finite cyclic path*. The set of all finite cyclic paths is called the *higher order cycles* in Γ and denoted by C. Formally,

$$C = \{\langle q; e_1, e_2, \ldots, e_{k-1}; q\rangle \mid \forall i, j,\ i \neq j \Rightarrow e_i \neq e_j,\ e_i, e_j \in E,\ q \in Q\}. \tag{2.1}$$

To ease future discussion, we define $c_0 = \langle q; e_1, e_2, \ldots, e_{k-1}; q\rangle \in C$. We will use this example cyclic path through the rest of this section.

We also define two operators, π_Q and π_E, on a cyclic path $c \in C$, where $\pi_Q(c)$ gives the starting and ending state of the path and $\pi_E(c)$ gives the first edge appearing in the path. When applied to the path c_0, $\pi_Q(c_0) = q$ and $\pi_E(c_0) = e_1$.

Consider any two edges $e_i, e_j \in E$ appearing a cyclic path such as c_0, where e_j immediately follows e_i, meaning $\mathfrak{s}(e_j) = \mathfrak{t}(e_i)$. We define a *restricted reset map* for the first edge e_i, $W_{e_i} : G_{e_i} \to G_{e_j}$ by

$$W_{e_i}(G_{e_i}) = R_{e_i}(G_{e_i}) \cap G_{e_j},$$

where G_{e_i} and G_{e_j} are guard sets associated with edges e_i and e_j respectively. The word "restricted" means that the image of the function W_{e_i} is constrained by the guard set of the destination state G_{e_j}.

For the last edge e_{k-1} appearing in the path c_0, we define the function $W_{e_{k-1}}$ by

$$W_{e_{k-1}}(G_{e_{k-1}}) = R_{e_{k-1}}(G_{e_{k-1}}) \cap G_{e_1},$$

where G_{e_1} is the guard set associated with the first edge e_1 appearing in the path c_0.

Then for a cyclic path $c \in C$, we define a function $R_c : G_{\pi_E(c)} \to G_{\pi_E(c)}$, where $\pi_E(c)$ is the first edge appearing in the path c. The function R_c is the composition of the reset maps along the path c, called the *reset map along path c*. Choosing the path c_0 as an example, we define the function $R_{c_0} : G_{e_1} \to G_{e_1}$ by

$$R_{c_0} = R_{\langle q;e_1,e_2,\ldots,e_{k-1};q\rangle} = W_{e_{k-1}} \circ W_{e_{k-2}} \circ \cdots \circ W_{e_2} \circ W_{e_1}. \tag{2.2}$$

We write the image of R_c as $R_c(G_{\pi_E(c)})$. From the definition of R_c, we know $R_c(G_{\pi_E(c)})$ must be a subset of $G_{\pi_E(c)}$ because it is resulted from intersection of sets. If we apply R_c again to $R_c(G_{\pi_E(c)})$, let $R_c^2(G_{\pi_E(c)})$ denote $R_c(R_c(G_{\pi_E(c)}))$, we have $R_c^2(G_{\pi_E(c)}) \subseteq R_c(G_{\pi_E(c)})$. Keeping applying the function R_c will result in a sequence of sets,

$$G_{\pi_E(c)} \supseteq R_c(G_{\pi_E(c)}) \supseteq R_c^2(G_{\pi_E(c)}) \supseteq \cdots \supseteq R_c^k(G_{\pi_E(c)}) \supseteq \cdots .$$

A question naturally arises: does the sequence always converge?

It is well known (c.f. [10]) that for any set X, the ordered set $(\mathscr{P}(X); \subseteq)$ with a set inclusion order is a *complete lattice*, where the *least upper bound (LUB)* is given by the union of subsets and the *greatest lower bound (GLB)* by the intersection of subsets. Note that the ordered set $(\mathscr{P}(X); \supseteq)$ with a *reverse* set inclusion order is also a complete lattice, where the LUB is given by the intersection and the GLB by the union of subsets. By this definition, $(\mathscr{P}(\mathbb{R}^n_\infty); \supseteq)$ is a complete lattice.

A complete lattice is a *CPO* (an abbreviation for *complete partially ordered set*). Therefore, $(\mathscr{P}(\mathbb{R}^n_\infty); \supseteq)$ is a CPO. A CPO has a bottom element, denoted as $\bot$. For $(\mathscr{P}(\mathbb{R}^n_\infty); \supseteq)$, the bottom element is $\mathbb{R}^n_\infty$.

Every chain in a CPO converges, therefore the sequence of sets,

$$G_{\pi_E(c)} \supseteq R_c(G_{\pi_E(c)}) \supseteq R_c^2(G_{\pi_E(c)}) \supseteq \cdots \supseteq R_c^k(G_{\pi_E(c)}) \supseteq \cdots ,$$

which forms a chain, always converges.

Now our question is whether we can constructively find the set that this sequence converges to. We will answer this question by applying a least fixed point theorem in [10] next.

We first define a function for a cyclic path c, $\mathbf{F}_c : \mathscr{P}(\mathbb{R}^n_\infty) \to \mathscr{P}(\mathbb{R}^n_\infty)$, by

$$\mathbf{F}_c(S) = R_c(S \cap G_{\pi_E(c)}), \text{ where } S \in \mathscr{P}(\mathbb{R}^n_\infty). \tag{2.3}$$

Theorem 1. *Function $\mathbf{F}_c$ is monotonic.*

Proof. Recall that $\mathbf{F}_c$ is monotonic if

$$\forall\, S_1, S_2 \in \mathscr{P}(\mathbb{R}^n_\infty), S_1 \supseteq S_2 \Rightarrow \mathbf{F}_c(S_1) \supseteq \mathbf{F}_c(S_2).$$

Let $e = \pi_E(c)$. Suppose $S_1 \supseteq S_2$, then $G_e \cap S_1 \supseteq G_e \cap S_2$, and $R_e(G_e \cap S_1) \supseteq R_e(G_e \cap S_2)$. We have $W_e(G_e \cap S_1) \supseteq W_e(G_e \cap S_2)$, and therefore $R_c(G_e \cap S_1) \supseteq R_c(G_e \cap S_2)$, which gives $\mathbf{F}_c(S_1) \supseteq \mathbf{F}_c(S_2)$. □

In fact, a more general conclusion is that function $\mathbf{F}_c$ is a monotonic function as long as function R_c is a total function.

Theorem 2. *Function $\mathbf{F}_c$ is continuous.*

Proof. Recall that $\mathbf{F}_c$ is continuous if for any chain $L \subseteq \mathscr{P}(\mathbb{R}^n_\infty)$, where $L = \{S_1, S_2, S_3, \cdots\}$ and $S_1 \supseteq S_2 \supseteq S_3 \supseteq \cdots$,

$$\mathbf{F}_c(\wedge L) = \wedge \hat{\mathbf{F}}_c(L) = \wedge\{\mathbf{F}_c(S_i) \mid S_i \in L\}.$$

Note that $\forall\, S_i, S_i \supseteq \wedge L$. Because $\mathbf{F}_c$ is monotonic, $\mathbf{F}_c(S_i) \supseteq \mathbf{F}_c(\wedge L)$. Therefore,

$$\mathbf{F}_c(\wedge L) \subseteq \wedge \hat{\mathbf{F}}_c(L).$$

Now we want to show

$$\wedge \hat{\mathbf{F}}_c(L) \subseteq \mathbf{F}_c(\wedge L).$$

For any $x \in \wedge\hat{\mathbf{F}}_c(L)$, there must exist $y \in S_i \cap G_{\pi_E(c)}$ for all S_i, s.t., $x = R_c(y)$. Also note that $\mathbf{F}_c(\wedge L) = R_c(\bigcap_i S_i \cap G_{\pi_E(c)})$ and $y \in \bigcap_i S_i \cap G_{\pi_E(c)}$, therefore $x \in \mathbf{F}_c(\wedge L)$. In other words, $\wedge\hat{\mathbf{F}}_c(L) \subseteq \mathbf{F}_c(\wedge L)$.

In the end,

$$\wedge\hat{\mathbf{F}}_c(L) \subseteq \mathbf{F}_c(\wedge L) \text{ and } \mathbf{F}_c(\wedge L) \subseteq \wedge\hat{\mathbf{F}}_c(L) \Rightarrow \mathbf{F}_c(\wedge L) = \wedge\hat{\mathbf{F}}_c(L).$$

□

Alternatively, we can prove theorem (2) according to 8.8(2) in [10] (page 178). In particular, because $(\mathscr{P}(X); \supseteq)$ satisfies the *ascending chain condition* (ACC) defined in 2.37(v) in [10] (page 51), and function $\mathbf{F}_c$ is monotonic according to theorem (1), $\mathbf{F}_c$ is a continuous function.

Theorem 3 (least fixed point theorem, c.f. [10])**.** *Every continuous function $f : X \to X$ over a CPO X has a least fixed point x, such that $f(x) = x$. Further more, x is given by $\bigsqcup_{n\geq 0} f^n(\bot)$, where $\bot$ is the bottom element of the CPO X.*

The proof of this theorem is omitted and can be found in [10]. The important result is that the sequence of sets,

$$G_{\pi_E(c)} \supseteq R_c(G_{\pi_E(c)}) \supseteq R_c^2(G_{\pi_E(c)}) \supseteq \cdots \supseteq R_c^k(G_{\pi_E(c)}) \supseteq \cdots$$

converges to a set, which is a least fixed point of $\mathbf{F}_c$.

Theorem (3) also gives us a constructive procedure to find such a fixed point. That is, start by evaluating $\mathbf{F}_c$ with $\mathbb{R}^n_\infty$, and then keep applying $\mathbf{F}_c$ on the evaluated results until they converge to a fixed point.

However we cannot guarantee that the fixed point can be computed with the above procedure in a *finite* number of steps of computation. In fact, example 2.2.4 in the next section will show that finding the fixed point may take an infinite number

of computation steps. In this case, our algorithm breaks down and this is one of the limitations.

We denote such a fixed point as Z_c, called *Zeno set* of cyclic path c, where

$$Z_c = \mathbf{F}_c(Z_c). \tag{2.4}$$

Equation (2.4) states that if a trajectory intersects with the guard set $G_{\pi_E(c)}$ at an element $z \in Z_c \subseteq G_{\pi_E(c)}$, then after a series of reset maps, R_c, the initial continuous state of the new trajectory z' is again in Zeno set Z_c, which is a subset of $G_{\pi_E(c)}$. Note that the new initial state z' does not have to be the same as the starting state z (c.f. [1]). However, the new state z' will nevertheless trigger the first transition appearing first in the path to be taken anyway. With the constraint that transitions happen instantaneously, there will be an infinite number of transitions happening at the same time point. Therefore, the existence of a nonempty Zeno set Z_c indicates the possible existence of Zeno equilibria (cf. [1]). This motivates the construction of the completed hybrid system based on a subset of cyclic paths, $C' = \{c \in C \mid Z_c \neq \emptyset\}$.

For a hybrid system $\mathscr{H}$, define the corresponding *completed hybrid system* $\overline{\mathscr{H}}$ by

$$\overline{\mathscr{H}} = (\overline{\Gamma}, \overline{D}, \overline{G}, \overline{R}, \overline{F}),$$

where

1. $\overline{\Gamma} = (\overline{Q}, \overline{E})$, where $\overline{\Gamma}$ has more discrete states and edges than Γ. The set of extra states is $Q' = \overline{Q} \setminus Q$, where Q' is called the set of *post-Zeno states*. The set of extra edges is $E' = \overline{E} \setminus E$. We pick the extra states and edges to be in bijective correspondence with Q', i.e., there exist bijections $g : Q' \to C'$ and $h : E' \to C'$. Consequently, $\forall c \in C'$, there always exist a unique $q \in Q'$ and a unique $e \in E'$.

We define the source and target maps, $\overline{\mathfrak{s}} : \overline{E} \to \overline{Q}$ and $\overline{\mathfrak{t}} : \overline{E} \to \overline{Q}$ for $e \in \overline{E}$ by

$$\overline{\mathfrak{s}}(e) = \begin{cases} \mathfrak{s}(e) & \text{if} \quad e \in E \\ \pi_Q(h(e)) & \text{if} \quad e \in E' \end{cases}, \quad \text{and} \quad \overline{\mathfrak{t}}(e) = \begin{cases} \mathfrak{t}(e) & \text{if} \quad e \in E \\ g^{-1}(h(e)) & \text{if} \quad e \in E' \end{cases}.$$

Intuitively, for each cyclic path $c \in C'$ found in Γ, we can find a new discrete state $q = g^{-1}(c) \in Q'$ and a new edge $e = h^{-1}(c) \in E'$ that goes from $\pi_Q(c)$ to q in $\overline{\Gamma}$.

The new discrete state q, which corresponds to the cyclic path $c = g(q)$, copies all edges of the original state $\pi_Q(c)$ except the first edge appearing in the path, $\pi_E(c)$. In fact, we can think that the newly introduced edge $e = h^{-1}(c)$ replaces the edge $\pi_E(c)$.

2. Define $\overline{D} = D \cup D'$, where D' is the set of domains of post-Zeno states, defined as $D' = \{D'_q \subseteq \mathbb{R}^n_\infty \mid q \in Q'\}$. For each $c \in C'$, D'_q is defined by

$$D'_q = Z_c, \quad \text{where} \quad q = g^{-1}(c) \in Q'. \tag{2.5}$$

Note that D'_q is not only the domain for post-Zeno state q but also the guard set that triggers the transition from the pre-Zeno state $\pi_Q(c)$ to the post-Zeno state q.

3. In order to define $\overline{G}$, we first modify the guard G_e in G by subtracting Z_c from G_e, where $c \in C'$ with $\pi_Q(c) = \overline{\mathfrak{s}}(e)$. Define, for all $e \in E$,

$$\widetilde{G_e} = G_e \setminus \bigcup_{c \in C' \text{ s.t. } \pi_Q(c) = \overline{\mathfrak{s}}(e)} Z_c, \tag{2.6}$$

and define, for all $e \in E'$,

$$G'_e = D'_q, \quad \text{where} \quad q = \overline{\mathfrak{t}}(e). \tag{2.7}$$

Then the complete definition of $\overline{G}$ is $\overline{G} = \{\widetilde{G_e} \mid e \in E\} \cup \{G'_e \mid e \in E'\}$.

4. $\overline{R} = \{R_e : \widetilde{G_e} \to D_{\bar{\mathfrak{t}}(e)} \mid e \in E\} \cup \{R'_e : G'_e \to D_{\bar{\mathfrak{t}}(e)} \mid e \in E'\}$, where the reset map R'_e is the identity map.

5. $\overline{F} = F \cup \{f'_q : D'_q \to \mathbb{R}^n_\infty \mid q \in Q'\}$, where f'_q is the vector field on D'_q. This vector field may be application-dependent, but in some circumstances, it can be obtained from the vector field $f_{q'}$ on $D_{q'}$, where $q' = \pi_Q(g(q)) \in Q$.

6. Check the existence of higher order cycles again on the completed hybrid systems. If there is any, go the step 1 and repeat the completing process. Otherwise, the process terminates. This step is necessary for completing hybrid systems with a state having multiple outgoing edges pointing back to itself.

Upon inspection of the definition of the completed hybrid system, it is evident that we have explicitly given a method for computing every part of this system except for the vector fields on the post-Zeno states. We do not claim to have an explicit method for generally computing f'_q, because this would depend on the constraints imposed by D'_q which we do not assume are of any specific form. However, in some special cases, it is possible to find such a vector field. In the next subsection, we will demonstrate how to carry out the process of completing hybrid systems by revisting the examples discussed in Sect. 1.

2.2 Examples

2.2.1 Example 1: Bouncing Ball.

We first revisit the bouncing ball example shown in Fig. 1.1. Write this example hybrid system as a tuple, $\mathscr{H} = ((Q, E), D, G, R, F)$. We have the discrete state set $Q = \{q_1\}$, the edge set $E = \{e_1\}$, the set of guards $G = \{G_{e_1}\}$, where $G_{e_1} =$

$\{(x_1, x_2) \in \mathbb{R}^2_\infty \mid x_1 = 0 \wedge x_2 \leq 0\}$, and the set of the reset maps $R = \{R_{e_1}\}$, where R_{e_1} is defined by $R_{e_1}(x_1, x_2) = (x_1, -e \cdot x_2)$, $\forall (x_1, x_2) \in G_{e_1}$.

There is only one element, $c = \langle q_1; e_1; q_1 \rangle$, in the set C of cyclic paths. For path c, the composition of reset maps along c is $R_c = W_{e_1}$ and $\pi_E(c) = e_1$. Evaluating equation (2.4) with Z_c as the guard G_{e_1}, we get

$$\begin{aligned}
Z'_c &= \mathbf{F}_c(\mathbb{R}^n_\infty) \\
&= R_c(\mathbb{R}^n_\infty \cap G_{e_1}) \\
&= W_{e_1}(G_{e_1}) \\
&= R_{e_1}(G_{e_1}) \cap G_{e_1} \\
&= \{(x_1, x_2) \mid x_1 = 0 \wedge x_2 \geq 0\} \cap \{(x_1, x_2) \mid x_1 = 0 \wedge x_2 \leq 0\} \\
&= \{(x_1, x_2) \mid x_1 = 0 \wedge x_2 = 0\} \\
&= \{(0, 0)\}.
\end{aligned}$$

Keep evaluating equation (2.4) with Z'_c, and we get

$$\begin{aligned}
Z''_c &= \mathbf{F}_c(Z'_c) \\
&= R_c(Z'_c \cap G_{e_1}) \\
&= W_{e_1}(Z'_c) \\
&= R_{e_1}(Z'_c) \cap Z'_c \\
&= \{(0, 0)\} \cap \{(0, 0)\} \\
&= \{(0, 0)\}.
\end{aligned}$$

Since Z''_c is the same as Z'_c, we have got a fixed point of $\mathbf{F}_c$, denoted as $Z_c = \{(0, 0)\}$. Since Z_c is nonempty, we introduce a new state q_2 and a new edge e_2 such that $\overline{Q} = \{q_1, q_2\}$ and $\overline{E} = \{e_1, e_2\}$. The source and target maps are

$$\overline{\mathfrak{s}}(e) = \quad q_1 \quad , \quad \forall e \in \overline{E} \ , \qquad \text{and} \qquad \overline{\mathfrak{t}}(e) = \begin{cases} q_1 & \text{if} \quad e = e_1 \\ q_2 & \text{if} \quad e = e_2 \end{cases} .$$

Because there is only one outgoing edge e_1 from the state q_1 and this edge appears in the cyclic path c, the new state q_2 copies no edge from q_1.

The domain for discrete state q_2 is $D'_{q_2} = Z_c$. Then $\overline{D} = D \cup \{D'_{q_2}\}$. Since the set D'_{q_2} only contains one element, the dynamics (vector fields) of the hybrid system is trivial, where $\dot{x_1}(t) = 0,\ \dot{x_2}(t) = 0$. This simply means that the ball cannot move at all, which is exactly the same as what we got in the introduction.

We must point out that the domain for a post-Zeno state may contain more than one element. In this case, the dynamics in general cannot be computed without a model designer's expertise. However, in some special cases such as mechanical systems, the vector fields describe the equations of motion for these systems. If in addition, the guards are derived from unilateral constraints on the configuration space, then the vector fields on the post-Zeno states can be obtained from the vector fields on the pre-Zeno states via holonomic constraints. In fact, the vector fields on the post-Zeno state of the above example can be obtained from a *hybrid Lagrangian* [7]. A detailed explanation of the process for computing vector fields and more examples can be found in [7].

Note that D'_{q_2} is also the guard set of e_2 that specifies the switching condition from q_1 to q_2, meaning $G'_{e_2} = \{(x_1, x_2) \in \mathbb{R}^2_\infty \mid x_1 = 0 \wedge x_2 = 0\}$. Following (2.6), we get a modified $\widetilde{G_{e_1}} = \{(x_1, x_2) \in \mathbb{R}^2_\infty \mid x_1 = 0 \wedge x_2 < 0\}$. The set of these two guard sets gives $\overline{G} = \{\widetilde{G_{e_1}}, G'_{e_2}\}$.

Finally, $\overline{R} = \{R_{e_1}, R'_{e_2}\}$, where R'_{e_2} is just the identity map.

Now we need to check the completed hybrid system again for possible non-empty Zeno sets. There is still only one cyclic path c. However by repeating the above procedure, we get $Z_c = \emptyset$. The detailed derivation process is the same as what described the above and is omitted here. Therefore the completing process is finished.

In summary we get the completed hybrid system $\overline{\mathscr{H}} = ((\overline{Q}, \overline{E}), \overline{D}, \overline{G}, \overline{R}, \overline{F})$, which is the same as the model shown in Fig. 1.2.

2.2.2 Example 2: Water Tank.

Now let us revisit the water tank example shown in Fig. 1.3. Write this example hybrid system as a tuple, $\mathscr{H} = ((Q, E), D, G, R, F)$. We have the discrete state set $Q = \{q_1, q_2\}$, the edge set $E = \{e_1, e_2\}$, the set of guards $G = \{G_{e_1}, G_{e_2}\}$, where $G_{e_1} = \{(x_1, x_2) \in \mathbb{R}^2_\infty \mid x_2 \leq r_2\}$ and $G_{e_2} = \{(x_1, x_2) \in \mathbb{R}^2_\infty \mid x_1 \leq r_1\}$, and the set of the reset maps $R = \{R_{e_1}, R_{e_2}\}$, where both reset maps are identity maps.

There are two elements, $c_1 = \langle q_1; e_1, e_2; q_1\rangle$ and $c_2 = \langle q_2; e_2, e_1; q_2\rangle$, in the set C that contains cyclic paths. For path c_1, the composition of reset maps along c_1 is $R_{c_1} = W_{e_2} \circ W_{e_1}$ and $\pi_E(c_1) = e_1$.

We first compute the fixed point Z_{c_1} for $\mathbf{F}_{c_1}$ by evaluating (2.4) with Z_{c_1} as the guard G_{e_1}, we get

$$\begin{aligned}
Z'_{c_1} &= \mathbf{F}_{c_1}(\mathbb{R}^n_\infty) \\
&= R_{c_1}(\mathbb{R}^n_\infty \cap G_{e_1}) \\
&= (W_{e_2} \circ W_{e_1})(G_{e_1}) \\
&= W_{e_2}(W_{e_1}(G_{e_1})) \\
&= W_{e_2}(R_{e_1}(G_{e_1}) \cap G_{e_2}) \\
&= W_{e_2}(G_{e_1} \cap G_{e_2}) \\
&= R_{e_2}(G_{e_1} \cap G_{e_2}) \cap G_{e_1} \\
&= G_{e_1} \cap G_{e_2} \cap G_{e_1} \\
&= G_{e_1} \cap G_{e_2} \\
&= \{(x_1, x_2) \mid x_2 \leq r_2\} \cap \{(x_1, x_2) \mid x_1 \leq r_1\}
\end{aligned}$$

$$= \{(x_1, x_2) \mid x_1 \leq r_1 \wedge x_2 \leq r_2\}.$$

Keep evaluating equation (2.4) with Z'_{c_1}, we get

$$\begin{aligned} Z''_{c_1} &= \mathbf{F}_{c_1}(Z'_{c_1}) \\ &= R_{c_1}(Z'_{c_1} \cap G_{e_1}) \\ &= (W_{e_2} \circ W_{e_1})(Z'_c) \\ &= \cdots \quad \text{(steps omitted)} \\ &= \{(x_1, x_2) \mid x_1 \leq r_1 \wedge x_2 \leq r_2\}. \end{aligned} \tag{2.8}$$

Since Z''_{c_1} is the same as Z'_{c_1}, we have got a fixed point of $\mathbf{F}_{c_1}$, denoted as $Z_{c_1} = \{(x_1, x_2) \mid x_1 \leq r_1 \wedge x_2 \leq r_2\}$. Similarly, for path c_2, we get $Z_{c_2} = \{(x_1, x_2) \mid x_1 \leq r_1 \wedge x_2 \leq r_2\}$, which is the same as Z_{c_1}.

Since both Z_{c_1} and Z_{c_2} are nonempty, we introduce two new states q_3 and q_4 and two new edges e_3 and e_4 such that $\overline{Q} = \{q_1, q_2, q_3, q_4\}$ and $\overline{E} = \{e_1, e_2, e_3, e_4\}$. The source and target maps are

$$\overline{\mathfrak{s}}(e) = \begin{cases} q_1 & \text{if} \quad e = e_1 \vee e = e_3 \\ q_2 & \text{if} \quad e = e_2 \vee e = e_4 \end{cases}, \quad \text{and} \quad \overline{\mathfrak{t}}(e) = \begin{cases} q_2 & \text{if} \quad e = e_1 \\ q_1 & \text{if} \quad e = e_2 \\ q_3 & \text{if} \quad e = e_3 \\ q_4 & \text{if} \quad e = e_4 \end{cases}.$$

Because there is only one outgoing edge e_1 from the state q_1 and this edge appears in the cyclic path c, the new state q_3 copies no edge from q_1. Similarly, the new state q_4 does not copy edges from q_2.

The domain for discrete state q_3 is $D'_{q_3} = Z_{c_1}$, and the domain for discrete state q_4 is $D'_{q_4} = Z_{c_2}$. Then $\overline{D} = D \cup \{D'_{q_3}, D'_{q_4}\}$.

As we pointed out earlier in the previous example, in order to derive the dynamics for post-Zeno states, a careful analysis has to be performed by model designers, and

the resulting dynamics may not be unique. For example, one might think that 3/4 of the input flow goes into the first tank and the rest goes into the second tank. This dynamics is different from what we had in the introduction chapter. We do not (in fact, we cannot) determine which result is better.

Note that D'_{q_3} is also the guard set of e_3 that specifies the switching condition from q_1 to q_3, meaning $G'_{e_3} = \{(x_1, x_2) \in \mathbb{R}^2_\infty \mid x_1 \leq r_1 \wedge x_2 \leq r_2\}$. Following (2.6), we get a modified $\widetilde{G_{e_1}} = \{(x_1, x_2) \in \mathbb{R}^2_\infty \mid x_1 \leq r_1 \wedge x_2 > r_2\}$. Similarly, we get $G'_{e_4} = \{(x_1, x_2) \in \mathbb{R}^2_\infty \mid x_1 \leq r_1 \wedge x_2 \leq r_2\}$, and a modified $\widetilde{G_{e_2}} = \{(x_1, x_2) \in \mathbb{R}^2_\infty \mid x_2 \leq r_2 \wedge x_1 > r_1\}$. The set of these two guard sets gives $\overline{G} = \{\widetilde{G_{e_1}}, \widetilde{G_{e_2}}, G'_{e_3}, G'_{e_4}\}$.

Finally, $\overline{R} = \{R_{e_1}, R_{e_2}, R'_{e_3}, R'_{e_4}\}$, where all reset maps are identity maps.

Now we need to check the completed hybrid system again for possible non-empty Zeno sets. There are still only two cyclic paths c_1 and c_2. However by repeating the above procedure, we get $Z_{c_1} = Z_{c_2} = \emptyset$. The detailed derivation process is the same as described the above and is omitted here. Therefore the completing process is finished.

In summary we get the completed hybrid system $\overline{\mathscr{H}} = ((\overline{Q}, \overline{E}), \overline{D}, \overline{G}, \overline{R}, \overline{F})$, which is slightly different from the model shown in Fig. 1.4 in that $\overline{\mathscr{H}}$ contains 4 discrete states. However, if we choose the same dynamics such as (1.1) for discrete states q_3 and q_4, then q_3 and q_4 are the same. Thus we get a model with the same dynamics as that of the model in Fig. 1.4.

2.2.3 Example 3: A Non-Zeno Hybrid System.

Now let us consider a hybrid system simpler than the bouncing ball example shown in Fig. 1.1. Similar to the bouncing ball example, this hybrid system has a single state, $Q = \{q_1\}$, and a single edge, $E = \{e_1\}$. The set of guards is $G = \{G_{e_1}\}$,

where $G_{e_1} = \{x \in R \mid 0 \leq x \leq 10\}$, and the set of the reset maps is $R = \{R_{e_1}\}$, where R_{e_1} is defined by $R_{e_1}(x) = x + 1$, $\forall x \in G_{e_1}$.

There is only one element, $c = \langle q_1; e_1; q_1 \rangle$, in the set C of cyclic paths. For path c, the composition of reset maps along c is $R_c = W_{e_1}$ and $\pi_E(c) = e_1$. Evaluating equation (2.4) with Z_c as the guard G_{e_1}, we get

$$\begin{aligned} Z'_c &= \mathbf{F}_c(\mathbb{R}^n_\infty) \\ &= R_c(\mathbb{R}^n_\infty \cap G_{e_1}) \\ &= W_{e_1}(G_{e_1}) \\ &= R_{e_1}(G_{e_1}) \cap G_{e_1} \\ &= \{x \in R \mid 1 \leq x \leq 11\} \cap \{x \in R \mid 0 \leq x \leq 10\} \\ &= \{x \in R \mid 1 \leq x \leq 10\}. \end{aligned}$$

Keep evaluating equation (2.4) with Z'_c, and we get

$$\begin{aligned} Z''_c &= \mathbf{F}_c(Z'_c) \\ &= R_c(Z'_c \cap G_{e_1}) \\ &= W_{e_1}(Z'_c) \\ &= R_{e_1}(Z'_c) \cap Z'_c \\ &= \{x \in R \mid 2 \leq x \leq 10\}. \end{aligned}$$

Keep evaluating equation (2.4) with the new results for 10 times, and we will get

$$\begin{aligned} Z^{11}_c &= \mathbf{F}_c(Z^{10}_c) \\ &= R_c(Z^{10}_c \cap G_{e_1}) \\ &= W_{e_1}(Z^{10}_c) \\ &= R_{e_1}(Z^{10}_c) \cap Z^{10}_c \\ &= R_{e_1}(\{x \in R \mid x = 10\}) \cap \{x \in R \mid x = 10\} \end{aligned}$$

$$
\begin{aligned}
&= \{x \in R \mid x = 11\} \cap \{x \in R \mid x = 10\} \\
&= \emptyset,
\end{aligned}
$$

where Z_c^n indicates the n-th result from evaluating equation (2.4).

The result set Z_c^{11} is an empty set, any set intersecting with an empty set will result in an empty set. The empty set is the least fixed point of $\mathbf{F}_c$. Because this fixed point is empty, there is no Zeno point. Therefore, we conclude this model is non-Zeno and it is already complete.

2.2.4 Example 4: A Zeno Hybrid System.

In this example, we use the same model in Example 2.2.3 but with a different reset map, $R = \{R_{e_1}\}$, where R_{e_1} is defined by $R_{e_1}(x) = x/2$, $\forall x \in G_{e_1}$.

There is only one element, $c = \langle q_1; e_1; q_1 \rangle$, in the set C of cyclic paths. For path c, the composition of reset maps along c is $R_c = W_{e_1}$ and $\pi_E(c) = e_1$. Evaluating equation (2.4) with Z_c as the guard G_{e_1}, we get

$$
\begin{aligned}
Z_c' &= \mathbf{F}_c(\mathbb{R}^n_\infty) \\
&= R_c(\mathbb{R}^n_\infty \cap G_{e_1}) \\
&= W_{e_1}(G_{e_1}) \\
&= R_{e_1}(G_{e_1}) \cap G_{e_1} \\
&= \{x \in R \mid 0 \le x \le 5\} \cap \{x \in R \mid 0 \le x \le 10\} \\
&= \{x \in R \mid 0 \le x \le 5\}.
\end{aligned}
$$

Keep evaluating equation (2.4) with Z_c', and we get

$$
\begin{aligned}
Z_c'' &= \mathbf{F}_c(Z_c') \\
&= R_c(Z_c' \cap G_{e_1})
\end{aligned}
$$

$$
\begin{aligned}
&= W_{e_1}(Z'_c) \\
&= R_{e_1}(Z'_c) \cap Z'_c \\
&= \{x \in R \mid 0 \leq x \leq 2.5\}.
\end{aligned}
$$

Keep evaluating equation (2.4) with the new results for n times, and we will get

$$
\begin{aligned}
Z_c^{n+1} &= \mathbf{F}_c(Z_c^n) \\
&= R_c(Z_c^n \cap G_{e_1}) \\
&= W_{e_1}(Z_c^n) \\
&= R_{e_1}(Z_c^n) \cap Z_c^n \\
&= R_{e_1}(\{x \in R \mid 0 \leq x \leq (1/2)^n\}) \cap \{x \in R \mid 0 \leq x \leq (1/2)^n\} \\
&= \{x \in R \mid 0 \leq x \leq (1/2)^{n+1}\} \cap \{x \in R \mid 0 \leq x \leq (1/2)^n\} \\
&= \{x \in R \mid 0 \leq x \leq (1/2)^{n+1}\},
\end{aligned}
$$

where Z_c^n indicates the n-th result from evaluating equation (2.4).

It is obvious that the more times we evaluate the equation (2.4), the bigger n will be. However, there will not be a limit for n.

Analytically, we can conclude that the least fixed point of $\mathbf{F}_c$ is a non-empty set containing only one element, $\{0\}$. However, this result cannot be computed in a finite number steps. Therefore, our constructive procedure does not work for this kind of hybrid systems.

2.2.5 Example 5: Another Zeno Hybrid System.

Here is another Zeno hybrid system example, we use the same model in Example 2.2.3 but with a different guard set, $G_{e_1} = \{x \in R \mid x \geq 0\}$.

There is only one element, $c = \langle q_1; e_1; q_1 \rangle$, in the set C of cyclic paths. For path c, the composition of reset maps along c is $R_c = W_{e_1}$ and $\pi_E(c) = e_1$. Evaluating

equation (2.4) with Z_c as the guard G_{e_1}, we get

$$\begin{aligned} Z'_c &= \mathbf{F}_c(\mathbb{R}^n_\infty) \\ &= R_c(\mathbb{R}^n_\infty \cap G_{e_1}) \\ &= W_{e_1}(G_{e_1}) \\ &= R_{e_1}(G_{e_1}) \cap G_{e_1} \\ &= \{x \in R \mid x \geq 1\} \cap \{x \in R \mid x \geq 0\} \\ &= \{x \in R \mid x \geq 1\}. \end{aligned}$$

Keep evaluating equation (2.4) with Z'_c, we get

$$\begin{aligned} Z''_c &= \mathbf{F}_c(Z'_c) \\ &= R_c(Z'_c \cap G_{e_1}) \\ &= W_{e_1}(Z'_c) \\ &= R_{e_1}(Z'_c) \cap Z'_c \\ &= \{x \in R \mid x \geq 2\}. \end{aligned}$$

Keep evaluating equation (2.4) with the new results for n times, we will get

$$\begin{aligned} Z^{n+1}_c &= \mathbf{F}_c(Z^n_c) \\ &= R_c(Z^n_c \cap G_{e_1}) \\ &= W_{e_1}(Z^n_c) \\ &= R_{e_1}(Z^n_c) \cap Z^n_c \\ &= R_{e_1}(\{x \in R \mid x \geq n\}) \cap \{x \in R \mid x \geq n\} \\ &= \{x \in R \mid x \geq n+1\} \cap \{x \in R \mid x \geq n\} \\ &= \{x \in R \mid x \geq n+1\}, \end{aligned}$$

where Z^n_c indicates the n-th result from evaluating equation (2.4).

It is obvious that the more times we evaluate the equation (2.4), the bigger n will be. However, there will not be a limit for n.

Analytically, noting that $\mathbb{R}_\infty$ contains ∞, we can conclude that the least fixed point of $\mathbf{F}_c$ is a set containing only one element $\{\infty\}$. Therefore this system may have a Zeno behavior, we need to introduce a post-Zeno state for describing the dynamics after the Zeno point. The completion process is omitted in this report. However, we have to point out that this fixed point set cannot be computed in a finite number steps. Therefore, our constructive procedure does not work for this kind of hybrid systems.

Chapter 3

Approximate Simulation

In [11], we proposed an operational semantics for simulating hybrid system models. The key idea of the operational semantics is to treat a complete simulation as a sequence of unit executions, where a unit execution consists of two phases. The discrete phase of execution handles all discrete events at the same time point, and the continuous phase resolves the continuum between two consecutive discrete events.

When simulating a Zeno hybrid system model, we encounter more challenging practical issues. The first difficulty is that before the Zeno time point, there will be an infinite number of discrete transitions (events). Handling a discrete event takes a non-zero time. So it is impossible to handle all discrete transitions in a finite time interval. In other words, the simulation gets stuck near the Zeno time point. The second difficulty is numerical errors, which make it impractical to get an exact simulation. We will first elaborate on the second issue, and then we will come back to the first issue in subsection 3.3.

3.1 Numerical Errors

There are two sources of numerical errors: round-off error and truncation error[1]. Round-off error arises from using a finite number of bits in a computer to represent a real value. We denote this kind of difference as η. Then we can say that each integration operation will incur a round-off error of order η, denoted as $O(\eta)$. Round-off error accumulates. Suppose we integrate with a fixed step-size solver with a integration step size as h. In order to simulate over a unit time interval, we need h^{-1} integration steps, then the total round-off error is $O(\eta/h)$. Clearly, the bigger the step size, the fewer integration steps, the smaller the total round-off error. Similar results can be inferred for variable step-size solvers.

Truncation error comes from the integration algorithms used by practical ODE solvers. For example, an nth-order explicit Runge-Kutta method, which is derived to match the first $n+1$ terms of Taylor's expansion, has a *local* truncation error of $O(h^{n+1})$ and an *accumulated* truncation error of $O(h^n)$. Note that both truncation errors decrease as h decreases. Ideally we will get no truncation errors as $h \to 0$.

The total numerical error ε for an ODE solver using an nth-order explicit Runge-Kutta method is the sum of the round-off error and truncation error,

$$\varepsilon \sim \eta/h + h^n. \tag{3.1}$$

We can see that with a big integration step size h, the total error is dominated by truncation error, whereas round-off error dominates with a small step size. Therefore, although it is desirable to choose a small step size to reduce truncation error, the accuracy of a calculation result may not be increased due to the accumulation of round-off error. If we take the derivative of (3.1) with respect to h, then we get that

[1]We will not give a thorough discussion of numerical errors, which have been extensively studied, e.g. in [12]. We would rather briefly review and explain the important trade-offs when choosing integration step sizes.

when $h \sim \eta^{1/(n+1)}$ the total error ε reaches its minimum $O(\eta^{n/(n+1)})$. Therefore, in practice, we need to set a lower bound for both the integration step size and error tolerance (or value resolution) of integration results. We denote them as h_0 and ε_0 respectively, where

$$h_0 \ \sim \ \eta^{1/(n+1)}, \quad \varepsilon_0 \ \sim \ \eta^{n/(n+1)}.$$

For a good simulation, accuracy is one concern and efficiency is another objective. Efficiency for numerical integration is usually measured in terms of computation time or the number of computing operations. Using a big integration step size is an effective way to improve efficiency but with the penalty of loss of accuracy. So there is a trade-off. Furthermore, step sizes have upper bounds that are enforced by the consistency, convergence, and stability requirements when deploying practical integration methods on concrete ODEs [12]. Therefore, most practical *adaptive* ODE solvers embed a mechanism inside the integration process to adjust the step size to meet the desired tolerance of integration results, so that efficiency gets improved while maintaining the required accuracy at the same time.

In summary, a practical ODE solver usually specifies a minimum integration step size h_0, some small error tolerance ε_0, and an algorithm to adapt step size to meet requirements on both efficiency and accuracy.

3.2 Computation Difficulties

It is well-known that numerical integration in general can only deliver an approximation to the exact solution of an initial value ODE. However, the distance of the approximation from the exact solution is controllable for certain kinds of vector fields. For example, if a vector field satisfies a Lipschitz condition along the time interval where it is defined, we can constrain the integration results to reside within a neigh-

borhood of the exact solution by introducing more bits for representing values to get better precision and integrating with a small step size.

The same difficulties that arise in numerical integration also appear in event detection. A few algorithms have been developed to solve this problem [13]–[15]. However, there is still a fundamental unsolvable difficulty: we can only get the simulation time close to the time point where an event occurs, but we are not assured of being able to determine that point precisely.

Simulating a Zeno hybrid system poses another fundamental difficulty. We will first explain it through a simple continuous-time example with dynamics

$$\dot{x}(t) = 1/(t-1),\ x(0) = 0,\ t \in [0, 2]. \tag{3.2}$$

We can analytically find the solution for this example, $x(t) = \ln|t-1|$. However, getting the same result through simulation is difficult. Suppose the simulation starts with $t = 0$. As t approaches 1, the derivative $\dot{x}(t)$ keeps decreasing without bound. To satisfy the convergence and stability requirements, the step size h has to be decreased. When the step size becomes smaller than h_0, round-off error is not neglectable any more and the simulation results become unreliable. Trying to reduce the step size further doesn't help, because the disturbance from round-off error will dominate.

A similar problem arises when simulating Zeno hybrid system models. Recall that Zeno executions have an infinite number of discrete events (transitions) before reaching the Zeno time point, and the time intervals between two consecutive transitions shrink to 0. When the time interval becomes less than h_0, round-off errors again dominate.

In summary, it is impractical to precisely simulate the behavior of a Zeno model. Therefore, similar to numerical integration, we need to develop a computationally feasible way to approximate the exact model behavior. The objective is to give a

close approximation under the limits enforced by numerical errors. We will do this in the next subsection.

3.3 Approximating Zeno Behaviors

In Sect. 2, we have described how to specify the behaviors of a Zeno hybrid system before and after the Zeno time point and how to develop transitions from pre-Zeno states to post-Zeno states. The construction procedure works for guards which are arbitrary sets. However, assuming that each guard is the sub-levelset of a function (or collection of functions) simplifies the framework for studying transitions to post-Zeno states. Therefore, we assume that a transition going from a pre-Zeno state to a post-Zeno state has a guard expression of form,

$$G_{e_c} = \{x \in \mathbb{R}^n_\infty \mid g_{e_c}(x) \leq 0\}, \tag{3.3}$$

for every $c \in C$, where $e_c = h^{-1}(c)$ and $g_{e_c} : \mathbb{R}^n_\infty \to \mathbb{R}^k_\infty$. Furthermore, we assume that $g_{e_c}(x)$ is continuously differentiable.

In this section, we will develop an algorithm such that the complete model behavior can be simulated. As the previous subsection pointed out, we can only approximate the model behaviors before the Zeno time point. Therefore, the first issue is to be able to tell how close the simulation results are to the exact solutions before the Zeno time point. This will decide when the transitions from pre-Zeno states to post-Zeno states are taken. The second issue is how to establish the initial conditions of the dynamics after the Zeno time point from the approximated simulation results.

3.3.1 Issue 1: Relaxing Guard Expressions.

To solve the first issue, we first relax the guard conditions defining the transitions from the pre-Zeno states to the post-Zeno states; if the current states fall into a neighborhood of the Zeno states (the states at the Zeno time point), the guard is enabled and transition is taken. Note that when the transition is taken, the system has a new dynamics and the rest of the events before the Zeno time point, which are infinite in number, are discarded. Therefore the computation before the approximated Zeno time point can be finished in finite time.

A practical problem now is to define a good neighborhood such that the approximation is "close enough" to the exact Zeno behavior. We propose two criteria. The first criterion is based on the error tolerance ε_0[2]. We rewrite (3.3) as

$$G_{e_c}^{\varepsilon_0} = \{x \in \mathbb{R}_\infty^n \mid g_{e_c}(x) \leq \varepsilon_0\}, \tag{3.4}$$

meaning if $x(t)$ is the solution of $\dot{x} = f_q(x)$ with $q = \bar{\mathfrak{s}}(e_c)$, and if the evaluation result of $g_{e_c}(x(t))$ falls inside $[0, \varepsilon_0]$, the simulation results of $x(t)$ will be thought as close enough to the exact solution at the Zeno time point, and the transition will be taken. In fact, because ε_0 is the smallest amount that can be reliably distinguished, any value in $[0, \varepsilon_0]$ will be treated the same.

The second criterion is based on the minimum step size h_0. Suppose the evaluation result of $g_{e_c}(x(t))$ is outside of the range $[0, \varepsilon_0]$. If it takes less than h_0 time for the dynamics to drive the value of $g_{e_c}(x(t))$ down to 0, then we will treat the current states as close enough to the Zeno states. This criterion prevents the numerical integration from failing with a step size smaller than h_0, which may be caused by some rapidly changing dynamics, such as those in (3.2).

[2]If $g_{e_c}(x)$ is a vector valued function, then ε_0 is a vector with ε_0 as the elements.

We first get a linear approximation to function $g_{e_c}(x(t))$ around t_0 (cf. [13], [15]),

$$g_{e_c}(x(t_0+h)) = g_{e_c}(x(t_0)) + \frac{\partial g_{e_c}(x)}{\partial x} \cdot f_q(x) \mid_{x=x(t_0)} \cdot h + O(h^2), \tag{3.5}$$

where h is the integration step size. Because we are interested in the model's behavior when h is close to h_0, where h is very small, we can discard the $O(h^2)$ term in (3.5). We are interested in how long it takes for the value of function $g_{e_c}(x(t_0))$ to go to 0, so we calculate the required step size by solving (3.5),

$$h = -\frac{g_{e_c}(x(t_0))}{\frac{\partial g_{e_c}(x)}{\partial x} \cdot f_q(x) \mid_{x=x(t_0)}}. \tag{3.6}$$

Now we say that if $h < h_0$, the states are close enough to the Zeno point. So we rewrite the boolean expression (3.3) as

$$G_{e_c}^{h_0} = \left\{ x \in \mathbb{R}^n_\infty \mid -\frac{g_{e_c}(x)}{\frac{\partial g_{e_c}(x)}{\partial x} \cdot f_q(x)} \leq h_0 \right\}. \tag{3.7}$$

In the end, we give a complete approximated guard expression of the transition e_c from a pre-Zeno state to a post-Zeno state:

$$G_{e_c}^{\text{approx}} = G_{e_c}^{\varepsilon_0} \cup G_{e_c}^{h_0}.$$

This means that if either guard expression in (3.4) and (3.7) evaluates to be true, the transition will be taken. Performing this process on each guard in the set $\{G_{e_c} \mid c \in C\}$ we obtain the set $\{G_{e_c}^{\text{approx}} \mid c \in C\}$. Note that to ensure deterministic transitions, we also subtract the same set from the original guard sets defined in (2.6). Replacing the guard expressions given in Sect. 2 with these approximated ones, we obtain an approximation to the completed hybrid system $\overline{\mathscr{H}}$, $\overline{\mathscr{H}}^{\text{approx}}$. This is the completed hybrid system that is implemented for simulation.

3.3.2 Issue 2: Reinitialization.

The other issue is how to reinitialize the initial continuous states of the new dynamics defined in a post-Zeno state. Theoretically, these initial continuous states

are just the states at the Zeno time point, meaning that they satisfy the guard expression in (3.3). This is guaranteed by the identity reset maps associated with the transitions.

In some circumstances, like the examples discussed in this report, the initial continuous states can be explicitly and precisely calculated. However, in general, if there are more variables involved in guard expressions than the constraints enforced by guard expressions, we cannot resolve all initial states. In this case, we have to use the simulation results as part of the initial states. Clearly, since in simulation we do not actually reach the Zeno time point, the initial states are just approximations. Consequently, the simulation of the dynamics of post-Zeno states will be approximation too.

Chapter 4

Conclusion

In this report, a systematic method is introduced for completing hybrid systems through the introduction of new post-Zeno states and transitions to these states at the Zeno point. A way to approximate model behaviors at Zeno points is developed such that the simulation does not halt nor break down. With these solutions, a Zeno hybrid system model can be simulated beyond its Zeno point and its dynamics is revealed completely.

References

[1] J. Zhang, K. H. Johansson, J. Lygeros, and S. Sastry, "Zeno hybrid systems," *Int. J. Robust and Nonlinear Control*, vol. 11, no. 2, pp. 435–451, 2001.

[2] A. D. Ames, A. Abate, and S. Sastry, "Sufficient conditions for the existence of zeno behavior," in *44th IEEE Conference on Decision and Control and European Control Conference ECC*, 2005.

[3] A. D. Ames, P. Tabuada, and S. Sastry, "On the stability of Zeno equilibria," in *Hybrid Systems: Computation and Control (HSCC)*, J. P. Hespanha and A. Tiwari, Eds., vol. LNCS 3927. Santa Barbara, CA, USA: Springer-Verlag, 2006, pp. 34 – 48.

[4] K. H. Johansson, J. Lygeros, S. Sastry, and M. Egerstedt, "Simulation of zeno hybrid automata," in *Proceedings of the 38th IEEE Conference on Decision and Control*, Phoenix, AZ, 1999.

[5] A. D. Ames and S. Sastry, "Blowing up affine hybrid systems," in *43rd IEEE Conference on Decision and Control*, 2004.

[6] P. J. Mosterman, "An overview of hybrid simulation phenomena and their support by simulation packages," in *Hybrid Systems: Computation and Control (HSCC)*, F. Varager and J. H. v. Schuppen, Eds., vol. LNCS 1569. Springer-Verlag, 1999, pp. 165–177.

[7] A. D. Ames, H. Zheng, R. Gregg, and S. Sastry, "Is there life after zeno? taking executions past the breaking (zeno) point," in *American Control Conference (ACC)*, 2006.

[8] A. van der Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems*, ser. Lecture Notes in Control and Information Sciences 251. Springer-Verlag, 2000.

[9] J. Lygeros, *Lecture Notes on Hybrid Systems*. ENSIETA 2-6/2/2004, 2004.

[10] B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order*, 2nd ed. Cambridge University Press, 2002.

[11] E. A. Lee and H. Zheng, "Operational semantics of hybrid systems," in *Hybrid Systems: Computation and Control (HSCC)*, M. Morari and L. Thiele, Eds., vol. LNCS 3414. Zurich, Switzerland: Springer-Verlag, 2005, pp. pp. 25–53.

[12] R. L. Burden and J. D. Faires, *Numerical analysis, 7th ed.* Brroks/Cole, 2001.

[13] L. F. Shampine, I. Gladwell, and R. W. Brankin, "Reliable solution of special event location problems for odes," *ACM Trans. Math. Softw.*, vol. 17, no. 1, pp. 11–25, 1991.

[14] T. Park and P. I. Barton, "State event location in differential-algebraic models," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 6, no. 2, pp. 137–165, 1996.

[15] J. M. Esposito, V. Kumar, and G. J. Pappas, "Accurate event detection for simulating hybrid systems," in *Hybrid Systems: Computation and Control (HSCC)*, vol. LNCS 2034. London, UK: Springer-Verlag, 2001, pp. 204–217.