UNIVERSITY OF CALIFORNIA
SANTA CRUZ

PROVABLY-CORRECT AND EFFICIENT MOTION PLANNING FOR HYBRID DYNAMICAL SYSTEMS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

by

Nan Wang

December 2024

The Dissertation of Nan Wang
is approved:

Professor Ricardo G. Sanfelice, Chair

Professor Gabriel H. Elkaim

Professor Steve McGuire

Professor Daniel Fremont

Peter F. Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

Notations

$\mathbb{R}^n$	n-dimensional Euclidean space
$\mathbb{R}$	The set of all real numbers
$\mathbb{R}_{\geq 0}$	The set of all nonnegative real numbers
$\mathbb{N}$	The set of nonnegative integers
$\lvert x\rvert$	The Euclidean norm of a vector x
$\operatorname{int} I$	The interior of the interval I
$\overline{S}$	The closure of the set S
∂S	The boundary of the set S
$P + Q$	The Minkowski sum of set P and set Q, namely, the set $\{p + q : p \in P, q \in Q\}$
$P - Q$	The Minkowski difference of set P and set Q, namely, the set $\{p - q : p \in P, q \in Q\}$
$\lvert x\rvert_S$	The distance between a point $x \in \mathbb{R}^n$ and a set $S \subset \mathbb{R}^n$, i.e., $\lvert x\rvert_S := \inf_{s \in S} \lvert x - s\rvert$
$\mathbb{B}$	The closed unit ball of appropriate dimension in the Euclidean norm

Prob(M)	The probability of the probabilistic event M
$\mu(S)$	The Lebesgue measure of set S
ζ_n	The Lebesgue measure of the n-th dimensional unit ball, i.e., $$\zeta_n := \begin{cases} \dfrac{\pi^k}{k!} & \text{if } n = 2k, k \in \mathbb{N} \\ \dfrac{2(k!)(4\pi)^k}{(2k+1)!} & \text{if } n = 2k+1, k \in \mathbb{N}; \end{cases} \quad (0.1)$$ see [16]
rge$f$	The range of the function f
dom f	The domain of the function f

List of Figures

List of Tables

To my loving family, steadfast friends, and all who have graciously supported and inspired me throughout this journey.

Acknowledgments

I am profoundly grateful to my advisor and mentor, Ricardo Sanfelice, for his unwavering support and invaluable guidance through both the academic and personal challenges I've faced. His wisdom and altitude have inspired me to pursue rigorous and mathematically sound research, making him an exemplary model for my academic journey and future professional endeavors.

I am thankful to my other committee members, Prof. Gabriel Elkaim, Prof. Daniel Fremont, and Prof. Steve McGuire for their wisdom and their help on my dissertation. My gratitude also goes to my collaborators Dr. Stefano Di Cairano, Prof. Adeel Akhtar, Eric Partika, Adam Ames and Beverly Xu for their insight and support.

A special thank you to my peers in the Hybrid Systems Laboratory: Dr. Kunal Garg, Prof. Adnane Saoud, Dr. Berk Altin, Dr. Mohamed Maghenem, Dr. Himadri Basu, Dr. Malladi BharaniPrabha, Dr. Nathalie Risso, Dr. Hyejin Han, Dr. Marcello Guarro, Dr. Dawn Hustig-Schultz, Dr. Parmita Ojaghi, Dr. Ryan Johnson, Dr. Santiago Jiminez Leudo, Masoumeh Ghanbarpour, Paul Wintz, Carlos Montenegro, Jan de Priester, Piyush Jirawankar, Nathan Wu, Xi Luo, Tommy Snijders, Hyung Tae Choi, Roger Berman, Eric Partika, David Kooi, Adam Ames, Haoyue Gao, Harsh Bhakta, Zachary Lamb, Jake Nations, Ishan Madan, Indy Spott, Kevin Sandoval, Cdric Chartier, and Akhil Datla. Your company during this journey has transformed the Hybrid Systems Laboratory into a cherished second home.

I am immensely thankful for the support of my family and friends throughout

my graduate studies. A heartfelt thank you to my mother, Liandong Gong, and my father, Haiping Wang, for their unconditional love and encouragement. Their courage in letting their only child study so far from home laid the most solid foundation for this dissertation. Among the many friends I have been fortunate to meet over the last six years, I would especially like to thank Pengyang Zhou and Dr. Jing Xiong, for the delicious food they made, and treasured experiences we have shared through this journey.

Abstract

Provably-Correct and Efficient Motion Planning for Hybrid Dynamical Systems

by

Nan Wang

Motion planning serves as a pivotal role in various robotics applications, endowing robots with intelligence and autonomy. It translates human commands and tasks into a reference trajectory, with a tracking controller steering the system along the reference trajectory, eventually implementing the commands and tasks. Despite extensive study in motion planning for continuous-time and discrete-time systems, the emergence of hybrid dynamical systems underscores their limitations. Hybrid dynamical systems feature state variables that may evolve continuously (flow) and, at times, evolve discretely (jump). Compared with motion planning for pure continuous-time/discrete-time system, only a few efforts have been devoted to motion planning for a special class of hybrid dynamical systems. Motivated by these gaps, this dissertation focuses on developing provably-correct and efficient motion planning algorithms for a broad class of hybrid dynamical systems.

Firstly, this dissertation lays the groundwork by establishing the fundamental theory of motion planning for hybrid dynamical systems. The motion planning problem for hybrid dynamical systems is formulated using the hybrid equations framework, which is general to capture most hybrid dynamical systems. To overcome the lack of the systematic analysis on the propagation, reversal, concatenation, and truncation operations,

which are used in almost all motion planning algorithms, on the solutions to hybrid dynamical systems, this dissertation formalizes the definitions of those operations for the hybrid dynamical systems. This dissertation proposes a bidirectional propagation algorithm template that describes a general framework using the aforementioned operations to solve the motion planning problem for hybrid dynamical systems.

Secondly, a rapidly-exploring random trees (RRT) implementation of the proposed algorithm template is designed to solve the motion planning problem for hybrid dynamical systems. At each iteration, the proposed algorithm, called HyRRT, randomly picks a state sample and extends the search tree by flow or jump, which is also chosen randomly when both regimes are possible. Through a definition of concatenation of functions defined on hybrid time domains, we show that HyRRT is probabilistically complete, namely, the probability of failing to find a motion plan approaches zero as the number of iterations of the algorithm increases. This property is guaranteed under mild conditions on the data defining the motion plan, which include a relaxation of the usual positive clearance assumption imposed in the literature of classical systems. The motion plan is computed through the solution of two optimization problems, one associated with the flow and the other with the jumps of the system.

Thirdly, this dissertation proposes a bidirectional RRT algorithm to solve the motion planning problem for hybrid dynamical systems, accelerating the search process. The proposed algorithm, called HyRRT-Connect, propagates in both forward and backward directions in hybrid time until an overlap between the forward and backward propagation results is detected. Then, HyRRT-Connect constructs a motion

plan through the reversal and concatenation of functions defined on hybrid time domains, ensuring the motion plan thoroughly satisfies the given hybrid dynamics. To address the potential discontinuity along the flow caused by tolerating some distance between the forward and backward partial motion plans, we reconstruct the backward partial motion plan by a forward-in-hybrid-time simulation from the final state of the forward partial motion plan. By applying the reversed input of the backward partial motion plan, the reconstruction process effectively eliminates the discontinuity and ensures that as the tolerance distance decreases to zero, the distance between the endpoint of the reconstructed motion plan and the final state set approaches zero.

At last, we formulate an optimal motion planning problem for hybrid dynamical systems and design a stable sparse RRT (SST) algorithm to solve the optimal motion planning problem for hybrid dynamical systems. At each iteration, the proposed algorithm, called HySST, selects a vertex with minimal cost among all the vertices within the neighborhood of a random sample, subsequently extending the search tree. In addition, HySST maintains a static set of witness points where all vertices within each witness's neighborhood are pruned, except for the ones with lowest cost. We show that HySST is asymptotically near-optimal, namely, the probability of failing to find a motion plan with cost close to the optimal approaches zero as the number of iterations of the algorithm increases to infinity.

The contributions of this dissertation go beyond the development of certain motion planning algorithms. It introduces a novel motion planning problem for a general dynamical system and furnishes theoretical tools, enabling researchers to design their

own algorithms for solving this problem while ensuring completeness guarantees. In addition, a hybrid dynamical system serves as a powerful modeling tool for addressing difficult motion planning problems. By employing a generic hybrid dynamical system model, users can tackle challenging motion planning tasks efficiently, with minimal modeling effort and theoretical guarantees.

Publication List:

1. **Wang, Nan**, and Ricardo G. Sanfelice. "Motion Planning for Hybrid Dynamical Systems: Framework, Algorithm Template, and a Sampling-based Approach" International Journal of Robotics Research (IJRR).

2. **Wang, Nan**, and Ricardo G. Sanfelice. "HyRRT-Connect: A Bidirectional Rapidly-Exploring Random Trees Motion Planning Algorithm for Hybrid Systems." 2024 IFAC 8th Conference on Analysis and Design of Hybrid Systems (ADHS). IFAC, 2024.

3. **Wang, Nan**, Stefano Di Cairano, and Ricardo G. Sanfelice. "A Switched Reference Governor for High Performance Trajectory Tracking." 2024 the American Control Conference (ACC), July, 2024.

4. **Wang, Nan**, and Ricardo G. Sanfelice. "HySST: An Asymptotically Near-Optimal Motion Planning Algorithm for Hybrid Systems." 2023 IEEE 62nd Conference on Decision and Control (CDC). IEEE, 2023.

5. **Wang, Nan**, and Ricardo G. Sanfelice. "A rapidly-exploring random trees motion planning algorithm for hybrid dynamical systems." 2022 IEEE 61st Conference on Decision and Control (CDC). IEEE, 2022.

6. Xu, Beverly, **Wang, Nan**, and Ricardo G. Sanfelice. "cHyRRT and cHySST: Two Motion Planning Tools for Hybrid Dynamical Systems." 28th ACM International Conference on Hybrid Systems: Computation and Control (HSCC). ACM, 2025. (submitted)

7. Ames, Adam, **Wang, Nan**, and Ricardo G. Sanfelice. "A Set-based Motion Planning Algorithm for Aerial Vehicles in the Presence of Obstacles Exhibiting Hybrid Dynamics." 2022 IEEE Conference on Control Technology and Applications (CCTA). IEEE, 2022.

Chapter 1

Introduction

1.1 Motivation

Motion planning consists of finding a state trajectory and associated inputs, connecting the initial and final state while satisfying the system dynamics and a given safety criterion. Motion planning technology has been widely used to help robotics applications, such as autonomous driving systems [52], satellites systems [45], walking robots [19], quadrupeds [51], unmanned underwater vehicles [41], and quadrotors [37, 3], to complete complicated tasks. Particularly, the system dynamics considered in the motion planning not only depend on the mechanical design of the robot, but also depend on the internal logic and specifications and the interaction between the robot systems and the environment. The former usually leads to continuous dynamics while the later usually leads to discrete dynamics. For example, the *continuous* evolution of the states of the wheeled vehicle can be modeled by either its kinematic model [43, Page 26] or its

dynamic model [15]. On the other hand, in scenarios like designing a motion planner for a robot to move to the highest floor in a building, a logic variable representing the floor number should be employed and updated discretely [7]. In addition, a motion planner to plan the motion of consecutive steps of a biped robot should consider *discrete* changes over the foot's speed when an impact between the foot and the ground occurs [30].

However, for some motion planning tasks, the states can both evolve continuously and, from time to time, execute some discrete changes. For example, in [67], collision-resilient motion planning for multicopters involves continuous evolution in open space and discrete changes upon collisions with a wall. Another example is the Task and Motion Planning (TAMP) problem, which contains elements of discrete task planning, discrete-continuous mathematical programming, and continuous motion planning [14, 22]. The motion planning approach in [1] enables agile automated driving on a slippery surface by planning a motion that switches between different driving modes. In both mentioned examples, neither a continuous-time model nor a discrete-time model adequately captures the behaviors of the systems. However, a hybrid dynamical system model is comprehensive, encompassing not only purely continuous or discrete-time systems but also those exhibiting both continuous and discrete behaviors. This motivates the research on motion planning for hybrid dynamical systems in this dissertation. Unlike purely continuous-time or discrete-time systems, typically modeled by a single differential or difference equation, hybrid dynamical systems within specific classes can be represented using different formulations. Previous research has explored motion planning for certain classes of hybrid dynamical systems. This dissertation focuses on

motion planning problems for hybrid dynamical systems modeled as hybrid equations [17]. In this modeling framework, differential and difference equations with constraints are used to describe the continuous and discrete behavior of the hybrid dynamical system, respectively. This general hybrid dynamical system framework can capture most hybrid dynamical systems emerging in robotic applications, not only the class of hybrid dynamical systems considered in [64] and [6], but also systems with memory states, timers, impulses, and interaction with the environment.

To efficiently solve the motion planning problems for hybrid dynamical systems with theoretical guarantees, we develop three RRT-type motion planning algorithms that implement motion planning tasks [56, 61, 62], achieve rapid searching speeds [59, 60], and optimize given cost functionals [58, 57], respectively.

1.2 Basic Operations on the Trajectory for Hybrid Dynamical Systems

1.2.1 Related Work

Most existing motion planning algorithms, such as RRT algorithm [34], incrementally construct a search tree in the state space and seek for a path in the search tree connecting the initial and final states. Such algorithms typically require propagation, concatenation, truncation, and, if they propagate in both forward and backward time, reversal operations of trajectories. Definitions of these operations for purely continuous-time systems and for purely discrete-time systems are avail-

able in the literature and heavily used for motion planning. For example, given a continuous function $x_1 : [0, t_1] \to \mathbb{R}$ where t_1 is a positive real number, the reversal of x_1 is $x_1' : [0, t_1] \to \mathbb{R}$ such that $x_1'(t) = x_1(t_1 - t)$. The other example is that given two discrete functions $x_1 : \{0, 1, ..., j_1\} \to \mathbb{R}$ and $x_2 : \{0, 1, ..., j_2\} \to \mathbb{R}$ where j_1 and j_2 are some positive integers, the concatenation of x_2 to the end of x_1 is $x_c : \{0, 1, ..., j_1 + j_2\} \to \mathbb{R}$ such that $x_c(j) = x_1(j)$ for all $j \in \{0, 1, ..., j_1\}$ and $x_c(j) = x_2(j - j_1)$ for all $t \in \{j_1 + 1, j_1 + 2, ..., j_1 + j_2\}$. Those definitions are used in most motion planning literature, see, e.g., [33, Chapter 14.3].

However, defining such operations for trajectories of hybrid dynamical systems is challenging because of their much more complicated domain structure. In fact, it is all possible that a trajectory of a hybrid dynamical system 1) evolves purely continuously, 2) exhibits jumps all the time, 3) evolves continuous and exhibits one or multiple jumps at times, or 4) exhibits Zeno behavior. To the authors' best knowledge, except for the concatenation operation defined in [2, Definition 5.1], there is no existing work formulating such operations for hybrid dynamical systems, let alone their properties.

1.2.2 Contributions

The main contributions of the forthcoming Chapter 3 are as follows.

1) *Mathematical formulation of the motion planning problem for hybrid dynamical systems*: In Section 3.1, inspired by the motion planning formulation in the classic literature [33] and the definition of solution to hybrid dynamical systems in [47], we formulate a motion planning problem for a broad class of hybrid dynamical

systems modeled using the hybrid equation framework.

2) *Formal definitions for operations used in motion planning algorithms for hybrid dynamical systems*: In Section 3.2, we rigorously establish the mathematical formulations for the concatenation, reversal, truncation, and propagation operations, while also providing theoretical validation for the exact implementation of those operations.

3) *A bidirectional propagation algorithm template*: In Section 3.3, we design a bidirectional propagation algorithm template that provides key insight on how to employ the aforementioned operations to solve the motion planning problem for hybrid dynamical systems.

4) *A general method to construct backward-in-time hybrid dynamical systems*: In Section 3.3.1, we present a general method for constructing backward-in-time hybrid dynamical systems, facilitating the backward propagation of solutions in the bidirectional propagation algorithm. Building upon the earlier introduction of reversal propagation in Section 3.2, we demonstrate that reversing the backward-in-time hybrid dynamical systems preserves the original hybrid dynamics.

1.3 Sampling-based Feasible Motion Planning for Hybrid Dynamical Systems

1.3.1 Related Work

In recent years, various planning algorithms have been developed to solve motion planning problems, from graph search algorithms [36] to artificial potential [26] and fluid-flow field methods [49, 63, 50]. A main drawback of graph search algorithms is that the number of vertices grows exponentially as the dimension of states grows, which makes computing motion plans inefficient for high-dimensional systems. The artificial potential field method suffers from getting stuck at local minimum. Arguably, the most successful algorithm to solve motion planning problems for purely continuous-time systems and purely discrete-time systems is the sampling-based Rapidly-exploring Random Tree (RRT) algorithm [32]. This algorithm incrementally constructs a tree of state trajectories toward random samples in the state space. Similar to graph search algorithms, RRT suffers from the curse of dimensionality, but, in practice, achieves rapid exploration in solving high-dimensional motion planning problems [10]. Compared with the artificial potential field method, RRT is probabilistically complete [34], which means that the probability of failing to find a motion plan converges to zero, as the number of samples approaches infinity.

While RRT algorithms have been used to solve motion planning problems for purely continuous-time systems [34] and purely discrete-time systems [7], fewer efforts have been devoted to applying RRT-type algorithms to solve motion planning prob-

lems for systems with combined continuous and discrete behavior. In [21], a modular framework, called FaSTrack, is developed to enable motion planning that is fast, dynamically feasible, and with guaranteed safety in the presence of the static obstacles. Though some discrete behavior is present in the algorithms in these references, those are limited to changes of certain parameters (e.g., waypoints) and do not consider discrete changes of the state in the dynamical model describing the system to generate a motion planning for – namely, they do not consider hybrid dynamics. For a special class of hybrid dynamical systems like continuous-time hybrid dynamical system or hybrid automata, some RRT-type motion planning algorithms, such as hybrid RRT [7] and R3T [64], have been developed to solve motion planning problems. These methodologies have also found application in falsification of hybrid automata such as the Monte-Carlo sampling algorithm [40] and the Breach toolbox [12]. These systems involve continuous states evolving continuously and discrete states (modes) that switch within a finite set of feasible modes. However, they do not encompass hybrid dynamical systems where states evolve continuously and intermittently execute jumps, which can be modeled using hybrid equation framework.

1.3.2 Contributions

The main contributions of the forthcoming Chapter 4 are as follows.

1. *Tools for an RRT-type motion planning algorithm for hybrid dynamical systems*: In Sections 4.2 to 4.4, we detail the formulation of the input library, continuous simulator, and discrete simulator essential for designing an RRT-type motion

planning algorithm for hybrid dynamical systems. The input library comprises feasible candidate inputs necessary satisfying the hybrid dynamics, while the continuous and discrete simulators compute solution pairs by applying inputs from the input library.

2. *An RRT-type motion planning algorithm for hybrid dynamical systems*: In Section 4.5, we develop an RRT-type motion planning algorithm, providing a mathematical formulation for each function within the algorithm.

3. *Probabilistic completeness guarantee*: In Section 4.6, we establish the probabilistic completeness of our proposed algorithm, indicating that if a motion plan exists, the probability of the algorithm failing to find one decreases to zero as the number of iterations approaches infinity.

1.4 A Bidirectional Sampling-based Motion Planning Approach for Hybrid Dynamical Systems

It is significantly challenging for almost all motion planning algorithms to maintain efficient computation performance, especially in solving high-dimensional problems. Although RRT-type algorithms have demonstrated notable efficiency in rapidly searching for solutions to high-dimensional problems compared to other algorithm types, there remains room for enhancing their computational performance.

1.4.1 Related Work

In [29], RRT-Connect algorithm is proposed that propagates both in forward direction and backward direction, where a notable improvement in computational performance is observed. Inspired by this work, we design a bidirectional RRT-type algorithm for hybrid dynamical systems, called HyRRT-Connect, that incrementally constructs two search trees, in which one tree is rooted in the initial state set and constructed forward in hybrid time, while the other is rooted in the final state set and constructed backward in hybrid time. However, the backward propagation is a nontrivial task for hybrid dynamical systems. In recent years, machine learning technology has made significant strides in the motion planning field. In [42], a learning-based neural planner is proposed, which incorporates existing RRT-type algorithms to generate samples in the subspace that is most likely to contain motion plans.

1.4.2 Contributions

The main contributions of the forthcoming Chapter 5 are as follows.

1. *A bidirectional RRT-type motion planning algorithm for hybrid dynamical systems*: In Section 5.2, we introduce a bidirectional RRT-type algorithm to address the motion planning problem for hybrid dynamical systems. Each function within the algorithm is accompanied by a mathematical formulation.

2. *A reconstruction process to remove the discontinuity at the concatenation point*: In Section 5.3, to address the potential discontinuity along the flow caused by

tolerating some distance between the forward and backward partial motion plans, we reconstruct the backward partial motion plan by a forward-in-hybrid-time simulation from the final state of the forward partial motion plan. By applying the reversed input of the backward partial motion plan, the reconstruction process effectively eliminates the discontinuity and ensures that as the tolerance distance decreases to zero, the distance between the endpoint of the reconstructed motion plan and the final state set approaches zero.

1.5 Sampling-based Optimal Motion Planning for Hybrid Dynamical Systems

1.5.1 Related Work

A feasible solution is not sufficient in most applications as the quality of the solution returned by the motion planning algorithms is critical. The optimality of the heuristic graph search algorithm, such as A* algorithm, is only guaranteed when the employed heuristics is admissible [20], which is difficult to verify in practice. The optimal motion planning algorithms using artificial potential field method are also developed. In [53], the artificial potential field method is combined with evolutionary algorithms to derive optimal potential field called evolutionary artificial potential field. However, there is no theoretical guarantee over the optimality of the solution when the system dynamics is considered. Arguably, the sampling-based optimal motion planning algorithms are most promising because of the success of the sampling-based algorithms in solving

the feasible motion planning problems. It has been shown in [39] that the solution returned by RRT converges to a sub-optimal solution. Therefore, variants of PRM and RRT, such as PRM* and RRT* [24], have been developed to solve optimal motion planning problems with guaranteed asymptotic optimality. However, both PRM* and RRT* require a steering function returning the solution of a two-point boundary value problem (TPBVP). Unfortunately, solutions to TPBVPs are difficult to generate for most dynamical systems, which prevents them from being widely applied. On the other hand, the stable sparse RRT (SST) algorithm [35] does not require a steering function and is guaranteed to be asymptotically near optimal, which means that the probability of finding a solution that has a cost close to the minimal cost converges to one as the number of iterations approaches infinity.

1.5.2 Contributions

The main contributions of the forthcoming Chapter 6 are as follows.

1. *An optimal motion planning problem for hybrid dynamical systems*: Building upon the feasible motion planning outlined in Section 3.1, in Section 6.1, we propose an optimal motion planning problem for a broad class of hybrid dynamical systems modeled using the hybrid equation framework.

2. *An SST-type optimal motion planning algorithm for hybrid dynamical systems*: In Section 6.3, we devise an SST-type algorithm to tackle the optimal motion planning problem for hybrid dynamical systems. Each function within the algorithm is

formulated mathematically.

3. *Asymptotically near-optimality guarantee*: In Section 6.4, we demonstrate that the proposed algorithm is asymptotically near-optimal. Specifically, as the number of iterations of the algorithm increases to infinity, the probability of failing to find a motion plan with a cost close to the optimal diminishes to zero.

1.6 Future Directions and Potential Applications

The future work outlined in Chapter 7 includes motion planning for hybrid dynamical systems under uncertainty, collaborative planning for multiple agents with hybrid dynamics, and motion planning for feedback hybrid dynamical systems.

The planning algorithms introduced in this dissertation empower robotics systems to leverage interactions with their environments to execute tasks previously beyond their capacity. Moreover, incorporating temporal logic/task specifications into the dynamics enables motion planning under logic specifications. The key advantage lies in modeling all these tasks using the generic hybrid dynamical system model and employing any motion planning algorithms for this model. To enhance the application, the proposed RRT-type algorithms are implemented in C++, as detailed in [65]. The resulting motion plan can be used as the reference for the tracking controllers as in [55, 54].

Chapter 2

Preliminaries

In this chapter, we present the hybrid dynamical systems framework and its basic properties.

2.1 Hybrid Dynamical System Model

Following [47], a hybrid dynamical system $\mathcal{H}$ with inputs is modeled as

$$\mathcal{H} : \begin{cases} \dot{x} = f(x,u) & (x,u) \in C \\ x^+ = g(x,u) & (x,u) \in D \end{cases} \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, $C \subset \mathbb{R}^n \times \mathbb{R}^m$ represents the flow set, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ represents the flow map, $D \subset \mathbb{R}^n \times \mathbb{R}^m$ represents the jump set, and $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ represents the jump map, respectively. The continuous evolution of x is captured by the flow map f. The discrete evolution of x is captured by the jump map g. The flow set C collects the points where the state can evolve continuously. The jump set D collects the points where jumps can occur.

Given a flow set C, the set $U_C := \{u \in \mathbb{R}^m : \exists x \in \mathbb{R}^n \text{ such that } (x,u) \in C\}$ includes all possible input values that can be applied during flows. Similarly, given a jump set D, the set $U_D := \{u \in \mathbb{R}^m : \exists x \in \mathbb{R}^n \text{ such that } (x,u) \in D\}$ includes all possible input values that can be applied at jumps. These sets satisfy $C \subset \mathbb{R}^n \times U_C$ and $D \subset \mathbb{R}^n \times U_D$. Given a set $K \subset \mathbb{R}^n \times U_\star$, where $\star$ is either C or D, we define

$$\Pi_\star(K) := \{x : \exists u \in U_\star \text{ s.t. } (x,u) \in K\} \tag{2.2}$$

as the projection of K onto $\mathbb{R}^n$, and define

$$C' := \Pi_C(C) \tag{2.3}$$

and

$$D' := \Pi_D(D). \tag{2.4}$$

In addition to ordinary time $t \in \mathbb{R}_{\geq 0}$, we employ $j \in \mathbb{N}$ to denote the number of jumps of the evolution of x and u for $\mathcal{H}$ in (2.1), leading to hybrid time (t,j) for the parameterization of its solutions and inputs. The domain of a solution to $\mathcal{H}$ is given by a hybrid time domain as is defined as follows.

Definition 2.1 (Hybrid time domain)**.** *A set $E \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ is a hybrid time domain if, for each $(T,J) \in E$, the set*

$$E \cap ([0,T] \times \{0,1,...,J\})$$

can be written in the form

$$\bigcup_{j=0}^{J} ([t_j, t_{j+1}] \times \{j\})$$

for some finite sequence of times $\{t_j\}_{j=0}^{J+1}$ satisfying $0 = t_0 \leq t_1 \leq t_2 \leq ... \leq t_{J+1} = T$.

Given a compact hybrid time domain E, we denote the maximum of t and j coordinates of points in E as

$$\max_t E := \max\{t \in \mathbb{R}_{\geq 0} : \exists j \in \mathbb{N} \text{ such that } (t, j) \in E\} \tag{2.5}$$

and

$$\max_j E := \max\{j \in \mathbb{N} : \exists t \in \mathbb{R}_{\geq 0} \text{ such that } (t, j) \in E\}. \tag{2.6}$$

The input to the hybrid dynamical systems is defined as a function on a hybrid time domain as follows.

Definition 2.2 (Hybrid input)**.** *A function $\upsilon : dom\ \upsilon \to \mathbb{R}^n$ is a hybrid input if dom υ is a hybrid time domain and if, for each $j \in \mathbb{N}$, $t \mapsto \upsilon(t, j)$ is Lebesgue measurable and locally essentially bounded on the interval $I_\upsilon^j := \{t : (t, j) \in dom\ \upsilon\}$.*

A hybrid arc describes the state trajectory of the system as is defined as follows.

Definition 2.3 (Hybrid arc)**.** *A function $\phi : dom\ \phi \to \mathbb{R}^n$ is a hybrid arc if dom ϕ is a hybrid time domain and if, for each $j \in \mathbb{N}$, $t \mapsto \phi(t, j)$ is locally absolutely continuous on the interval $I_\phi^j := \{t : (t, j) \in dom\ \phi\}$.*

Definition 2.4 (Types of hybrid arcs)**.** *A hybrid arc ϕ is called*

- nontrivial *if dom ϕ contains at least two points;*
- purely discrete *if nontrivial and dom $\phi \subset \{0\} \times \mathbb{N}$;*
- purely continuous *if nontrivial and dom $\phi \subset \mathbb{R}_{\geq 0} \times \{0\}$.*

The definition of solution pair to a hybrid dynamical system is given as follows.

Definition 2.5 (Solution pair to a hybrid dynamical system)**.** *A hybrid input υ and a hybrid arc ϕ define a solution pair (ϕ, υ) to the hybrid dynamical system $\mathcal{H}$ if*

1) $(\phi(0,0), \upsilon(0,0)) \in \overline{C} \cup D$ and dom ϕ = dom υ(= dom (ϕ, υ)).

2) For each $j \in \mathbb{N}$ such that I_ϕ^j has nonempty interior $\operatorname{int}(I_\phi^j)$, (ϕ, υ) satisfies

$$(\phi(t,j), \upsilon(t,j)) \in C$$

for all $t \in \operatorname{int} I_\phi^j$, and

$$\frac{d}{dt}\phi(t,j) = f(\phi(t,j), \upsilon(t,j))$$

for almost all $t \in I_\phi^j$.

3) For all $(t,j) \in$ dom (ϕ, υ) such that $(t, j+1) \in$ dom (ϕ, υ),

$$\begin{aligned} (\phi(t,j), \upsilon(t,j)) &\in D \\ \phi(t, j+1) &= g(\phi(t,j), \upsilon(t,j)). \end{aligned} \tag{2.7}$$

In the main result, the following definition of closeness between hybrid arcs is used.

Definition 2.6 ((τ,ϵ)-closeness of hybrid arcs)**.** *Given $\tau, \epsilon > 0$, two hybrid arcs ϕ_1 and ϕ_2 are (τ,ϵ)-close if*

1. *for all $(t,j) \in$ dom ϕ_1 with $t + j \leq \tau$, there exists s such that $(s,j) \in$ dom ϕ_2, $|t-s| < \epsilon$, and $|\phi_1(t,j) - \phi_2(s,j)| < \epsilon$;*

2. *for all $(t,j) \in$ dom ϕ_2 with $t + j \leq \tau$, there exists s such that $(s,j) \in$ dom ϕ_1, $|t-s| < \epsilon$, and $|\phi_2(t,j) - \phi_1(s,j)| < \epsilon$.*

Chapter 3

Framework, Basic Operations, and Algorithm Template

This chapter focuses on the motion planning problem for hybrid dynamical systems. Firstly, the motion planning problem for hybrid dynamical systems is formulated using the hybrid equation framework, which is general to capture most hybrid dynamical systems. Secondly, to overcome the lack of the systematic analysis on the propagation, reversal, concatenation, and truncation operations, which are used in almost all motion planning algorithms, on the solutions to hybrid dynamical systems, this chapter formalizes the definitions of those operations for the hybrid dynamical systems and validates them theoretically. Thirdly, this chapter proposes a bidirectional propagation algorithm template that describes a general framework using the aforementioned operations to solve the motion planning problem for hybrid dynamical systems.

3.1 Feasible Motion Planning Problem for Hybrid Dynamical Systems

The motion planning problem for hybrid dynamical systems is formulated as follows.

Problem 3.1 (Feasible motion planning problem for hybrid dynamical systems)**.** *Given a hybrid dynamical system $\mathcal{H}$ with input $u \in \mathbb{R}^m$ and state $x \in \mathbb{R}^n$, initial state set $X_0 \subset \mathbb{R}^n$, final state set $X_f \subset \mathbb{R}^n$, and unsafe set $X_u \subset \mathbb{R}^n \times \mathbb{R}^m$, find a pair (ϕ, υ) : dom $(\phi, u) \to \mathbb{R}^n \times \mathbb{R}^m$, namely,* a motion plan*, such that for some $(T, J) \in$ dom (ϕ, υ), the following hold:*

1. *$\phi(0, 0) \in X_0$, namely, the initial state of the solution belongs to the given initial state set X_0;*
2. *(ϕ, υ) is a solution pair to $\mathcal{H}$ as defined in Definition 2.5;*
3. *(T, J) is such that $\phi(T, J) \in X_f$, namely, the solution belongs to the final state set at hybrid time (T, J);*
4. *$(\phi(t, j), \upsilon(t, j)) \notin X_u$ for each $(t, j) \in$ dom (ϕ, υ) such that $t + j \leq T + J$, namely, the solution pair does not intersect with the unsafe set before its state trajectory reaches the final state set.*

Therefore, given sets X_0, X_f, and X_u, and a hybrid dynamical system $\mathcal{H}$ with data (C, f, D, g), a motion planning problem for hybrid dynamical systems, denoted $\mathcal{P}$, is defined as $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$.

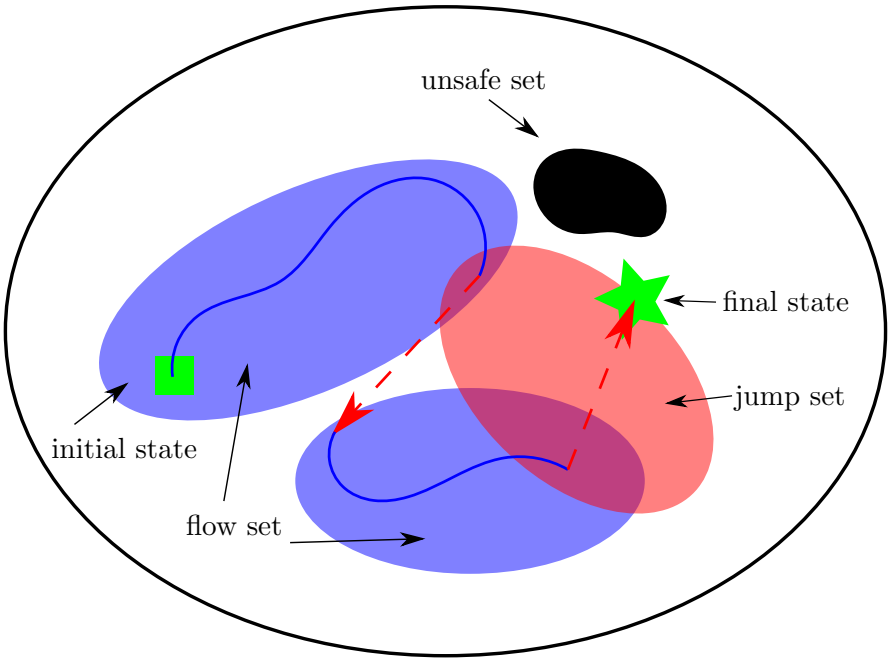


Figure 3.1: Motion planning for hybrid dynamical systems.

There are some special cases of Problem 3.1. For example, when $D = \emptyset$ and C is nonempty, then $\mathcal{P}$ denotes the motion planning problem for purely continuous-time systems. In addition, when $C = \emptyset$ and D is nonempty, then $\mathcal{P}$ denotes the motion planning problem for purely discrete-time systems. Therefore, Problem 3.1 covers the motion planning problems for purely continuous-time and purely discrete-time systems studied in [34] and [33], respectively. The algorithm template to solve the motion planning problem is illustrated in the following examples.

Example 3.2 (Actuated bouncing ball system)**.** *Consider a ball bouncing on a fixed horizontal surface as is shown in Figure 3.2. The surface is located at the origin and, through control actions, is capable of affecting the velocity of the ball after the impact.*

The dynamics of the ball while in the air is given by

$$\dot{x} = \begin{bmatrix} x_2 \\ \\ -\gamma \end{bmatrix} =: f(x,u) \qquad (x,u) \in C \tag{3.1}$$

where $x := (x_1, x_2) \in \mathbb{R}^2$. *The height of the ball is denoted by* x_1. *The velocity of the ball is denoted by* x_2. *The gravity constant is denoted by* γ. *The flow is allowed when*

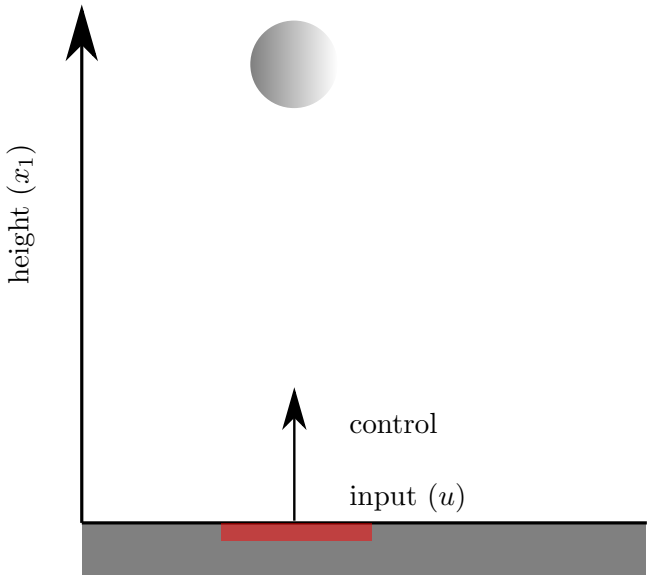


Figure 3.2: A motion plan to the sample motion planning problem for actuated bouncing ball system.

the ball is above the surface. Hence, the flow set is

$$C := \{(x,u) \in \mathbb{R}^2 \times \mathbb{R} : x_1 \geq 0\}. \tag{3.2}$$

At every impact, the velocity of the ball changes from pointing down to pointing up while the height remains the same. The dynamics at jumps of the actuated bouncing ball

system is given as

$$x^+ = \begin{bmatrix} x_1 \\ -\lambda x_2 + u \end{bmatrix} =: g(x,u) \qquad (x,u) \in D \tag{3.3}$$

where $u \geq 0$ *is the input and* $\lambda \in (0,1)$ *is the coefficient of restitution. Jumps are allowed when the ball is on the surface with negative velocity. Hence, the jump set is*

$$D := \{(x,u) \in \mathbb{R}^2 \times \mathbb{R} : x_1 = 0, x_2 \leq 0, u \geq 0\}. \tag{3.4}$$

The hybrid model of the actuated bouncing ball system is given by (2.1) where the flow

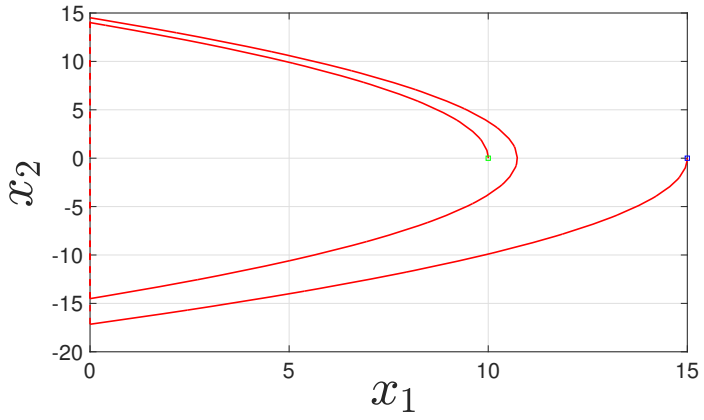


Figure 3.3: The actuated bouncing ball system in Example 3.2. The final state set is denoted by a green square. The red trajectory denotes the state trajectory of a sample motion plan.

map f *is given in (3.1), the flow set* C *is given in (3.2), the jump map* g *is given in (3.3), and the jump set* D *is given in (3.4). A sample motion planning problem for the actuated bouncing ball system is given as follows. The initial state set is* $X_0 = \{(15,0)\}$*. The final state set is* $X_f = \{(10,0)\}$*. The unsafe set is* $X_u = \{(x,u) \in \mathbb{R}^2 \times \mathbb{R} : u \in [5,\infty)\}$*. The motion planning problem* $\mathcal{P}$ *is given as* $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$*. A state trajectory*

solving this motion planning problem is shown in Figure 3.3. In the figure, the initial state set is denoted by a blue square. The final state set is denoted by a green square. The red trajectory denotes the state trajectory of a sample motion plan.

Example 3.3 (Walking robot)**.** *The state x of the compass model of a walking robot is composed of the angle vector θ and the velocity vector ω [19]. The angle vector θ contains the planted leg angle θ_p, the swing leg angle θ_s, and the torso angle θ_t. The velocity vector ω contains the planted leg angular velocity ω_p, the swing leg angular velocity ω_s, and the torso angular velocity ω_t. The input u is the input torque, where u_p is the torque applied on the planted leg from the ankle, u_s is the torque applied on the swing leg from the hip, and u_t is the torque applied on the torso from the hip. The continuous dynamics of $x = (\theta, \omega)$ are obtained from the Lagrangian method and are given by $\dot{\theta} = \omega, \dot{\omega} = D_f(\theta)^{-1}(-C_f(\theta,\omega)\omega - G_f(\theta) + Bu) =: \alpha(x,u)$, where D_f and C_f are the inertial and Coriolis matrices, respectively, and B is the actuator relationship matrix. In [48], the input torques that produce an acceleration a for a special state x are determined by a function μ, defined as $\mu(x,a) := B^{-1}(D_f(\theta)a + C_f(\theta,\omega)\omega + G_f(\theta))$. By applying $u = \mu(x,a)$ to $\dot{\omega} = \alpha(x,u)$, we obtain $\dot{\omega} = a$. Then, the flow map f is defined as*

$$f(x,a) := \begin{bmatrix} \omega \\ \\ a \end{bmatrix} \quad (x,a) \in C.$$

Flow is allowed when only one leg is in contact with the ground. To determine if the biped has reached the end of a step, we define $h(x) := \phi_s - \theta_p$ for all $x \in \mathbb{R}^6$ where ϕ_s denotes the step angle. The condition $h(x) \geq 0$ indicates that only one leg is in contact

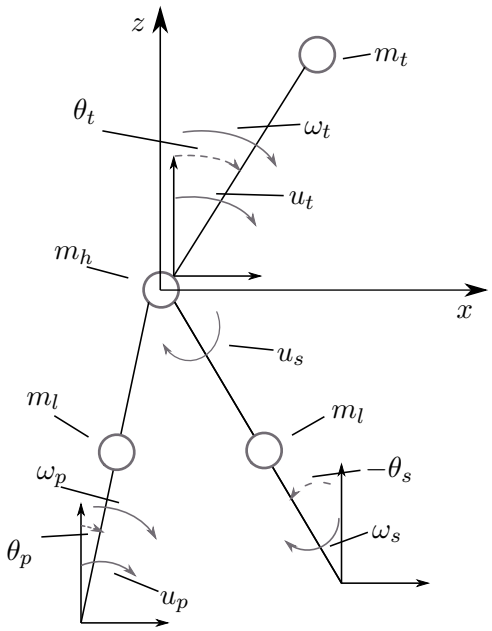


Figure 3.4: The biped system in Example 3.3. The angle vector θ contains the planted leg angle θ_p, the swing leg angle θ_s, and the torso angle θ_t. The velocity vector ω contains the planted leg angular velocity ω_p, the swing leg angular velocity ω_s, and the torso angular velocity ω_t. The input u is the input torque, where u_p is the torque applied on the planted leg from the ankle, u_s is the torque applied on the swing leg from the hip, and u_t is the torque applied on the torso from the hip.

with the ground. Thus, the flow set is defined as $C := \{(x, a) \in \mathbb{R}^6 \times \mathbb{R}^3 : h(x) \geq 0\}$. *Furthermore, a step occurs when the change of* h *is such that* θ_p *is approaching* ϕ_s, *and* h *equals zero. Thus, the jump set* D *is defined as* $D := \{(x, a) \in \mathbb{R}^6 \times \mathbb{R}^3 : h(x) = 0, \omega_p \geq 0\}$.

Following [19], when a step occurs, the swing leg becomes the planted leg, and the planted leg becomes the swing leg. The function Γ *is defined to swap angles and velocity variables as* $\theta^+ = \Gamma(\theta)$. *The angular velocities after a step are determined by a contact model denoted as* $\Omega(x) := (\Omega_p(x), \Omega_s(x), \Omega_t(x))$, *where* Ω_p, Ω_s, *and* Ω_t *are the angular velocity of the planted leg, swing leg, and torso, respectively. Then, the jump map* g *is defined as*

$$g(x, a) := \begin{bmatrix} \Gamma(\theta) \\ \\ \Omega(x) \end{bmatrix} \quad \forall (x, a) \in D. \tag{3.5}$$

For more information about the contact model, see [19, Appendix A].

An example of a motion planning problem for the walking robot system is as follows: using a bounded input signal, find a solution pair to (2.1) associated to the walking robot system completing a step of a walking circle. One way to characterize a walking cycle is to define the final state and the initial state as the states before and after a jump occurs, respectively. One such motion planning problem is given by defining the final state set as $X_f = \{(\phi_s, -\phi_s, 0, 0.1, 0.1, 0)\}$, *the initial state set as* $X_0 = \{x_0 \in \mathbb{R}^6 : x_0 = g(x_f, 0), x_f \in X_f\}$ *where the input argument of* g *can be set arbitrarily because input does not affect the value of* g; *see (3.5), and the unsafe set as* $X_u = \{(x, a) \in \mathbb{R}^6 \times \mathbb{R}^3 : a_1 \notin [a_1^{\min}, a_1^{\max}]$ *or* $a_2 \notin [a_2^{\min}, a_2^{\max}]$ *or* $a_3 \notin [a_3^{\min}, a_3^{\max}]$ *or* $(x, a) \in D\}$, *where* $a_1^{\min}$,

$a_2^{\min}$, and $a_3^{\min}$ are the lower bounds of a_1, a_2, and a_3, respectively, and $a_1^{\max}$, $a_2^{\max}$, and $a_3^{\max}$ are the upper bounds of a_1, a_2, and a_3, respectively. In this example, the step angle ϕ_s is set as 0.70. The length of the torso l_t and the legs l_l are set as 1. The leg mass, hip mass and torso mass are set as 1. The walking velocity is set as 0.6. $a_1^{\min}$ and $a_1^{\max}$ are set as -3 and 3, respectively. $a_2^{\min}$ and $a_2^{\max}$ are set as -3 and 3, respectively. $a_3^{\min}$ and $a_3^{\max}$ are set as -0.2 and 0.2, respectively. We also solve this motion planning problem later.

3.2 Reversal, Concatenation, and Truncation of Solutions to Hybrid Dynamical Systems

The operations related to our algorithm template include reversal and concatenation operations of solution pairs.

Definition 3.4 (Reversal of a solution pair)**.** *Given a compact solution pair (ϕ, υ) to $\mathcal{H} = (C, f, D, g)$, where $\phi : dom\ \phi \to \mathbb{R}^n$, $\upsilon : dom\ \upsilon \to \mathbb{R}^m$, and $(T, J) = \max dom\ (\phi, \upsilon)$, the pair (ϕ', υ') is the reversal of (ϕ, υ), where $\phi' : dom\ \phi' \to \mathbb{R}^n$ with $dom\ \phi' \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ and $\upsilon' : dom\ \upsilon' \to \mathbb{R}^m$ with $dom\ \upsilon' = dom\ \phi'$, if the following hold:*

1) The function ϕ' is defined as

a) $dom\ \phi' = \{(T, J)\} - dom\ \phi$, where the minus sign denotes Minkowski difference;

b) $\phi'(t, j) = \phi(T - t, J - j)$ for all $(t, j) \in dom\ \phi'$.

2) The function υ' is defined as

a) dom $\upsilon' = \{(T, J)\} -$ dom υ, where the minus sign denotes Minkowski difference.

b) For all $j \in \mathbb{N}$ such that $I^j = \{t : (t, j) \in$ dom $\upsilon'\}$ has nonempty interior,

i) For all $t \in \operatorname{int} I^j$, $\upsilon'(t, j) = \upsilon(T - t, J - j)$;

ii) If I^0 has nonempty interior, then $\upsilon'(0, 0) \in \mathbb{R}^m$ is such that $(\phi'(0, 0), \upsilon'(0, 0)) \in \overline{C}$;

iii) For all $t \in \partial I^j$ such that $(t, j + 1) \notin$ dom υ' and $(t, j) \neq (0, 0)$, $\upsilon'(t, j) \in \mathbb{R}^m$.

c) For all $(t, j) \in$ dom υ' such that $(t, j+1) \in$ dom υ', $\upsilon'(t, j) = \upsilon(T-t, J-j-1)$.

The reversal of a given solution pair is not always unique, since $\upsilon'(t, j) \in \mathbb{R}^m$ in item 2biii in Definition 3.4 is free. The value assigned to υ' at such hybrid time can be given by the continuous extension of υ' at (t, j) in the interval I^j, if it exists.

Definition 3.5 (Concatenation operation)**.** *Given two functions $\phi_1 :$ dom $\phi_1 \to \mathbb{R}^n$ and $\phi_2 :$ dom $\phi_2 \to \mathbb{R}^n$, where dom ϕ_1 and dom ϕ_2 are hybrid time domains, ϕ_2 can be concatenated to ϕ_1 if ϕ_1 is compact and $\phi :$ dom $\phi \to \mathbb{R}^n$ is the concatenation of ϕ_2 to ϕ_1, denoted $\phi = \phi_1|\phi_2$, namely,*

1) dom $\phi =$ dom $\phi_1 \cup$ (dom $\phi_2 + \{(T, J)\}$), where $(T, J) = \max$ dom ϕ_1 and the plus sign denotes Minkowski addition;

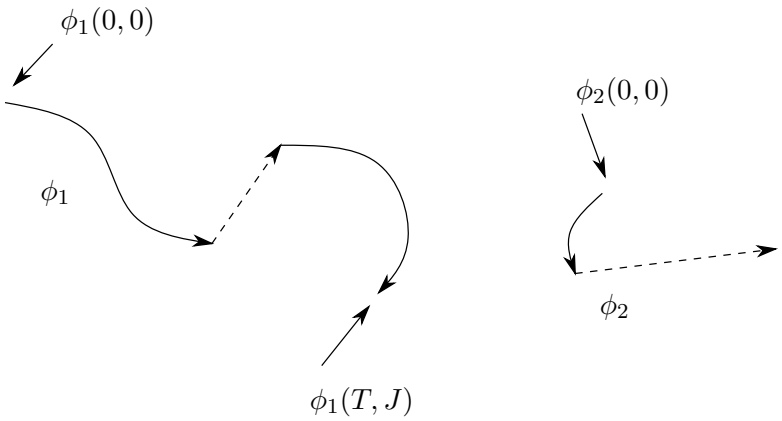
2) $\phi(t,j) = \phi_1(t,j)$ *for all* $(t,j) \in dom\ \phi_1 \backslash \{(T,J)\}$ *and* $\phi(t,j) = \phi_2(t-T, j-J)$ *for all* $(t,j) \in dom\ \phi_2 + \{(T,J)\}$.

The following example in Figure 3.5 illustrates the concatenation of the functions on hybrid time domains. Let ϕ_1 and ϕ_2 be two functions defined on hybrid time domains and $(T,J) = \max \operatorname{dom} \phi_1$. When ϕ_2 is concatenated to ϕ_1, the domain of the concatenation is constructed by translating the domain of ϕ_2 by (T,J) and concatenating the translated domain to the domain of ϕ_1. Particularly, if ϕ_1 and ϕ_2 are hybrid arcs, their concatenation is not guaranteed to be a hybrid arc. The reason is that the concatenation result may not be continuous at (T,J). The concatenation of two trajectories is illustrated in Figure 3.5. In the figure, the solid lines denote the flows and the dotted lines denote the jumps. In Figure 3.5(a), the trajectories ϕ_1 and ϕ_2 denote the original trajectories and the trajectory ϕ in Figure 3.5(b) denotes the concatenation of ϕ_2 to ϕ_1.

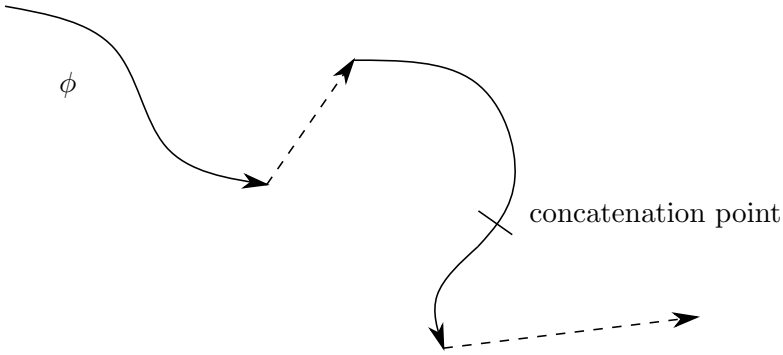
The truncation operation of the solution pair is used in the proofs of the main results. The definition of the truncation operation is given as follows.

Definition 3.6 (Truncation and translation operation)**.** *Given a function* $\phi : dom\ \phi \to \mathbb{R}^n$*, where* $dom\ \phi$ *is a hybrid time domain, and pairs of hybrid time* $(T_1, J_1) \in dom\ \phi$ *and* $(T_2, J_2) \in dom\ \phi$ *such that* $T_1 \leq T_2$ *and* $J_1 \leq J_2$*, the function* $\widetilde{\phi} : dom\ \widetilde{\phi} \to \mathbb{R}^n$ *is the truncation of* ϕ *between* (T_1, J_1) *and* (T_2, J_2) *and translation by* (T_1, J_1) *if*

1) $dom\ \widetilde{\phi} = (dom\ \phi \cap ([T_1, T_2] \times \{J_1, J_1 + 1, ..., J_2\})) - \{(T_1, J_1)\}$*, where the minus sign denotes Minkowski difference;*



(a) The original trajectory ϕ_1 and ϕ_2.



(b) The trajectory ϕ resulting from concatenating ϕ_2 to ϕ_1.

Figure 3.5: The concatenation ϕ of trajectory ϕ_2 to trajectory ϕ_1.

2) $\widetilde{\phi}(t,j) = \phi(t + T_1, j + J_1)$ *for all* $(t,j) \in dom\ \widetilde{\phi}$.

3.3 A Forward/Backward Propagation Algorithm Template

In this section, a bidirectional propagation algorithm template, which we refer to as forward/backward propagation algorithm template, is proposed to help design motion planning algorithms to solve the motion planning problem for hybrid dynamical systems $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$. The main steps of the forward/backward propagation algorithm template are as follows.

Step 1: Propagate the state from the initial state set X_0 forward in hybrid time and from the final state set X_f backward in hybrid time without intersecting with the unsafe set X_u.

Step 2: If an appropriate overlap between forward and backward propagation results is found, reverse the solution pair generated by the backward-in-time propagation, concatenate it to the solution pair generated by forward-in-time propagation and return the concatenation result.

First, we construct the backward-in-time hybrid dynamical system, which is used to implement the backward propagation. Then, we formalize the propagation operation used in the algorithm template. After that, we show that the result constructed by the propagation, reversal, and concatenation operations satisfies the requirements in Problem 3.1. Finally, we present the template framework and analyze its soundness and exactness properties.

3.3.1 Backward-in-time Hybrid Dynamical Systems

In the proposed template, a backward-in-time version of the hybrid dynamical system $\mathcal{H} = (C, f, D, g)$, denoted $\mathcal{H}^{\text{bw}} = (C^{\text{bw}}, f^{\text{bw}}, D^{\text{bw}}, g^{\text{bw}})$, is required when propagating trajectories backward in hybrid time. The backward-in-time hybrid dynamical system $\mathcal{H}^{\text{bw}}$ is constructed as follows.

Definition 3.7 (Backward-in-time hybrid dynamical system)**.** *Given a hybrid dynamical system $\mathcal{H} = (C, f, D, g)$, the backward-in-time hybrid dynamical system of $\mathcal{H}$, denoted $\mathcal{H}^{bw}$, is the hybrid dynamical system*

$$\mathcal{H} : \begin{cases} \dot{x} = f^{bw}(x,u) & (x,u) \in C^{bw} \\ x^+ \in g^{bw}(x,u) & (x,u) \in D^{bw} \end{cases} \tag{3.6}$$

where

1) The backward-in-time flow set is constructed as

$$C^{bw} := C. \tag{3.7}$$

2) The backward-in-time flow map is constructed as

$$f^{bw}(x,u) := -f(x,u) \quad \forall (x,u) \in C^{bw}. \tag{3.8}$$

3) The backward-in-time jump map is constructed as

$$g^{bw}(x,u) := \{z \in \mathbb{R}^n : \ x = g(z,u), (z,u) \in D\} \quad \forall (x,u) \in \mathbb{R}^n \times \mathbb{R}^m. \tag{3.9}$$

4) The backward-in-time jump set is constructed as

$$D^{bw} := \{(x,u) \in \mathbb{R}^n \times \mathbb{R}^m : \exists z \in g^{bw}(x,u) : (z,u) \in D\}. \tag{3.10}$$

The above shows a general method of constructing the backward-in-time system given the forward-in-time hybrid dynamical system. Note that although the jump map g in the forward-in-time system $\mathcal{H}$ is a single-valued map, the jump map g^{bw} in the backward-in-time system $\mathcal{H}^{\mathrm{bw}}$ is not necessarily single valued. Hence, a difference inclusion is used in (3.6). The backward-in-time hybrid dynamical system of the hybrid dynamical system in Example 3.2 is constructed next.

Example 3.8 (Actuated bouncing ball system in Example 3.2, revisited)**.** *The backward-in-time hybrid dynamical system of the actuated bouncing ball system is constructed as follows.*

- *From (3.7), the backward-in-time flow set C^{bw} is given by*

$$C^{bw} := C = \{(x, u) \in \mathbb{R}^2 \times \mathbb{R} : x_1 \geq 0\}. \tag{3.11}$$

- *From (3.8), the backward-in-time flow map f^{bw} is given by*

$$f^{bw}(x, u) := -f(x, u) = \begin{bmatrix} -x_2 \\ \\ \gamma \end{bmatrix} \qquad (x, u) \in C^{bw}. \tag{3.12}$$

- *From (3.10), the backward-in-time jump set is given by*

$$D^{bw} := \{(x, u) \in \mathbb{R}^2 \times \mathbb{R} : x_1 = 0, x_2 \geq u, u \geq 0\}. \tag{3.13}$$

- *From (3.9), the backward-in-time jump map g^{bw} is given by*

$$g^{bw}(x, u) := \begin{bmatrix} x_1 \\ \\ -\frac{x_2}{\lambda} + \frac{u}{\lambda} \end{bmatrix} \qquad (x, u) \in D^{bw}. \tag{3.14}$$

In conclusion, the backward-in-time hybrid dynamical system is given by:

$$\mathcal{H}^{bw}: \begin{cases} \dot{x} = f^{bw}(x,u) & (x,u) \in C^{bw} \\ x^{+} = g^{bw}(x,u) & (x,u) \in D^{bw} \end{cases} \tag{3.15}$$

where the flow map f is given in (4.18), the flow set C is given in (4.17), the jump map g is given in (4.20) and the jump set D is given in (4.19).

Proposition 3.9 shows that the reversal of the solution pair to a hybrid dynamical system is a solution pair to its backward-in-time hybrid dynamical system.

Proposition 3.9. *Given a hybrid dynamical system $\mathcal{H}$ and its backward-in-time system $\mathcal{H}^{bw}$, if $\psi = (\phi, u)$ is a compact solution pair to $\mathcal{H}$, the reversal $\psi' = (\phi', u')$ of $\psi = (\phi, u)$ is a compact solution pair to $\mathcal{H}^{bw}$.*

Proof. This proof is in Appendix A.1. □

3.3.2 Propagation Operation

The proposed template requires an operation to collect all solution pairs that start from the initial/final state set, without reaching the unsafe set. The definition of the propagation operation is formalized as follows.

Definition 3.10 (Propagation operation)**.** *Given a hybrid dynamical system $\mathcal{H} = (C, f, D, g)$, an initial state set X_0, and an unsafe set X_u, the propagation operation with parameters $(X_0, X_u, (C, f, D, g))$ constructs a set $\mathcal{S}$ that collects all the solution pairs $\psi = (\phi, u)$ that satisfy the following conditions:*

1. *$\psi = (\phi, u)$ is a solution pair to $\mathcal{H}$ (see Definition 2.5);*

2. $\phi(0,0) \in X_0$*;*

3. $\psi(t,j) \notin X_u$ *for all* $(t,j) \in dom\ \psi$*.*

Remark 3.11. *The result of the forward propagation operation is defined as the set of all possible collision-free solution pairs to the hybrid dynamical system starting from the given initial state set. The propagation operation with parameters* $(X_f, X_u, (C^{bw}, f^{bw}, D^{bw}, g^{bw}))$ *corresponds to the backward propagation starting from the final state set.* ▲

Remark 3.12. *Note that Definition 3.10 describes an exact searching process since it constructs a set that includes all possible solution pairs. The implementation of an algorithm constructing such set is difficult in practice because there can be infinite many possible values for the input at each hybrid time instance. Additionally, the solution pairs collected in* $\mathcal{S}$ *are not limited to maximal solutions, which means that any truncation of a solution pair in* $\mathcal{S}$ *should also be collected in* $\mathcal{S}$*. However, this operation can be implemented incrementally and numerically. For example, the search trees constructed in the classic motion planning algorithm, such as RRT, can be seen as an approximation of the set* $\mathcal{S}$ *because any path in the search trees starting from* X_0 *can be seen as an element in* $\mathcal{S}$*.* ▲

3.3.3 Construction of Motion Plans

In the proposed template, reversal and concatenation operations are executed to build motion plans. The two operations are formalized in Definition 3.4 and Definition 3.5. Proposition 3.9 has shown that the reversal of a solution pair to a hybrid dynamical

system is a solution pair to its backward-in-time system. Proposition 3.13 below shows that the concatenation of solution pairs satisfies the definition of the solution pair in Definition 2.5 under mild conditions.

Proposition 3.13. *Given two solution pairs $\psi_1 = (\phi_1, \upsilon_1)$ and $\psi_2 = (\phi_2, \upsilon_2)$ to a hybrid dynamical system $\mathcal{H}$, their concatenation $\psi = (\phi, \upsilon) = (\phi_1|\phi_2, \upsilon_1|\upsilon_2)$, denoted $\psi = \psi_1|\psi_2$, is a solution pair to $\mathcal{H}$ if the following hold:*

1) $\psi_1 = (\phi_1, \upsilon_1)$ is compact;

2) $\phi_1(T, J) = \phi_2(0, 0)$, where $(T, J) = \max dom\ \psi_1$;

3) If both $I^J_{\psi_1}$ and $I^0_{\psi_2}$ have nonempty interior, where $I^j_\psi = \{t : (t, j) \in dom\ \psi\}$ and $(T, J) = \max dom\ \psi_1$, then $\psi_2(0, 0) \in C$.

Proof. This proof is in Appendix A.2. □

Remark 3.14. *Item 1 in Proposition 3.13 guarantees that ψ_2 can be concatenated to ψ_1. Definition 3.5 suggests that if ψ_2 can be concatenated to ψ_1, ψ_1 is required to be compact. Item 2 in Proposition 3.13 guarantees that the ϕ is a hybrid arc or ψ satisfies item 3 in Definition 2.5 at hybrid time (T, J), where $(T, J) = \max dom\ \psi_1$. Item 3 in Proposition 3.13 guarantees that the concatenation result ψ satisfies item 2 in Definition 2.5 at hybrid time (T, J). Note that item 2 therein does not require that $\psi_1(T, J) \in C$ and $\psi_2(0, 0) \in C$ since $T \notin \operatorname{int} I^J_{\psi_1}$ and $0 \notin \operatorname{int} I^0_{\psi_2}$. However, T may belong to the interior of I^J_ψ after concatenation. Hence, item 3 guarantees that if T belongs to the interior of I^J_ψ after concatenation, then $\psi(T, J) \in C$.* ▲

The following assumption is imposed on the solution pairs that are used to construct motion plans in the forthcoming forward/ backward propagation algorithm template.

Assumption 3.15. *Given a solution pair $\psi_1 = (\phi_1, \upsilon_1)$ to a hybrid dynamical system $\mathcal{H} = (C, f, D, g)$ and a solution pair $\psi_2 = (\phi_2, \upsilon_2)$ to the backward-in-time hybrid dynamical system $\mathcal{H}^{bw}$ associated to $\mathcal{H}$, the following hold:*

1) ψ_1 and ψ_2 are compact;

2) $\phi_1(T_1, J_1) = \phi_2(T_2, J_2)$, where $(T_1, J_1) = \max dom\ \psi_1$ and $(T_2, J_2) = \max dom\ \psi_2$;

3) If both $I_{\psi_1}^{J_1}$ and $I_{\psi_2}^{J_2}$ have nonempty interior, where $I_{\psi}^{j} = \{t : (t, j) \in dom\ \psi\}$, $(T_1, J_1) = \max dom\ \psi_1$, and $(T_2, J_2) = \max\ dom\ \psi_2$, then $\psi_2(T_2, J_2) \in C$.

Remark 3.16. *Given a hybrid dynamical system $\mathcal{H}$, its backward-in-time system $\mathcal{H}^{bw}$, a solution pair ψ_1 to $\mathcal{H}$, and a solution pair ψ_2 to $\mathcal{H}^{bw}$, Assumption 3.15 is imposed on ψ_1 and ψ_2 to guarantee that the concatenation of the reversal of ψ_2 to ψ_1 is a solution pair to $\mathcal{H}$. Assumption 3.15 guarantees that the conditions needed to apply Proposition 3.9 and Proposition 3.13 hold. Note that conditions that guarantee the existence of nontrivial solutions have been proposed in Proposition 3.4 in [9]. For the propagation with parameters $(X_0, X_u, (C, f, D, g))$, if $\xi \in X_0$ is such that[1] $\xi \in \Pi(D)$ or there exist $\epsilon > 0$, an absolutely continuous function $z : [0, \epsilon] \to \mathbb{R}^n$ with $z(0) = \xi$, and a Lebesgue measurable and locally essentially bounded function[2] $\widetilde{\upsilon} : [0, \epsilon] \to U_C$ such*

[1]Given a jump set $D \subset \mathbb{R}^n \times \mathbb{R}^m$, define $\Pi(D) := \{x \in \mathbb{R}^n : \exists u \in U_D, \text{s.t. } (x, u) \in D\}$ as the projection of D onto $\mathbb{R}^n$, where $U_D := \{u \in \mathbb{R}^m : \exists x \in \mathbb{R}^n \text{ such that } (x, u) \in D\}$

[2]Given a flow set $C \subset \mathbb{R}^n \times \mathbb{R}^m$, the set denoted U_C, includes all possible input values that can be applied during flows and is defined as $U_C := \{u \in \mathbb{R}^n : \exists x \in \mathbb{R}^n \text{ such that } (x, u) \in C\}$.

that $(z(t), \widetilde{\upsilon}(t)) \in C$ *for all* $t \in (0, \epsilon)$ *and* $\frac{d}{dt}z(t) = f(z(t), \widetilde{\upsilon}(t))$ *for almost all* $t \in [0, \epsilon]$, *where*[3] $\widetilde{\upsilon}(t) \in \Psi_c^u(z(t))$ *for every* $t \in [0, \epsilon]$, *then the existence of nontrivial solution pairs is guaranteed from* ξ. *Items 2 and 3 in Assumption 3.15 relate the final states and their "last" interval of flow of the given solution pairs.* ▲

The following result validates that a solution pair constructed using solution pairs satisfying Assumption 3.15 satisfies Definition 2.5.

Lemma 3.17. *Given a hybrid dynamical system* $\mathcal{H}$ *and its backward-in-time hybrid dynamical system* $\mathcal{H}^{bw}$, *if* ψ_1 *is a solution pair to* $\mathcal{H}$ *and* ψ_2 *is a solution pair to* $\mathcal{H}^{bw}$ *such that* ψ_1 *and* ψ_2 *satisfy Assumption 3.15, then the concatenation* $\psi = \psi_1 | \psi_2'$ *is a solution pair to* $\mathcal{H}$, *where* ψ_2' *is the reversal of* ψ_2.

Proof. According to Definition 3.4, the reversal operation is executable on ψ_2 when ψ_2 is compact. The first condition in Assumption 3.15 guarantees that ψ_2 is compact. According to Proposition 3.9, the reversal ψ_2' of ψ_2 is a solution pair to $\mathcal{H}$.

Next, we show that the conditions in Proposition 3.13 are satisfied.

1) The first condition in Assumption 3.15 shows that ψ_1 is compact. Therefore, the first condition in Proposition 3.13 is satisfied.

2) Note that the second condition in Assumption 3.15 suggests that $\phi_1(T_1, J_1) = \phi_2(T_2, J_2)$. Since $\phi_2'(0,0) = \phi_2(T_2, J_2)$, then $\phi_1(T_1, J_1) = \phi_2'(0,0)$ is satisfied. Hence, ψ_1 and ψ_2' satisfy the second condition in Proposition 3.13.

[3]Given a flow set $C \subset \mathbb{R}^n \times \mathbb{R}^m$, the set-valued maps $\Psi_c^u : \mathbb{R}^n \to U_C$ is defined for each $x \in \mathbb{R}^n$ as $\Psi_c^u(x) := \{u \in U_C : (x, u) \in C\}$.

3) Since $I_{\psi_2}^{J_2} = I_{\psi_2'}^{0}$ and $\psi_2(T_2, J_2) = \psi_2'(0, 0)$, then the third condition in Assumption 3.15 implies that ψ_1 and ψ_2' satisfy the third condition in Proposition 3.13.

Since the conditions in Proposition 3.13 are satisfied, Proposition 3.13 guarantees that the concatenation of ψ_2' to ψ_1 is a solution pair to $\mathcal{H}$. □

Lemma 3.17 is exploited by our forthcoming algorithm template, which also checks numerically if Assumption 3.15 holds.

3.3.4 Forward/Backward Propagation Algorithm Template and Property Analysis

With all the components introduced above, the forward/backward propagation algorithm is given in Algorithm 1. The inputs of the proposed algorithm are given as follows.

- The motion planning problem is given as $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$ (defined in Problem 3.1).
- The backward-in-time hybrid dynamical system of the system $\mathcal{H} = (C, f, D, g)$, denoted as $\mathcal{H}^{\text{bw}}$ and constructed following (3.6).
- Sets of solution pairs $\mathcal{S}^{\text{fw}}$ and $\mathcal{S}^{\text{bw}}$ to store forward and backward propagation operation results, respectively.

The algorithm template collects all collision-free solution pairs to $\mathcal{H}$ ($\mathcal{H}^{\text{bw}}$) starting from X_0 (X_f) and stores them in the set $\mathcal{S}^{\mathbf{fw}}$ ($\mathcal{S}^{\mathbf{bw}}$, respectively) (Line 1). Then, the

Algorithm 1 Forward/backward propagation algorithm template

1: Propagate with parameters $(X_0, X_u, (C, f, D, g))$ (see Definition 3.10) and propagate with parameters $(X_f, X_u, (C^{\mathrm{bw}}, f^{\mathrm{bw}}, D^{\mathrm{bw}}, g^{\mathrm{bw}}))$ (see Definition 3.10), with respective results stored in $\mathcal{S}^{\mathrm{fw}}$ and $\mathcal{S}^{\mathrm{bw}}$.
2: **if** $\exists \psi^{\mathrm{fw}} \in \mathcal{S}^{\mathrm{fw}}$ and $\psi^{\mathrm{bw}} \in \mathcal{S}^{\mathrm{bw}}$ that satisfy Assumption 3.15 **then**
3: Reverse ψ^{bw} to get $\psi^{\mathrm{bw}'}$ (see Definition 3.4).
4: Concatenate $\psi^{\mathrm{bw}'}$ to ψ^{fw} to get ψ (see Definition 3.5).
5: **return** ψ.
6: **else**
7: **return** Failure.
8: **end if**

algorithm template searches for two solution pairs in the sets $\mathcal{S}^{\mathbf{fw}}$ and $\mathcal{S}^{\mathbf{bw}}$, respectively, that satisfy Assumption 3.15 (Line 2). If such two solution pairs are found, then the algorithm reverses the solution pair found in $\mathcal{S}^{\mathbf{bw}}$, concatenates the reversal result to the solution pair found in $\mathcal{S}^{\mathbf{fw}}$, and returns the concatenation result (Line 3 - 5). If such two solution pairs are not found, then the algorithm template returns a failure signal (Line 7).

Soundness and exactness are important properties of any planning algorithm. Before discussing soundness and completeness of the algorithm, Proposition 3.19 shows that the truncation of a solution pair of a given hybrid dynamical system following a translation remains a solution pair to the same hybrid dynamical system, provided that Assumption 3.18 is satisfied. Assumption 3.18 is imposed on the hybrid time at

which the solution pair is truncated to guarantee that the initial state-input pair of the truncation result belongs to $\overline{C} \cup D$, as required in Definition 2.5.

Assumption 3.18. *Given a function ψ defined on a hybrid time domain and a hybrid time $(T, J) \in \text{dom } \psi$, one of the following is satisfied:*

1) $T = 0$, $J = 0$;

2) $T \in \text{int } I_\psi^J$, where I_ψ^J has nonempty interior; or

3) If $(T, J) \neq (0, 0)$, $(T, J) \in \text{dom } \psi$ and $(T, J + 1) \in \text{dom } \psi$.

Proposition 3.19. *Given a solution pair $\psi = (\phi, \upsilon)$ to a hybrid dynamical system $\mathcal{H}$ and a pair of hybrid times $(T_1, J_1) \in \text{dom } \psi$ and $(T_2, J_2) \in \text{dom } \psi$ such that*

1) $T_1 \leq T_2$, $J_1 \leq J_2$;

2) ψ and (T_1, J_1) satisfy Assumption 3.18,

then $\widetilde{\psi} = (\widetilde{\phi}, \widetilde{\upsilon})$ is a solution pair to $\mathcal{H}$, where $(\widetilde{\phi}, \widetilde{\upsilon})$ is the truncation of (ϕ, υ) between (T_1, J_1) and (T_2, J_2) following translation by (T_1, J_1).

Proof. This proof is in Appendix A.3. □

Soundness guarantees the correctness of the returned solution, as defined next.

Definition 3.20 (Soundness [13])**.** *An algorithm is said to be sound for a problem if for any possible data defining the problem (e.g., $(X_0, X_f, X_u, (C, f, D, g))$ in Problem 3.1), the result returned by the algorithm for the problem is always a correct solution to it.*

The next result shows that the proposed forward/backward propagation algorithm template is sound for the motion planning problem in Problem 3.1.

Theorem 3.21. *The forward/backward propagation algorithm in Algorithm 1 is sound for the motion planning problem $\mathcal{P}$ in Problem 3.1.*

Proof. This proof is divided into two cases:

1) When a non-failure result is returned, we show that the result is a motion plan to the given motion planning problem;

2) When a failure result is returned, we show that there does not exist a motion plan to the given motion planning porblem.

For case 1), Lemma 3.17 shows that the returned result is a solution pair to the given hybrid dynamical system. Note that in Algorithm 1, the returned result is constructed by concatenating $\psi^{\mathrm{bw}'}$ to ψ^{fw}, where ψ^{fw} and ψ^{bw} are from propagatgion operations with parameters $(X_0, X_u, (C, f, D, g))$ and $(X_f, X_u, (C^{\mathrm{bw}}, f^{\mathrm{bw}}, D^{\mathrm{bw}}, g^{\mathrm{bw}}))$, respectively. Therefore, the constructed solution starts from X_0, ends in X_f, and does not intersect with X_u, due to Definition 3.10, Definition 3.4, and Definition 3.5.

For case 2), we proceed by contradiction. Assume that there exists a motion plan ψ when a failure result is returned. Then, there exist $\psi^{\mathrm{fw}} \in \mathcal{S}^{\mathrm{fw}}$ and $\psi^{\mathrm{bw}} \in \mathcal{S}^{\mathrm{bw}}$ that satisfy Assumption 3.15 in line 2 in Algorithm 1 and a motion plan, rather than a failure signal, is returned. The existence of ψ^{fw} and ψ^{bw} can be proved by truncating ψ into two pieces and showing the two pieces belong to the sets $\mathcal{S}^{\mathrm{fw}}$ and $\mathcal{S}^{\mathrm{bw}}$, respectively, according to Proposition 3.19. □

Exactness guarantees that the algorithm will always return a solution when one exists, as defined next.

Definition 3.22 (Exactness [18])**.** *An algorithm is said to be exact for a problem if for any possible data defining the problem (e.g.,* $(X_0, X_f, X_u, (C, f, D, g))$ *in Problem 3.1), it finds a solution when one exists.*

The next result shows that the proposed forward/backward propagation algorithm template is exact for the motion planning problem in Problem 3.1.

Theorem 3.23. *The forward/backward propagation algorithm in Algorithm 1 is an exact algorithm for the motion planning problem* $\mathcal{P}$ *in Problem 3.1.*

Proof. If a solution ψ to the given motion planning problem exists, then we can truncate ψ into ψ_1 and ψ_2, and reverse ψ_2 to get ψ_2'. Note that ψ_1 should be collected in $\mathcal{S}^{\text{fw}}$ and ψ_2' should be collected in $\mathcal{S}^{\text{bw}}$ according the settings of propagation operations and Proposition 3.19. Then, it follows that ψ_1 and ψ_2' satisfy Assumption 3.15. Hence, the condition in line 2 in Algorithm 1 is satisfied and ψ is constructed and returned. Therefore, if a motion plan exists, Algorithm 1 is able to find and return it. □

Remark 3.24. *The exactness property guarantees that the proposed algorithm returns a solution when one exists. However, when no solution exists, the proposed algorithm is not guaranteed to return a failure signal in finite time because the propagation operations are not always guaranteed to be completed within finite time. This is not a unique problem of our algorithm template, but rather a general issue in motion planning algorithms; see [25] [28] [44].* ▲

3.4 Time Complexity Analysis

Let n_f be the number of input signals in $\mathcal{U}_C$ and n_g be the number of input values in $\mathcal{U}_D$. Assume there exists a solution $\psi = (\phi, \upsilon)$ to the given motion planning problem that can be generated by applying the input signals/values in $\mathcal{U}$. Let $(T, J) = \max \operatorname{dom} \psi$. Note that each possible input is applied to each existing vertex in the search graph. Therefore, at most $n_f + n_j$ input signals/values are applied. When $n_f + n_j$ input signals/values are applied, at most $n_f + n_j$ new vertices are generated. Then if a motion plan is not found, at most $n_f + n_j$ input signals/values will be applied to each newly generated vertices. Therefore, the number of vertices grows exponentially during the search. Hence, since the proposed algorithm propagates both forward and backward in hybrid time, the time complexity of the proposed implementation is $O((n_f + n_j)^{(T/t_{\min}+J)/2})$, where $t_{\min}$[4] denotes the minimal time duration of the input signals in $\mathcal{U}_C$.

3.5 Software Tool for Motion Planning for Hybrid Dynamical Systems

Following the proposed algorithm template, a motion planning algorithm is implemented leading to a software tool to solve motion planning problems for hybrid systems. In this algorithm, the propagation operation is implemented by incrementally constructing the search tree using the well-known Breadth-First-Search (BFS) strategy.

[4] $t_{\min} = \min_{\widetilde{\upsilon} \in \mathcal{U}_C} \overline{t}(\widetilde{\upsilon})$ where given $\widetilde{\upsilon} \in \mathcal{U}_C$, the functional $\overline{t} : \mathcal{U}_C \to [0, \infty)$ returns the time duration of $\widetilde{\upsilon}$. Namely, given any $\widetilde{\upsilon} : [0, t^*] \to U_C$, $\overline{t}(\widetilde{\upsilon}) = t^*$.

This tool only requires the motion planning problem data $(X_0, X_f, X_u, (C, f, D, g))$. The data of the backward-in-time system $(C^{\text{bw}}, f^{\text{bw}}, D^{\text{bw}}, g^{\text{bw}})$ can be either filled in automatically, or imported from input files. Note that item 2 in Assumption 3.15 requires equality between $\phi_1(T_1, J_1)$ and $\phi_2(T_2, J_2)$. Given $\epsilon > 0$ representing the tolerance for the satisfaction of this condition, we implement item 2 in Assumption 3.15 as

$$|\phi_1(T_1, J_1) - \phi_2(T_2, J_2)| \leq \epsilon. \tag{3.16}$$

Next, the proposed algorithm and this tool are illustrated in the bouncing ball in Example 3.2 and a walking robot.

Example 3.25 (Actuated bouncing ball system in Example 3.2, revisited)**.** *This part illustrates that the proposed implementation is able to solve the sample motion planning problem in Example 3.2. The inputs fed to the proposed implementation are given as follows.*

- *The backward-in-time system $\mathcal{H}^{bw}$ of the actuated bouncing ball system is given in Example 3.8.*
- *The tolerance ϵ in (4.11) is set to* 0.2*.*

The simulation result is shown in Figure 3.6. The simulation is implemented in MATLAB software and processed by a 2.2 *GHz Quad-Core Intel Core i7 processor. The simulation takes* 0.65 *seconds to complete.*

Example 3.26 (Walking robot in Example 3.3, revisited)**.** *Inputs fed to the proposed implementation are given as follows.*

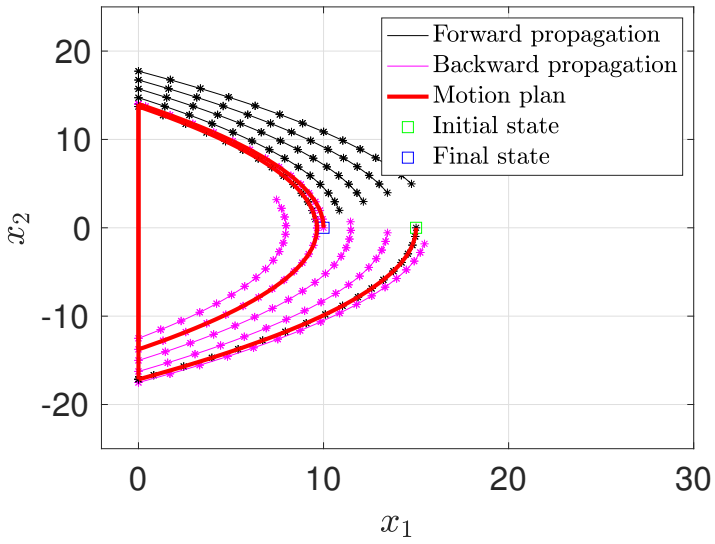


Figure 3.6: The above shows search graphs constructed for the sample motion planning problem in Example 3.2. The states represented by vertices in the search graphs are denoted by $*$. The lines between $*$'s denote the state trajectories of the solution pairs represented by edges in the search graphs.

- *The backward-in-time model of the walking robot is filled automatically.*

- *The tolerance ϵ in (4.11) is set to* 0.3.

This simulation is processed on an HPC cluster [5]*. Simulation results are shown in Figure 4.4. Table 3.1 shows that when a smaller ϵ in (4.11) is applied, the algorithm takes more time to find the solution. Tables 3.2 and 3.3 show that when input signals of shorter time duration are applied, or when the input library includes more input signals, the algorithm generates more vertices and takes more time to find the motion plan. It is consistent with our time complexity analysis in Section 3.4.*

[5] Configuration information available at https://hummingbird.ucsc.edu.

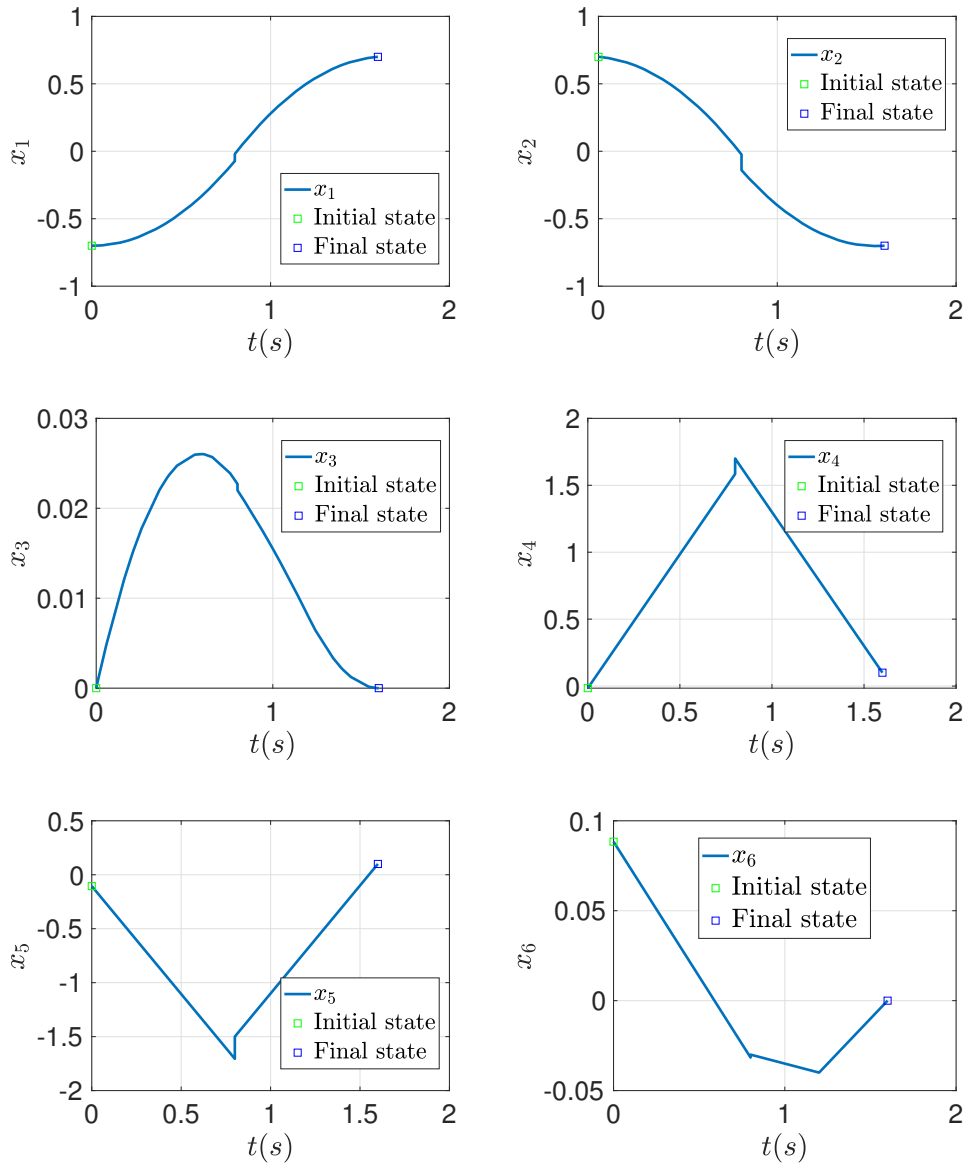


Figure 3.7: Trajectories of each state components of the generated motion plan for the walking robot.

ϵ	Time (seconds)	Vertices
0.3	1608.2	3796
0.5	300.1	511
0.8	234.3	200

Table 3.1: The time consumption/number of generated vertices using different tolerance ϵ in (4.11).

Input signals	Time (minutes)	Vertices
125	26.8	3796
1000	55.9	3084
3375	185.7	3108

Table 3.2: The time consumption/number of generated vertices when using different number of the input signals.

τ (seconds)	Time (minutes)	Vertices
0.2	511.9	6452
0.3	178.7	6696
0.5	26.8	3796
1.0	1.1	193

Table 3.3: The time consumption/number of generated vertices when using input signals of different time duration τ.

Chapter 4

Sampling-based Feasible Motion Planning for Hybrid Dynamical Systems

In this chapter, we introduce HyRRT, an RRT-type motion planning algorithm for hybrid dynamical systems, designed to solve Problem 3.1. HyRRT is a one-directional sampling-based implementation of the algorithm template in Algorithm 1, which approximates the propagation operation outlined in Definition 3.10 by incrementally constructing a search tree.

4.1 Overview

HyRRT searches for a motion plan by incrementally constructing a search tree. The search tree is modeled by a directed tree. A directed tree $\mathcal{T}$ is a pair $\mathcal{T} = (V, E)$, where V is a set whose elements are called vertices and E is a set of paired vertices whose elements are called edges. The edges in the directed tree are directed, which means

the pairs of vertices that represent edges are ordered. The set of edges E is defined as $E \subseteq \{(v_1, v_2) : v_1 \in V, v_2 \in V, v_1 \neq v_2\}$. The edge $e = (v_1, v_2) \in E$ represents an edge from v_1 to v_2. A path in $\mathcal{T} = (V, E)$ is a sequence of vertices $p = (v_1, v_2, ..., v_k)$ such that $(v_i, v_{i+1}) \in E$ for all $i = 1, 2, ..., k-1$.

Each vertex in the search tree $\mathcal{T}$ is associated with a state value of $\mathcal{H}$. Each edge in the search tree is associated with a solution pair to $\mathcal{H}$ that connects the state values associated with their endpoint vertices. The state value associated with vertex $v \in V$ is denoted as $\overline{x}_v$ and the solution pair associated with edge $e \in E$ is denoted as $\overline{\psi}_e$, as shown in Figure 4.1. The solution pair that the path $p = (v_1, v_2, ..., v_k)$ represents is the concatenation of all the solutions associated with the edges therein, namely,

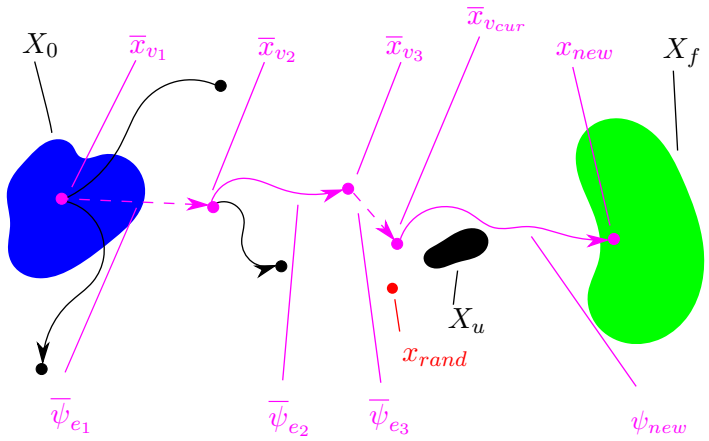
$$\tilde{\psi}_p := \overline{\psi}_{(v_1,v_2)} | \overline{\psi}_{(v_2,v_3)} | \ ... \ | \overline{\psi}_{(v_{k-1},v_k)} \tag{4.1}$$

where $\tilde{\psi}_p$ denotes the solution pair associated with the path p. An example of the path p and its associated solution pair $\tilde{\psi}_p$ is shown in Figure 4.1.

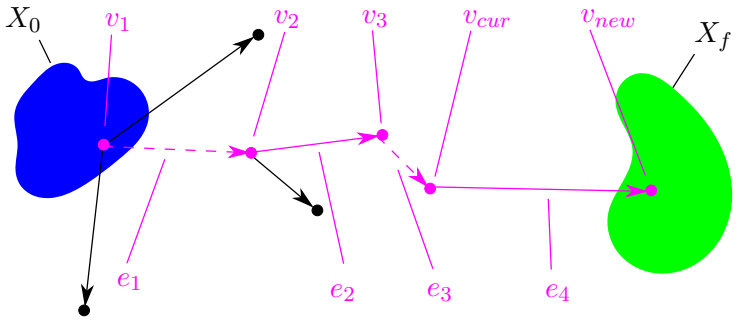
The proposed HyRRT algorithm requires a library of possible inputs. The input library $(\mathcal{U}_C, \mathcal{U}_D)$ includes the input signals that can be applied during flows (collected in $\mathcal{U}_C$) and the input values that can be applied at jumps (collected in $\mathcal{U}_D$). More details about the input library are presented in the forthcoming Section 4.2.

Next, we introduce the main steps executed by HyRRT. Given the motion planning problem $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$ and the input library $(\mathcal{U}_C, \mathcal{U}_D)$, HyRRT performs the following steps:

Step 1: Sample a finite number of points from X_0 and initialize a search tree $\mathcal{T} = (V, E)$



(a) States and solution pairs.



(b) Search tree associated with the states and solution pairs in Figure 4.1(a).

Figure 4.1: The association between states/solution pairs and the vertices/edges in the search tree. The blue region denotes X_0, the green region denotes X_f, and the black region denotes X_u. The dots and lines between dots in Figure 4.1(b) denote the vertices and edges associated with the states and solution pairs in Figure 4.1(a). The path $p = (v_1, v_2, v_3, v_{cur}, v_{new})$ in the search graph in Figure 4.1(b) represents the solution pair $\tilde{\psi}_p = \overline{\psi}_{e_1}|\overline{\psi}_{e_2}|\overline{\psi}_{e_3}|\psi_{new}$ in Figure 4.1(a).

by adding vertices associated with each sampling point.

Step 2: Randomly select one regime among flow regime and jump regime for the evolution of $\mathcal{H}$.

Step 3: Randomly select a point x_{rand} from C' (D') if the flow (jump, respectively) regime is selected in Step 2.

Step 4: Find the vertex associated with the state value that has minimal Euclidean distance to x_{rand}, denoted v_{cur}, as is shown in Figure 4.1(b).

Step 5: Randomly select an input signal (value) from $\mathcal{U}_C$ ($\mathcal{U}_D$) if the flow (jump, respectively) regime is selected. Then, compute a solution pair using continuous (discrete, respectively) dynamics simulator starting from $\overline{x}_{v_{cur}}$ with the selected input applied, denoted $\psi_{new} = (\phi_{new}, \upsilon_{new})$. Denote the final state of ϕ_{new} as x_{new}, as is shown in Figure 4.1(a). If ψ_{new} does not intersect with X_u, add a vertex v_{new} associated with x_{new} to V and an edge (v_{cur}, v_{new}) associated with ψ_{new} to E. Then, go to **Step 2**.

In the remainder of this chapter, first, we formalize the input library for the motion planning problem for hybrid dynamical systems. Then, the continuous dynamics simulator and discrete dynamics simulator to compute a new solution pair from the given starting state and input are introduced. After those components being nicely introduced, the HyRRT algorithm to solve the motion planning problem for hybrid dynamical systems is presented.

4.2 Hybrid Input Library

HyRRT requires a library of inputs to simulate solution pairs. Note that inputs are constrained by the flow set C and the jump set D of the hybrid dynamical system $\mathcal{H}$. Specifically, given C and D, the set of input signals allowed during flows, denoted $\mathcal{U}_C$, and the set of input values at jumps, denoted $\mathcal{U}_D$, are described as follows.

1. The input signal applied during flows is a continuous-time signal, denoted $\tilde{v}$, that is specified by a function from an interval of time of the form $[0, t^*]$ to U_C, namely, for some $t^* \in \mathbb{R}_{\geq 0}$,

   $$\tilde{v} : [0, t^*] \to U_C.$$

   Definition 2.2 also requires that $\tilde{v}$ is Lebesgue measurable and locally essentially bounded. Then, the set $\mathcal{U}_C$ collects each such $\tilde{v}$. Given $\tilde{v} \in \mathcal{U}_C$, the functional $\bar{t} : \mathcal{U}_C \to [0, \infty)$ returns the time duration of $\tilde{v}$. Namely, given $\tilde{v} : [0, t^*] \to U_C$, $\bar{t}(\tilde{v}) = t^*$.

2. The input applied at a jump is specified by the set U_D. The set $\mathcal{U}_D$ collects possible input values that can be applied at jumps, namely,

   $$\mathcal{U}_D \subset U_D.$$

The pair of sets $(\mathcal{U}_C, \mathcal{U}_D)$ defines the input library, denoted $\mathcal{U}$, namely,

$$\mathcal{U} := (\mathcal{U}_C, \mathcal{U}_D).$$

Input Library Construction Procedure: A procedure to construct $\mathcal{U}_C$ using constant inputs is given as follows:

Step 1: Set T_m to a positive constant. Construct the safe input set during the flows from U_C as $U_C^s := \{u \in U_C : \exists x \in C', (x, u) \notin X_u\}$.

Step 2: For each point $u^s \in U_C^s$ and $t_m \in (0, T_m]$, construct an input signal $[0, t_m] \to \{u^s\}$ and add it to $\mathcal{U}_C$.

Set $\mathcal{U}_D$ can be constructed as $\mathcal{U}_D := \{u \in U_D : \exists x \in D', (x, u) \notin X_u\}$ from U_D.

4.3 Continuous Dynamics Simulator

HyRRT requires a simulator to compute the solution pair starting from a given initial state $x_0 \in C'$ with a given input signal $\tilde{v} \in \mathcal{U}_C$ applied, following continuous dynamics. The initial state x_0, the flow set C, and the flow map f are used in the simulator.

Note that when the simulated solution enters the intersection between the flow set C and the jump set D, it can either keep flowing or jump. In [46], the hybrid dynamical system simulator HyEQ uses a scalar priority option flag `rule` to show whether the simulator gives priority to jumps (`rule` $= 1$), priority to flows (`rule` $= 2$), or no priority (`rule` $= 3$) when both $x \in C$ and $x \in D$ hold. When no priority is selected, then the simulator randomly selects to flow or jump. The case `rule` $= 3$ is not considered[1].

The proposed simulator should be able to solve the following problem.

[1] `rule` $= 3$ is not considered since the input signal $\tilde{v}$ is randomly selected from $\mathcal{U}_C$ which satisfies the forthcoming Assumption 4.20. Therefore, the time duration of the simulation result has been randomized. However, `rule` $= 3$ requires an additional random process to determine whether to proceed with the flow or the jump when the simulation result enters $C \cap D$ leading to a redundant random procedure.

Problem 4.1. *Given the flow set C, the flow map f, and the jump set D of a hybrid dynamical system $\mathcal{H}$ with input $u \in \mathbb{R}^m$ and state $x \in \mathbb{R}^n$, a priority option flag* `rule` $\in \{1, 2\}$*, an initial state $x_0 \in \mathbb{R}^n$, and an input signal $\tilde{\upsilon} \in \mathcal{U}_C$ such that $(x_0, \tilde{\upsilon}(0)) \in C$, find a pair $(\phi, \upsilon) : [0, t^*] \times \{0\} \to \mathbb{R}^n \times \mathbb{R}^m$, where $t^* \in [0, \bar{t}(\tilde{\upsilon})]$, such that the following hold:*

1. *$\phi(0, 0) = x_0$;*

2. *For all $t \in [0, t^*]$, $\upsilon(t, 0) = \tilde{\upsilon}(t)$;*

3. *If $[0, t^*]$ has nonempty interior,*

   (a) *the function $t \mapsto \phi(t, 0)$ is locally absolutely continuous,*

   (b) *for all $t \in (0, t^*)$,*

   $$(\phi(t, 0), \upsilon(t, 0)) \in C \backslash D \qquad \text{if } \texttt{rule} = 1$$

   $$(\phi(t, 0), \upsilon(t, 0)) \in C \qquad \text{if } \texttt{rule} = 2,$$

   (c) *for almost all $t \in [0, t^*]$,*

   $$\frac{d}{dt}\phi(t, 0) = f(\phi(t, 0), \upsilon(t, 0)). \tag{4.2}$$

Remark 4.2. *In general, a solution pair to $\mathcal{H}$ that solves Problem 4.1 may not be unique. Note that we can impose assumptions to get uniqueness as in [17, Proposition 2.11], as follows:*

1. *for every $x_0 \in \overline{C} \backslash D$, $T > 0$ and $\tilde{\upsilon} \in \mathcal{U}_C$, if two absolutely continuous $z_1, z_2 : [0, T] \to \mathbb{R}^n$ are such that $z_i(t) = f(z_i(t), \tilde{\upsilon}(t))$ for almost all $t \in [0, T]$, $(z_i(t), \tilde{\upsilon}(t)) \in C$ for all $t \in (0, T]$, and $z_i(0) = x_0$, $i = 1, 2$, then $z_1(t) = z_2(t)$ for all $t \in [0, T]$;*

2. *for every* $x_0 \in C \cap D$, *there does not exist* $\epsilon > 0$ *and an absolutely continuous function* $z : [0, \epsilon] \to \mathbb{R}^n$ *such that* $z(0) = x_0$, $\dot{z}(t) = f(z(t), \upsilon(t))$ *for almost all* $t \in [0, \epsilon]$ *and* $(z_i(t), \upsilon(t)) \in C$ *for all* $t \in (0, \epsilon]$.

The simulator is designed to simulate a maximal solution pair (ϕ, υ) solving Problem 4.1. The definition of maximal solution is given as follows; see [17].

Definition 4.3. *(Maximal solution) A solution pair* ψ *to* $\mathcal{H}$ *that solves Problem 4.1 is said to be maximal if there does not exist another solution pair* ψ' *to* $\mathcal{H}$ *that solves Problem 4.1 such that dom* ψ *is a proper subset of dom* ψ' *and* $\psi(t, 0) = \psi'(t, 0)$ *for all* $t \in$ *dom* ${}_t\psi$.

The module to simulate the maximal solution pair to $\mathcal{H}$ that solves Problem 4.1 is called the *continuous dynamics simulator*. The inputs of this module are the flow set C, the flow map f, the jump set D, the priority option `rule`, the initial state x_0, and the input signal $\tilde{\upsilon}$. The output of this module is the maximal solution pair (ϕ, u) to $\mathcal{H}$ that solves Problem 4.1. This module is denoted as

$$(\phi, u) \leftarrow \texttt{continuous\_simulator}(C, f, D, \texttt{rule}, x_0, \tilde{\upsilon}). \tag{4.3}$$

The continuous dynamics simulator performs the following steps. Given the flow set C, the flow map f, the jump set D the priority option `rule`, the initial state x_0, and the input signal $\tilde{\upsilon}$,

Step 1: Solve for $\hat{\phi} : [0, \overline{t}(\tilde{\upsilon})] \to \mathbb{R}^n$ to satisfy

$$\hat{\phi}(t) = x_0 + \int_0^t f(\hat{\phi}(\tau), \tilde{\upsilon}(\tau)) d\tau \quad t \in [0, \overline{t}(\tilde{\upsilon})]. \tag{4.4}$$

Step 2: Calculate the largest time t such that over $[0, t)$, (ϕ, υ) is in $C \backslash D$ if `rule` $= 1$, or, if `rule` $= 2$, is in C, as follows:

$$\hat{t} := \begin{cases} \max\{t \in [0, \overline{t}(\tilde{\upsilon})] : (\hat{\phi}(t'), \tilde{\upsilon}(t')) \in C \backslash D, \forall t' \in (0, t)\} \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } \texttt{rule} = 1, \\ \max\{t \in [0, \overline{t}(\tilde{\upsilon})] : (\hat{\phi}(t'), \tilde{\upsilon}(t')) \in C, \quad \forall t' \in (0, t)\} \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } \texttt{rule} = 2. \end{cases} \tag{4.5}$$

Step 3: Construct the solution function pair $(\phi, \upsilon) : [0, \hat{t}] \times \{0\} \to \mathbb{R}^n \times \mathbb{R}^m$ by

$$\phi(t, 0) = \hat{\phi}(t), \upsilon(t, 0) = \tilde{\upsilon}(t) \ \forall t \in [0, \hat{t}]. \tag{4.6}$$

The solution function pair (ϕ, u) is a maximal solution to $\mathcal{H}$ that solves Problem 4.1 and, hence, the output of module `continuous_simulator`. The construction of $\hat{\phi}$ in Step 1 can be approximated by employing numerical integration methods. To determine $\hat{t}$ in Step 2, zero-crossing detection algorithms can be used to detect the largest time t such that over $[0, t)$, (ϕ, υ) is in $C \backslash D$ if `rule` $= 1$, or, if `rule` $= 2$, is in C. For the computation framework that implements the simulator of continuous dynamics, see Appendix B.1.

4.4 Discrete Dynamics Simulator

HyRRT algorithm also requires a simulator to compute the solution pair with a single jump, starting from an initial state $x_0 \in D'$ with a given input value $u_D \in \mathcal{U}_D$ applied. Such a simulator only requires evaluating g as the following problem states.

Problem 4.4. *Given the jump set D and jump map g of hybrid dynamical system $\mathcal{H}$ with input $u \in \mathbb{R}^m$, state $x \in \mathbb{R}^n$, an initial state $x_0 \in \mathbb{R}^n$, and an input value $u_D \in \mathcal{U}_D$ such that $(x_0, u_D) \in D$, find a pair $(\phi, \upsilon) : \{0\} \times \{0,1\} \to \mathbb{R}^n \times \mathbb{R}^m$ such that the following hold:*

1. $\phi(0,0) = x_0$;
2. $\upsilon(0,0) = u_D$;
3. $\phi(0,1) = g(\phi(0,0), \upsilon(0,0))$.

Problem 4.4 can be solved by constructing a function pair $(\phi, \upsilon) : \{0\} \times \{0,1\} \to \mathbb{R}^n \times \mathbb{R}^m$ in the following way:

$$\begin{aligned} \phi(0,0) \leftarrow x_0, \quad \phi(0,1) \leftarrow g(x_0, u_D) \\ \upsilon(0,0) \leftarrow u_D, \quad \upsilon(0,1) \leftarrow p \in \mathbb{R}^m. \end{aligned} \tag{4.7}$$

In (4.7), we can implement $\upsilon(0,1) \leftarrow p \in \mathbb{R}^m$ by selecting an arbitrary point p in $\mathbb{R}^m$ such that $(\phi(0,1), p) \notin X_u$.

The function pair in (4.7) can be constructed by a module called *discrete dynamics simulator*. The inputs of this module are the jump set D, the jump map g, the initial state $x_0 \in D'$, and the input $u_D \in \mathcal{U}_D$ such that $(x_0, u_D) \in D$. The output of this module is the solution pair (ϕ, υ) constructed in (4.7). This module is denoted as

$$(\phi, \upsilon) \leftarrow \texttt{discrete\_simulator}(D, g, x_0, u_D). \tag{4.8}$$

4.5 HyRRT Algorithm

Following the overview in Section 4.1, the proposed algorithm is given in Algorithm 2. The inputs of Algorithm 2 are the problem $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$, the input library $(\mathcal{U}_C, \mathcal{U}_D)$, a parameter $p_n \in (0, 1)$, which tunes the probability of proceeding with the flow regime or the jump regime, an upper bound $K \in \mathbb{N}_{>0}$ for the number of iterations to execute, and two tunable sets $X_c \supset \overline{C'}$ and $X_d \supset D'$, which act as constraints in finding a closest vertex to x_{rand}. HyRRT is implemented in the following algorithm:

Algorithm 2 HyRRT algorithm

Input: $X_0, X_f, X_u, \mathcal{H} = (C, f, D, g), (\mathcal{U}_C, \mathcal{U}_D), p_n \in (0, 1)$, $K \in \mathbb{N}_{>0}$

1: $\mathcal{T}$.`init`(X_0).
2: **for** $k = 1$ to K **do**
3: randomly select a real number r from $[0, 1]$.
4: **if** $r \leq p_n$ **then**
5: $x_{rand} \leftarrow$ `random_state`$(\overline{C'})$.
6: `extend`$(\mathcal{T}, x_{rand}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u, X_c)$.
7: **else**
8: $x_{rand} \leftarrow$ `random_state`(D').
9: `extend`$(\mathcal{T}, x_{rand}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u, X_d)$.
10: **end if**
11: **end for**
12: **return** $\mathcal{T}$.

Step 1 in the overview provided in Section 4.1 corresponds to the function call $\mathcal{T}$.`init` in line 1 of Algorithm 2. **Step** 2 is implemented in line 3. **Step** 3 is implemented

Algorithm 3 Extend function

1: **function** EXTEND($(\mathcal{T}, x, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u, X_*)$)

2: $v_{cur} \leftarrow$ `nearest_neighbor`$(x, \mathcal{T}, \mathcal{H}, X_*)$;

3: (`is_a_new_vertex_generated`$, x_{new}, \psi_{new}) \leftarrow$ `new_state`$(v_{cur}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u)$

4: **if** `is_a_new_vertex_generated` $= true$ **then**

5: $v_{new} \leftarrow \mathcal{T}$.`add_vertex`$(x_{new})$;

6: $\mathcal{T}$.`add_edge`$(v_{cur}, v_{new}, \psi_{new})$;

7: **return** `Advanced`;

8: **end if**

9: **return** `Trapped`;

10: **end function**

by the function call `random_state` in lines 5 and 8. **Step** 4 corresponds to the function call `nearest_neighbor` in line 1 of the function call `extend`. **Step** 5 is implemented by the function calls `new_state`, $\mathcal{T}$.`add_vertex`, and $\mathcal{T}$.`add_edge` in lines 3, 4, and 5 of the function call `extend`.

Each function in Algorithm 2 is defined next.

4.5.1 $\mathcal{T}$.`init`(X_0)

The function call $\mathcal{T}$.`init` is used to initialize a search tree $\mathcal{T} = (V, E)$. It randomly selects a finite number of points from X_0. For each sampling point $x_0 \in X_0$, a vertex v_0 associated with x_0 is added to V. At this step, no edge is added to E.

4.5.2 $x_{rand} \leftarrow$ `random_state`(S)

The function call `random_state` randomly selects a point from the set $S \subset \mathbb{R}^n$. Rather than to select from $\overline{C'} \cup D'$, it is designed to select points from $\overline{C'}$ and D' separately depending on the value of r. The reason is that if $\overline{C'}$ (or D') has zero measure while D' (respectively, $\overline{C'}$) does not, the probability that the point selected from $\overline{C'} \cup D'$ lies in $\overline{C'}$ (respectively, D') is zero, which would prevent establishing probabilistic completeness.

4.5.3 $v_{cur} \leftarrow$ `nearest_neighbor`$(x_{rand}, \mathcal{T}, \mathcal{H}, X_*)$

The function call `nearest_neighbor` searches for a vertex v_{cur} in the search tree $\mathcal{T} = (V, E)$ such that its associated state value has minimal distance to x_{rand}. This function is implemented as solving the following optimization problem over $X_\star$, where $\star$ is either c or d.

Problem 4.5. *Given a hybrid dynamical system $\mathcal{H} = (C, f, D, g)$, $x_{rand} \in \mathbb{R}^n$, and a search tree $\mathcal{T} = (V, E)$, solve*

$$\begin{aligned} \underset{v \in V}{\arg\min} \quad & |\overline{x}_v - x_{rand}| \\ s.t. \quad & \overline{x}_v \in X_\star. \end{aligned}$$

The data of Problem 4.5 comes from the arguments of the `nearest_neighbor` function call. This optimization problem can be solved by traversing all the vertices in $\mathcal{T} = (V, E)$.

4.5.4 $(\texttt{is\_a\_new\_vertex\_generated}, x_{new}, \psi_{new}) \leftarrow \texttt{new\_state}(v_{cur}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H} = (C, f, D, g), X_u)$

Given v_{cur}, if $\overline{x}_{v_{cur}} \in \overline{C'} \backslash D'$, the function call `new_state` randomly selects an input signal $\tilde{v}$ from $\mathcal{U}_C$ such that $(\overline{x}_{v_{cur}}, \tilde{v}(0)) \in \overline{C}$ and generates a new maximal solution pair, denoted $\psi_{new} = (\phi_{new}, \upsilon_{new})$, by

$$\psi_{new} \leftarrow \texttt{continuous\_simulator}(C, f, D, 2, \overline{x}_{v_{cur}}, \tilde{v}) \tag{4.9}$$

where `continuous_simulator` is formulated as in (4.3).

If $\overline{x}_{v_{cur}} \in D' \backslash \overline{C'}$, the function call `new_state` randomly selects an input signal u_D from $\mathcal{U}_D$ such that $(\overline{x}_{v_{cur}}, u_D) \in D$ and generates a new solution pair, denoted $\psi_{new} = (\phi_{new}, \upsilon_{new})$, by

$$\psi_{new} \leftarrow \texttt{discrete\_simulator}(D, g, \overline{x}_{v_{cur}}, u_D) \tag{4.10}$$

where `discrete_simulator` is formulated as in (4.8).

If $\overline{x}_{v_{cur}} \in \overline{C'} \cap D'$, then this function generates ψ_{new} by randomly selecting flows or jump. This random selection is implemented by randomly selecting a real number r_D from the interval $[0, 1]$ and comparing r_D with a user-defined parameter $p_D \in (0, 1)$. If $r_D \leq p_D$, then the function call `new_state` generates ψ_{new} by flow, otherwise, by jump. The final state of ψ_{new} is denoted as x_{new}.

After ψ_{new} and x_{new} are generated, the function `new_state` checks if ψ_{new} is trivial. If so, then ψ_{new} does not explore any unexplored space and is not necessary to be added into $\mathcal{T}$. Hence, `is_a_new_vertex_generated` $\leftarrow$ `false` and the function

call `new_state` is returned. Else, the function `new_state` checks if there exists $(t, j) \in$ dom ψ_{new} such that $\psi_{new}(t, j) \in X_u$. If so, then ψ_{new} intersects with the unsafe set and `is_a_new_vertex_generated` $\leftarrow$ `false`. Otherwise, `is_a_new_vertex_generated` $\leftarrow$ `true`.

4.5.5 $v_{new} \leftarrow \mathcal{T}$.add_vertex$(x_{new})$ and $\mathcal{T}$.add_edge$(v_{cur}, v_{new}, \psi_{new})$

The function call $\mathcal{T}$.`add_vertex`(x_{new}) adds a new vertex v_{new} associated with x_{new} to $\mathcal{T}$ and returns v_{new}. The function call $\mathcal{T}$.`add_edge`$(v_{cur}, v_{new}, \psi_{new})$ adds a new edge $e_{new} = (v_{cur}, v_{new})$ associated with ψ_{new} to $\mathcal{T}$.

4.5.6 Solution Checking during HyRRT Construction

When the function call `extend` returns `Advanced`, HyRRT checks if a path in $\mathcal{T}$ can be used to construct a motion plan solving the given motion planning problem. If this function finds a path $p = ((v_0, v_1), (v_1, v_2), ..., (v_{n-1}, v_n)) =: (e_0, e_1, ..., e_{n-1})$ in $\mathcal{T}$ such that

1) $\overline{x}_{v_0} \in X_0$,

2) $\overline{x}_{v_n} \in X_f$,

3) for $i \in \{0, 1, ..., n-2\}$, if $\overline{\psi}_{e_i}$ and $\overline{\psi}_{e_{i+1}}$ are both purely continuous, then $\overline{\psi}_{e_{i+1}}(0, 0) \in C$,

then the solution pair $\tilde{\psi}_p$ associated with path p, defined in (4.1), is a motion plan to the given motion planning problem. More specifically, the items listed above guarantee

that ψ satisfies all the conditions in Problem 3.1. Items 1 and 2 guarantee that $\tilde{\psi}_p$ starts from X_0 and ends within X_f. Then, we show that each condition in Proposition 3.13 is satisfied such that Proposition 3.13 guarantees that ψ is a solution pair to $\mathcal{H}$ as follows:

1. Because the input signal in $\mathcal{U}_C$ is compact, the solution pairs generated by `new_state` by flow are compact. The solution pairs generated by `new_state` by jump are compact for free. Therefore, the first condition in Proposition 3.13 is satisfied.

2. Note that for $i \in \{0, 1, 2, ..., n-2\}$, the final state of $\overline{\psi}_{e_i} = \overline{\psi}_{(v_i, v_{i+1})}$ equals the initial state of $\overline{\psi}_{e_{i+1}} = \overline{\psi}_{(v_{i+1}, v_{i+2})}$ because both of them are $\overline{x}_{v_{i+1}}$. Then, the second condition in Proposition 3.13 is satisfied.

3. Then, item 3 above guarantees that the third condition in Proposition 3.13 is satisfied.

Therefore, Proposition 3.13 guarantees that ψ_p is a solution pair to $\mathcal{H}$. Note that for each e in p, $\overline{\psi}_e$ does not intersect with unsafe set because the solution pairs that intersect the unsafe set have been excluded by the function *new_state*. Therefore, ψ_p satisfies all the conditions in Problem 3.1.

Remark 4.6. *Note that the choices of inputs in the fucntion call* `new_state` *are random. Some RRT variants choose the optimal input that drives x_{new} closest to x_{rand}. However, [31] proves that such a choice makes the RRT algorithm probabilistically incomplete.* ▲

Remark 4.7. *In practice, item 2 above is too restrictive. Given $\epsilon > 0$ representing the*

tolerance associated with this condition, we implement item 2 as

$$|\overline{x}_{v_n}|_{X_f} \leq \epsilon. \tag{4.11}$$

▲

4.6 Probabilistic Completeness Analysis

This section establishes probabilistic completeness of the HyRRT algorithm. Probabilistic completeness means that the probability that the planner fails to return a motion plan, if one exists, approaches zero as the number of samples approaches infinity.

4.6.1 Clearance of Motion Plan and Inflation of a Hybrid Dynamical System

The clearance of a motion plan captures the distance between the motion plan and the boundary of the constraint sets, which in Problem 3.1, includes the initial state set X_0, the final state set X_f, the unsafe set X_u, the flow set C, and the jump set D. We propose two different clearances, *safety clearance* and *dynamics clearance*, that capture the distance to the constraint sets (X_0, X_f, X_u) and (C, D), respectively.

Definition 4.8 (Safety clearance of a motion plan)**.** *Given a motion plan $\psi = (\phi, \upsilon)$ to the motion planning problem $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$, the safety clearance of $\psi = (\phi, \upsilon)$ is given by $\delta_s > 0$ if, for each $\delta' \in [0, \delta_s]$, the following conditions are satisfied:*

1) $\phi(0,0) + \delta'\mathbb{B} \subset X_0$*;*

2) $\phi(T, J) + \delta'\mathbb{B} \subset X_f$, *where* $(T, J) = \max dom\ \psi$;

3) For all $(t, j) \in dom\ \psi$, $(\phi(t, j) + \delta'\mathbb{B}, \upsilon(t, j) + \delta'\mathbb{B}) \cap X_u = \emptyset$.

Definition 4.9 (Dynamics clearance of a motion plan)**.** *Given a motion plan* $\psi = (\phi, \upsilon)$ *to the motion planning problem* $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$*, the dynamics clearance of* $\psi = (\phi, \upsilon)$ *is given by* $\delta_d > 0$ *if, for each* $\delta' \in [0, \delta_d]$*, the following conditions are satisfied:*

1) For all $(t, j) \in dom\ \psi$ *such that* I^j *has nonempty interior,* $(\phi(t, j) + \delta'\mathbb{B}, \upsilon(t, j) + \delta'\mathbb{B}) \subset C$;

2) For all $(t, j) \in dom\ \psi$ *such that* $(t, j+1) \in dom\ \psi$, $(\phi(t, j)+\delta'\mathbb{B}, \upsilon(t, j)+\delta'\mathbb{B}) \subset D$.

Remark 4.10. *The definition of the two types of clearance is analogous to the clearance in [27], albeit with a specific consideration for the boundaries of different constraint sets. Safety clearance in Definition 4.8 is defined as the minimal positive distance between the motion plan and the nearest boundaries of the initial state set, final state set, and the unsafe set. Dynamics clearance in Definition 4.9, on the other hand, represents the minimal positive distance between the motion plan and the closest boundaries of the flow set and the jump set, depending on whether the motion plan is during a flow or at a jump.*

With both safety clearance and dynamics clearance defined, we are ready to define the clearance of a motion plan.

Definition 4.11 (Clearance of a motion plan)**.** *Given a motion plan $\psi = (\phi, \upsilon)$ to the motion planning problem $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$, the clearance of ψ, denoted δ, is defined as the minimum between its safety clearance δ_s and dynamics clearance δ_d, i.e., $\delta := \min\{\delta_s, \delta_d\}$.*

In the probabilistic completeness result in [27, Theorem 2], a motion plan with positive clearance is assumed to exist. However, the assumption that there exists a positive dynamics clearance is restrictive for hybrid dynamical systems. Indeed, if the motion plan reaches the boundary of the flow set or of the jump set, then the motion plan has no (dynamics) clearance; see Definition 4.9. Figure 4.2(a) shows a motion plan to the sample motion planning problem for the actuated bouncing ball system in Example 3.2 without clearance.

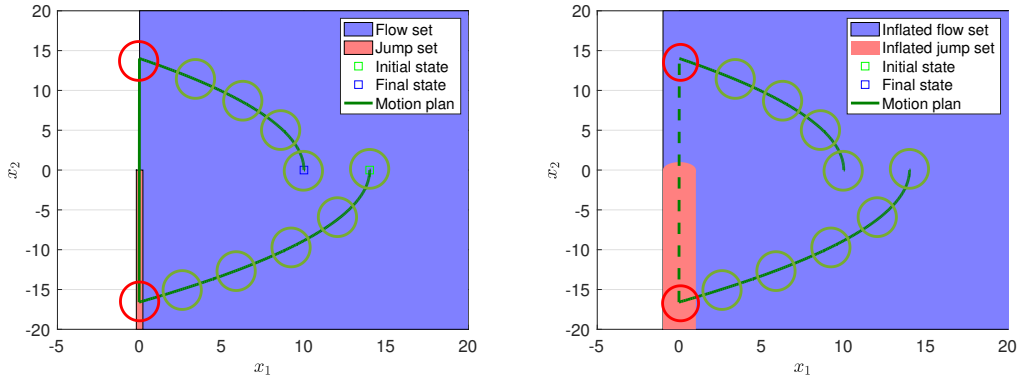
To overcome this issue and to assure that HyRRT is probabilistically complete, we inflate the hybrid dynamical system $\mathcal{H} = (C, f, D, g)$ as follows.

Definition 4.12 (δ-inflation of a hybrid dynamical system)**.** *Given a hybrid dynamical system $\mathcal{H} = (C, f, D, g)$ and $\delta > 0$, the δ-inflation of the hybrid dynamical system $\mathcal{H}$, denoted $\mathcal{H}_\delta$ with data $(C_\delta, f_\delta, D_\delta, g_\delta)$, is given by*[2]

$$\mathcal{H}_\delta : \begin{cases} \dot{x} = f_\delta(x, u) & (x, u) \in C_\delta \\ x^+ = g_\delta(x, u) & (x, u) \in D_\delta \end{cases} \tag{4.12}$$

where

[2]The flow set C (and the jump set D) inflates the state x and input u, respectively, yielding their independent ranges. This ensures that the sampling process is a Bernoulli process.



(a) (b)

Figure 4.2: Figure 4.2(a) shows a sample motion plan for the bouncing ball system in Example 3.2 without clearance. The dark green trajectory shows the motion plan. The blue region denotes the projection of flow set on the state space. The circles denote the boundaries of the balls along the motion plan at specific hybrid time instances. Note that the red balls depicted in Figure 4.2(a), which encircle the state of the motion plan at the boundary of the flow set, are not subsets of the flow set. In this case, the clearance δ is zero. However, for the inflated system in (4.12), the existence of the circles along the motion plan with radius δ, as shown in Figure 4.2(b), implies the positive clearance δ of the motion plan.

1) The δ-inflation of the flow set is constructed as

$$C_\delta := \{(x,u) \in \mathbb{R}^n \times \mathbb{R}^m : \exists (y,v) \in C : x \in y + \delta\mathbb{B}, u \in v + \delta\mathbb{B}\}, \tag{4.13}$$

2) The δ-inflation of the flow map is constructed as

$$f_\delta(x,u) := f(x,u) \quad \forall (x,u) \in C_\delta, \tag{4.14}$$

3) The δ-inflation of the jump set is constructed as

$$D_\delta := \{(x,u) \in \mathbb{R}^n \times \mathbb{R}^m : \exists (y,v) \in D : x \in y + \delta\mathbb{B}, u \in v + \delta\mathbb{B}\}, \tag{4.15}$$

4) The δ-inflation of the jump map is constructed as

$$g_\delta(x,u) := g(x,u) \quad \forall (x,u) \in D_\delta. \tag{4.16}$$

The above outlines a general method of constructing the δ-inflation of the given hybrid dynamical system. Next, this method is exemplified in the actuated bouncing ball system.

Example 4.13 (Actuated bouncing ball system in Example 3.2, revisited)**.** *Given $\delta > 0$, the δ-inflation of hybrid dynamical system of the actuated bouncing ball system is constructed as follows.*

- *From (4.13), the δ-inflation of flow set C, denoted C_δ, is given by*

$$C_\delta = \{(x,u) \in \mathbb{R}^2 \times \mathbb{R} : x_1 \geq -\delta\}. \tag{4.17}$$

- *From (4.14), the δ-inflation of flow map f, denoted f_δ, is given by*

$$f_\delta(x,u) = f(x,u) = \begin{bmatrix} x_2 \\ \\ -\gamma \end{bmatrix} \qquad \forall (x,u) \in C_\delta. \tag{4.18}$$

- *From (4.15), the δ-inflation of jump set D, denoted D_δ, is given by*

$$D_\delta := (\{(0,0)\} + \delta\mathbb{B}) \cup \{(x,u) \in \mathbb{R}^2 \times \mathbb{R} : x_2 \leq 0 \text{ and } -\delta \leq x_1 \leq \delta\}. \tag{4.19}$$

- *From (4.16), the δ-inflation of jump map g, denoted g_δ, is given by*

$$g_\delta(x,u) := g(x,u) = \begin{bmatrix} x_1 \\ \\ -\lambda x_2 + u \end{bmatrix} \qquad \forall (x,u) \in D_\delta. \tag{4.20}$$

As is shown in Figure 4.2(b), the inflation of a hybrid dynamical system defined above serves to extend the boundary of both the flow and jump sets in $\mathcal{H}$ by δ and to capture the same continuous and discrete dynamics in those extended sets as in $\mathcal{H}$. Consequently, under the assumption that a motion plan to $\mathcal{P}$ with positive safety clearance exists, this methodology facilitates the establishment of such a motion plan with positive dynamics clearance and, in turn, with positive clearance.

Next, we show that a motion plan to the original motion planning problem is also a motion plan to the motion planning problem for its δ-inflation.

Proposition 4.14. *Given a motion planning problem $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$ in Problem 3.1 with positive safety clearance, if ψ is a motion plan to $\mathcal{P}$, then for each $\delta > 0$, ψ is also a motion plan to the motion planning problem $\mathcal{P}_\delta = (X_0, X_f, X_u, (C_\delta, f_\delta, D_\delta, g_\delta))$, where $(C_\delta, f_\delta, D_\delta, g_\delta)$ is the δ-inflation of the hybrid dynamical system defined by (C, f, D, g).*

Proof. See Appendix B.2. □

Next we show that the existing motion plan with positive safety clearance has positive clearance for the motion planning problem for the δ-inflation of the original hybrid dynamical system $\mathcal{H}$.

Lemma 4.15. *Let ψ be a motion plan to the motion planning problem $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$ formulated as Problem 3.1 with positive safety clearance $\delta_s > 0$. Then, for each $\delta_f > 0$, ψ is a motion plan to the motion planning problem $\mathcal{P}_{\delta_f} = (X_0, X_f, X_u, (C_{\delta_f}, f_{\delta_f}, D_{\delta_f}, g_{\delta_f}))$ with clearance $\delta = \min\{\delta_s, \delta_f\}$, where $\mathcal{H}_{\delta_f} = (C_{\delta_f}, f_{\delta_f}, D_{\delta_f}, g_{\delta_f})$ is the δ_f-inflation of $\mathcal{H} = (C, f, D, g)$.*

Proof. See Appendix B.3. □

4.6.2 Assumptions

Similar to [27, Definition 2], we consider input functions that are piecewise constant in the following sense.

Definition 4.16 (Piecewise-constant function)**.** *A function $\tilde{v}_c : [0, T] \to U_C$ is said to be a piecewise-constant function for probabilistic completeness if there exists $\Delta t \in \mathbb{R}_{>0}$, called resolution, such that*

1) $k := \frac{T}{\Delta t} \in \mathbb{N}$;

2) for each $i \in \{1, 2, ..., k\}$, $t \mapsto \tilde{v}_c(t)$ is constant over $[(i-1)\Delta t, i\Delta t)$.

From Definition 4.16, we define a motion plan notion with piecewise-constant inputs as follows.

Definition 4.17 (Motion plan with piecewise-constant input)**.** *Given a motion planning problem $\mathcal{P}$, a motion plan (ϕ, υ) to $\mathcal{P}$ is said to be a* motion plan with piecewise-constant input with resolution Δt *if, for all $j \in \mathbb{N}$ such that I_υ^j has nonempty interior, $t \mapsto \upsilon(t, j)$ is a piecewise constant function with resolution Δt.*

Similar to [35], HyRRT is assumed to be executed uniformly in relation to the random selection involved in **Step** 1, **Step** 3, and **Step** 5 of HyRRT, in the following sense.

Definition 4.18 (Uniform HyRRT)**.** *HyRRT is said to be executed uniformly if the probability distribution of the random selection in the function calls $\mathcal{T}$.`init`, `random_state`, and `new_state` is the uniform distribution.*

Remark 4.19. *By Definition 4.18, the computation of the probability of randomly selecting a point that lies in a given set is simplified. When randomly selecting a point s from a set S, according to [4, Page 257 (20.9)], the probability that s belongs to a subset $R \subset S$ is*

$$\text{Prob}[s \in R] = \int_{s \in R} \frac{1}{\mu(S)} ds = \frac{\mu(R)}{\mu(S)} \tag{4.21}$$

where μ denotes the Lebesgue measure of a set.

Subsequently, we define a complete input library that encompasses each possible input value that can be utilized either during the flow or at a jump.

Definition 4.20 (Complete input library)**.** *The input library ($\mathcal{U}_C$,$\mathcal{U}_D$) is said to be a complete input library if*

1) $\mathcal{U}_C$ *is the set of constant input signals and includes all possible input signals such that, for some* $T_m > 0$*, their time domains are closed subintervals of the interval* $[0, T_m]$ *including zero and their images belong to* U_C*. In other words, there exists* $T_m > 0$ *such that* $\mathcal{U}_C = \{\tilde{v} : dom\ \tilde{v} = [0, T] \subset [0, T_m], \tilde{v}\ is\ constant, \tilde{v} \in U_C\}$*;*

2) $\mathcal{U}_D = U_D$*.*

Remark 4.21. *From Definition 4.20, the input signals in* $\mathcal{U}_C$ *are all constant functions. This property of* $\mathcal{U}_C$ *allows for the inputs of a motion plan with piecewise-constant input to be constructed by concatenating constant input signals in* $\mathcal{U}_C$*. The set* $\mathcal{U}_C$ *collects each possible constant input signal taking values from* U_C *and with maximal duration* $[0, T_m]$*. The upper bound* T_m *on their duration ensures a positive lower bound on the probability of sampling an appropriate input duration by the function call* `new_state` *in Section 4.5.4, where an input signal* $\tilde{v}$ *is randomly selected from* $\mathcal{U}_C$*. Without this upper bound* T_m*, according to (4.21), the probability of selecting any finite subintervals from* $[0, \infty)$ *is* 0 *because the Lebesgue measure of* $[0, \infty)$ *is infinity. The selection process involves a random choice of* t_m *from the interval* $[0, T_m]$ *and* u_C *from* U_C*, from where a constant input signal is constructed as* $\tilde{v} : [0, t_m] \to \{u_C\}$*.* ▲

To ensure that each random process in HyRRT, specially within the function calls $\mathcal{T}$.`init`, `random_state`, and `new_state`, returns a suitable sample with positive probability, it is essential to assume that the Lebesgue measure of the sets being sampled is both nonzero and finite.

Assumption 4.22. *The sets X_0, C', D', U_C, U_D, which HyRRT makes random selection from, have finite and positive Lebesgue measure.*

Remark 4.23. *Under Assumption 4.22, it is guaranteed that $\mu(S) < \infty$. Then, if HyRRT is executed uniformly and $\mu(R)$ is positive, from (4.21), it follows that* $\text{Prob}[s \in R] \in (0, 1]$. ▲

The following assumption is imposed on the flow map f of the hybrid dynamical system $\mathcal{H}$ in (2.1).

Assumption 4.24. *The flow map f is Lipschitz continuous. In particular, there exist $K_x^f, K_u^f \in \mathbb{R}_{>0}$ such that, for each (x_0, x_1, u_0, u_1) such that $(x_0, u_0) \in C$, $(x_0, u_1) \in C$, and $(x_1, u_0) \in C$,*

$$|f(x_0, u_0) - f(x_1, u_0)| \leq K_x^f |x_0 - x_1|$$

$$|f(x_0, u_0) - f(x_0, u_1)| \leq K_u^f |u_0 - u_1|.$$

Remark 4.25. *Assumption 4.24 adheres to the Lipschitz continuity assumption on differential constraints, as outlined in [27]. Assumption 4.24 guarantees that the flow map is Lipschitz continuous for both state and input arguments. This assumption establishes an explicit upper bound, parameterized by both state and input, on the distance between the motion plan and the simulated purely continuous solution pair in the forthcoming Lemma 4.33. Following the methodology in the proof of [27, Lemma 3], by ensuring this upper bound below the motion plan's clearance, a range of the input signals in $\mathcal{U}_C$ with positive Lebesgue measure is determined that are capable of simulating a purely continuous solution pair that stays within the motion plan's clearance. As per (4.21),*

there is a guaranteed positive probability of randomly sampling an input signal from $\mathcal{U}_C$ that falls within this specified range. This positive probability is instrumental in ensuring that in each Bernoulli trials in the proof of the upcoming Proposition 4.37, there is a positive probability of achieving a successful outcome, which eventually leads to the probabilistic completeness guarantee. ▲

Example 4.26 (Actuated bouncing ball system in Example 3.2, revisited)**.** *In the bouncing ball system, the flow map is defined as* $f(x,u) = \begin{bmatrix} x_2 \\ \\ -\gamma \end{bmatrix}$. *The flow set is defined as* $C := \{(x,u) \in \mathbb{R}^2 \times \mathbb{R} : x_1 \geq 0\}$. *For each* (x_0, x_1, u_0) *such that* $(x_0, u_0) \in C$ *and* $(x_1, u_0) \in C$, *we have*

$$|f(x_0,u_0) - f(x_1,u_0)| = \left| \begin{bmatrix} x_{0,2} - x_{1,2} \\ \\ 0 \end{bmatrix} \right| = |x_{0,2} - x_{1,2}|$$

$$\leq |x_0 - x_1|$$

where $x_{0,2}$ *and* $x_{1,2}$ *denote the second component of* x_0 *and* x_1, *respectively. Therefore,* $K_x^f = 1 > 0$ *is such that* $|f(x_0,u_0) - f(x_1,u_0)| \leq K_x^f |x_0 - x_1|$ *for each* (x_0, x_1, u_0) *such that* $(x_0, u_0) \in C$ *and* $(x_1, u_0) \in C$. *Now, for each* (x_0, u_0, u_1) *such that* $(x_0, u_0) \in C$ *and* $(x_0, u_1) \in C$, *we have* $|f(x_0,u_0) - f(x_0,u_1)| = 0$. *Therefore, any* $K_u^f > 0$ *is such that* $|f(x_0,u_0) - f(x_0,u_1)| \leq K_u^f |u_0 - u_1|$ *for each* (x_0, u_0, u_1) *such that* $(x_0, u_0) \in C$ *and* $(x_0, u_1) \in C$. *Therefore, the flow map and flow set in Example 3.2 satisfy Assumption 4.24.*

The following assumption is imposed on the jump map g of the hybrid dynamical system $\mathcal{H}$ in (2.1).

Assumption 4.27. *The jump map g is such that there exist $K_x^g, K_u^g \in \mathbb{R}_{>0}$ such that, for each $(x_0, u_0) \in D$ and each $(x_1, u_1) \in D$,*

$$|g(x_0, u_0) - g(x_1, u_1)| \leq K_x^g |x_0 - x_1| + K_u^g |u_0 - u_1|.$$

Remark 4.28. *Assumption 4.27 enables to establish an explicit upper bound for the distance between the motion plan and the simulated purely discrete solution pair, as will be further detailed in the Lemma 4.35, which is forthcoming. Adopting the approach from the proof of [27, Lemma 3], by keeping this upper bound beneath the motion plan's clearance, a specific range of the input values in $\mathcal{U}_D$, possessing a positive Lebesgue measure, is identified that are capable of simulating a purely discrete solution pair that stays within the motion plan's clearance. As per (4.21), there is a guaranteed positive probability of randomly sampling an input value from $\mathcal{U}_D$ that falls within this specified range. This positive probability ensures that, in the Bernoulli trials in the proof of the upcoming Proposition 4.37, aside from those already covered by Assumption 4.24, each trial has a positive probability of yielding a successful outcome, eventually ensuring the probabilistic completeness of HyRRT.* ▲

Example 4.29 (Actuated bouncing ball example in Example 3.2, revisited)**.** *In Example 3.2, the jump map is defined as* $g(x, u) = \begin{bmatrix} x_1 \\ -\lambda x_2 + u \end{bmatrix}$. *The jump set is defined as* $D := \{(x, u) \in \mathbb{R}^2 \times \mathbb{R} : x_1 = 0, x_2 \leq 0, u \geq 0\}$. *For each pair of* $(x_0, u_0) \in D$ *and*

$(x_1, u_1) \in D$, *we have*

$$
\begin{aligned}
|g(x_0, u_0) - g(x_1, u_1)| &= \left| \begin{bmatrix} x_{0,1} - x_{1,1} \\ -\lambda(x_{0,2} - x_{1,2}) + u_0 - u_1 \end{bmatrix} \right| \\
&= \left| \begin{bmatrix} x_{0,1} - x_{1,1} \\ -\lambda(x_{0,2} - x_{1,2}) \end{bmatrix} + \begin{bmatrix} 0 \\ u_0 - u_1 \end{bmatrix} \right| \\
&\leq \left| \begin{bmatrix} x_{0,1} - x_{1,1} \\ -\lambda(x_{0,2} - x_{1,2}) \end{bmatrix} \right| + \left| \begin{bmatrix} 0 \\ u_0 - u_1 \end{bmatrix} \right| \\
&\leq \sqrt{1 + \lambda^2}|x_0 - x_1| + |u_0 - u_1|.
\end{aligned}
$$

Therefore, $K_x^g = \sqrt{1 + \lambda^2}$ *and* $K_u^g = 1$ *are such that* $|g(x_0, u_0) - g(x_1, u_1)| \leq K_x^g|x_0 - x_1| + K_u^g|u_0 - u_1|$ *for each pair of* $(x_0, u_0) \in D$ *and* $(x_1, u_1) \in D$.

4.6.3 Probabilistic Completeness Guarantee

Our main result shows that HyRRT is probabilistically complete without assuming positive clearance, achieved through properly exploiting the inflation $\mathcal{H}_{\delta_f}$.

Theorem 4.30. *Given a motion planning problem* $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$, *suppose that Assumptions 4.22, 4.24, and 4.27 are satisfied, and that there exists a compact motion plan* (ϕ, υ) *to* $\mathcal{P}$ *with* safety *clearance* $\delta_s > 0$ *and piecewise-constant input. Then, using a complete input library and when executed uniformly (as defined in Definition 4.18) to solve the problem* $\mathcal{P}_{\delta_f} = (X_0, X_f, X_u, (C_{\delta_f}, f_{\delta_f}, D_{\delta_f}, g_{\delta_f}))$, *where, for some* $\delta_f > 0$, $(C_{\delta_f}, f_{\delta_f}, D_{\delta_f}, g_{\delta_f})$ *denotes the* δ_f*-inflation of* (C, f, D, g) *in (4.12), the probability that HyRRT fails to find a motion plan* $\psi' = (\phi', \upsilon')$ *to* $\mathcal{P}_{\delta_f}$ *such that* ϕ' *is* $(\tilde{\tau}, \tilde{\delta})$*-close to* ϕ

after k iterations is at most $a\exp(-bk)$, where $a, b \in \mathbb{R}_{>0}$, $\tilde{\tau} = \max\{T + J, T' + J'\}$, $(T, J) = \max dom\ \psi$, $(T', J') = \max dom\ \psi'$, and $\tilde{\delta} = \min\{\delta_s, \delta_f\}$.

The proof of Theorem 4.30 is established as follows. By the positive safety clearance assumption and exploiting the inflation of $\mathcal{H}$ in (4.12), we establish that (ϕ, υ) is a motion plan to $\mathcal{P}_\delta$ with positive clearance (see Lemma 4.15 in Section 4.6.1). Then, given that (ϕ, υ) is a motion plan to $\mathcal{P}_\delta$ with positive clearance, we demonstrate that the probability of HyRRT failing to find a motion plan *with positive clearance* is converging to zero as the number of iterations approaches infinity. To demonstrate this result, we first establish that the probabilities of the following probabilistic events are positive:

E1) The function call `nearest_neighbor` in Section 4.5.3 returns a current vertex in the search tree within the clearance of (ϕ, υ) (see Lemma 4.31 in the forthcoming Section 4.6.4);

E2) If E1 occurs, the function call `new_state` in Section 4.5.4 adds a new vertex and a new edge within the clearance of (ϕ, υ) to the search tree (see Lemma 4.33 and Lemma 4.35 in the forthcoming Section 4.6.4).

Therefore, the probability that both $E1$ and $E2$ occur, which is denoted as E, resulting in adding a new vertex and a new edge within the clearance of (ϕ, υ) to the search tree, is positive. By the truncation operation in Definition B.11, the compact motion plan (ϕ, υ) is discretized into a finite number, say $m \in \mathbb{N}$, of segments. Each of those segments can be approximated by a solution pair associated with an edge that was added when the event E occurs. Therefore, if E occurs less than m times, then HyRRT will fail

to generate a motion plan that is close to (ϕ, υ). Proposition 4.37 in the forthcoming Section 4.6.5 demonstrates that the probability that E occurs less than m times, leading to the failure of HyRRT, approaches zero as the number of iterations increases, thereby establishing the property in Theorem 4.30. A proof of Theorem 4.30 following these steps is given in the forthcoming Section 4.6.6.

4.6.4 Probabilistic Guarantees on the Function Calls `nearest_neighbor` and `new_state`

We first characterize the probability that a vertex in the search tree that is close to an existing motion plan is selected as v_{cur} by the function call `nearest_neighbor` in Algorithm 2.

Lemma 4.31. *Given a hybrid dynamical system $\mathcal{H} = (C, f, D, g)$ with state $x \in \mathbb{R}^n$ and some $\delta \in \mathbb{R}_{>0}$, let $x_c \in \mathbb{R}^n$ be such that $x_c + \delta\mathbb{B} \subset S$, where S is either C' or D'. Suppose that there exists a vertex v in the search graph $\mathcal{T} = (V, E)$ such that $\overline{x}_v \in x_c + 2\delta/5\mathbb{B}$. Denote by v_{cur} the return of the function call `nearest_neighbor` in Algorithm 3. When HyRRT is executed uniformly as defined in Definition 4.18, the probability that $\overline{x}_{v_{cur}} \in x_c + \delta\mathbb{B}$ is at least $\frac{\zeta_n(\delta/5)^n}{\mu(S)}$, where ζ_n is given in (0.1) and $\mu(S)$ denotes the Lebesgue measure of set S.*

Proof. See Appendix B.4. □

Remark 4.32. *Lemma 4.31 shows that given $x_c \in \mathbb{R}^n$, when there exists a vertex v such that $\overline{x}_v \in x_c + 2\delta/5\mathbb{B}$, then the probability that the function call `nearest_neighbor`*

selects a vertex that is close to x_c is bounded from below by a positive constant. This lemma is used to provide a positive lower bound over the probability that a vertex that is close enough to the motion plan is returned by the function `nearest_neighbor` *in Algorithm 2.* ▲

The following lemma characterizes the probability that, given an initial state near a specific motion plan as input to the function call `new_state`, the simulated solution pair in the flow regime is within the clearance of this motion plan.

Lemma 4.33. *Given a hybrid dynamical system $\mathcal{H}$ with state $x \in \mathbb{R}^n$ and input $u \in \mathbb{R}^m$ that satisfies Assumption 4.24 and an input library that satisfies item 1 of Definition 4.20, let $\psi = (\phi, \upsilon)$ be a purely continuous solution pair to $\mathcal{H}$ with clearance $\delta > 0$, $(\tau, 0) = \max dom\ \psi$, and constant input function υ. Suppose that $\tau \in (0, T_m]$, where T_m comes from item 1 in Definition 4.20. Let $\psi_{new} = (\phi_{new}, \upsilon_{new})$ be the purely continuous solution pair generated by (4.9) in the function call* `new_state` *and initial state $\overline{x}_{v_{cur}} = \phi_{new}(0,0) \in \phi(0,0) + \kappa_1 \delta \mathbb{B}$ for some $\kappa_1 \in (0, 1/2)$. When HyRRT is executed uniformly as defined in Definition 4.18, for each $\kappa_2 \in (2\kappa_1, 1)$ and each $\epsilon \in (0, \frac{\kappa_2 \delta}{2})$, there exists $p_t \in (0, 1]$ such that*

$$Prob[E_1 \& E_2] \geq p_t \frac{\zeta_n \left(\max \left\{ \min \left\{ \frac{\frac{\kappa_2 \delta}{2} - \epsilon - \exp(K_x^f \tau)\kappa_1 \delta}{K_u^f \tau \exp(K_x^f \tau)}, \delta \right\}, 0 \right\} \right)^m}{\mu(U_C)} \tag{4.22}$$

where

1. *E_1 denotes the probabilistic event that ϕ and ϕ_{new} are $(\overline{\tau}, \kappa_2 \delta)$-close, where $(\tau', 0) = \max dom\ \phi_{new}$ and $\overline{\tau} = \max\{\tau, \tau'\}$;*

2. *E_2 denotes the probabilistic event that $x_{new} = \phi_{new}(\tau', 0) \in \phi(\tau, 0) + \kappa_2\delta\mathbb{B}$, where x_{new} stores the final state of ϕ_{new} in the function call* `new_state` *as is introduced in Section 4.5.4,*

ζ_n is given in (0.1), $\mu(U_C)$ denotes the Lebesgue measure of U_C, and K_x^f and K_u^f come from Assumption 4.24.

Proof. See Appendix B.5. □

Remark 4.34. *Since $\kappa_1 \in (0, 1/2)$, $\kappa_2 \in (2\kappa_1, 1)$ and $\delta > 0$, therefore, we have*

$$\kappa_1\delta < \frac{1}{2}\delta \tag{4.23a}$$

$$\kappa_2\delta < \delta, \tag{4.23b}$$

ensuring that there is no intersection between ψ_{new} and X_u, which prevents the function call `new_state` *from returning ψ_{new}, as stated in Section 4.5.4.*

To ensure that Lemma 4.33 provides a positive lower bound, the following additional requirement is imposed on κ_1 and κ_2:

$$\frac{\kappa_2}{2} > \kappa_1. \tag{4.24}$$

Then, then there exists $\epsilon \in (0, \frac{\kappa_2\delta}{2})$ satisfying

$$\epsilon < \frac{\kappa_2\delta}{2} - \exp(K_x^f\tau)\kappa_1\delta \tag{4.25}$$

to guarantee that the first argument in the min *operator in (4.22) is positive. From (4.25) and $\epsilon > 0$, any $\tau \in \mathbb{R}_{>0}$ satisfying the following condition suffices for the existence of ϵ*

satisfying (4.25):

$$\tau < \frac{\ln \frac{\kappa_2}{2\kappa_1}}{K_x^f} \tag{4.26}$$

where, by (4.24), $\frac{\ln \frac{\kappa_2}{2\kappa_1}}{K_x^f} > \frac{\ln \frac{2\kappa_1}{2\kappa_1}}{K_x^f} = 0$.

Therefore, if κ_1 *and* κ_2 *adhere to the constraints in (4.23a), (4.23b) and (4.24) and if* κ_1, κ_2, *and* τ *satisfy the conditions in (4.26), then the lower bound in (4.22) is guaranteed to be positive.*

The following result characterizes the probability that, given an initial state near a specific motion plan as input to the function call `new_state`, the simulated solution pair computed by the function call `new_state` in the jump regime is within the clearance of this motion plan.

Lemma 4.35. *Given a hybrid dynamical system* $\mathcal{H}$ *with state* $x \in \mathbb{R}^n$ *and input* $u \in \mathbb{R}^m$ *that satisfies Assumption 4.27 and an input library that satisfies item 2 of Definition 4.20, let* $\psi = (\phi, \upsilon)$ *be a purely discrete solution pair to* $\mathcal{H}$ *with a single jump, i.e.,* $\max dom\ \psi = (0, 1)$ *and clearance* $\delta > 0$. *Let* $\psi_{new} = (\phi_{new}, \upsilon_{new})$ *be the purely discrete solution pair generated by (4.10) in the function call* `new_state` *and initial state* $\overline{x}_{v_{cur}} = \phi_{new}(0, 0) \in \phi(0, 0) + \kappa_1 \delta \mathbb{B}$ *for some positive* $\kappa_1 \in (0, 1)$. *When HyRRT is executed uniformly as defined in Definition 4.18, for each* $\kappa_2 \in (0, 1)$, *we have that*

$$Prob[E] \geq \frac{\zeta_n \left(\max \left\{ \min \left\{ \frac{(\kappa_2 - K_x^g \kappa_1)\delta}{K_u^g}, \delta \right\}, 0 \right\} \right)^m}{\mu(U_D)} \tag{4.27}$$

where E *denotes the probabilistic event that* $x_{new} = \phi_{new}(0, 1) \in \phi(0, 1) + \kappa_2 \delta \mathbb{B}$, x_{new} *stores the final state of* ϕ_{new} *in the function call* `new_state` *as is introduced in Section*

4.5.4, ζ_n is given in (0.1), $\mu(U_D)$ denotes the Lebesgue measure of U_D, and K_x^g and K_u^g come from Assumption 4.27.

Proof. See Appendix B.6. □

Remark 4.36. *Since $\kappa_1 \in (0, 1]$, $\kappa_2 \in (0, 1]$ and $\delta > 0$, therefore, we have*

$$\kappa_1 \delta < \delta \tag{4.28a}$$

$$\kappa_2 \delta < \delta, \tag{4.28b}$$

ensuring that there is no intersection between ψ_{new} and X_u, which prevents the function call `new_state` *from returning ψ_{new}.*

To ensure that the lower bound in (4.27) is positive, an additional requirement is imposed on κ_1 and κ_2:

$$\frac{\kappa_2}{\kappa_1} > K_x^g. \tag{4.29}$$

Since this implies that $\frac{(\kappa_2 - K_x^g \kappa_1)\delta}{K_u^g} > 0$, the first argument of the min *operator in (4.27) is positive. Therefore, if κ_1 and κ_2 adhere to the constraints in (4.28a), (4.28b), and (4.29), then the lower bound in (4.27) is guaranteed to be positive.*

Note that if $\phi_{new}(0, 0) \in \phi(0, 0) + \kappa_1 \delta \mathbb{B}$ and $\phi_{new}(0, 1) \in \phi(0, 1) + \kappa_2 \delta \mathbb{B}$ are satisfied, ϕ and ϕ_{new} are $(1, \max\{\kappa_1 \delta, \kappa_2 \delta\})$-close. ▲

4.6.5 Probabilistic Completeness Guarantee of Finding a Motion Plan with Positive Clearance

We then establish a preliminary result demonstrating that, if there exists a motion plan to $\mathcal{P}$ with positive clearance and all assumptions outlined in Section 4.6.2

are met, HyRRT is probabilistically complete.

Proposition 4.37. *Given a motion planning problem* $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$*, suppose that Assumptions 4.22, 4.24, and 4.27 are satisfied, and there exists a compact motion plan* (ϕ, υ) *to* $\mathcal{P}$ *with clearance* $\delta > 0$ *and piecewise-constant input. Then, using a complete input library and when executed uniformly as defined in Definition 4.18, the probability that HyRRT fails to find a motion plan* $\psi' = (\phi', \upsilon')$ *such that* ϕ' *is* $(\tilde{\tau}, \delta)$*-close to* ϕ *after* k *iterations is at most* ae^{-bk}*, where* $a, b \in \mathbb{R}_{>0}$*,* $\tilde{\tau} = \max\{T + J, T' + J'\}$*,* $(T, J) = \max dom\ \psi$*, and* $(T', J') = \max dom\ \psi'$*.*

Proof. To show the claim, first, given a compact motion plan (ϕ, υ) to $\mathcal{P}$ with clearance $\delta > 0$, we construct a sequence of hybrid time instances

$$\mathbb{T} := \{(T_i, J_i) \in \text{dom } (\phi, \upsilon)\}_{i=0}^{m} \tag{4.30}$$

for some $m \in \mathbb{R}_{\geq 0} \backslash \{0\}$ to be chosen later. Alongside, we construct a corresponding geometric sequence[3] of positive real numbers

$$\mathbb{D} := \{r_i \in \mathbb{R}_{>0}\}_{i=0}^{m}, \tag{4.31}$$

with positive common ratio $q \in \mathbb{R}_{>0}$, in which case,

$$r_i := q^i r_0 \quad \forall i \in \{0, 1, ..., m\}.$$

Later in this proof, for each $i \in \{0, 1, ..., m-1\}$, Lemmas 4.31, 4.33, and 4.35 are utilized to provide positive lower bounds for their respective probabilistic events

[3] A geometric sequence is a sequence of non-zero numbers where each term after the first is found by multiplying the previous one by a fixed, non-zero number called the common ratio.

for the balls $\phi(T_i, J_i) + r_i\mathbb{B}$ and $\phi(T_{i+1}, J_{i+1}) + r_{i+1}\mathbb{B}$, where $(T_i, J_i), (T_{i+1}, J_{i+1}) \in \mathbb{T}$ and $r_i, r_{i+1} \in \mathbb{D}$. This requires that $\phi(T_i, J_i) + r_i\mathbb{B}$ and $\phi(T_{i+1}, J_{i+1}) + r_{i+1}\mathbb{B}$ meet each condition to establish these positive lower bounds.

The construction of $\mathbb{D}$ starts with selecting a proper common ratio q and the initial term r_0 in $\mathbb{D}$. To ensure that the positive lower bounds stated in Lemmas 4.31, 4.33, and 4.35 hold, the geometric sequence $\mathbb{D}$ must meet the following conditions:

D1) For each $i \in \{0, 1, ..., m-1\}$, $\frac{5}{2}r_i < r_{i+1}$ to meet the requirements for Lemma 4.31;

D2) For each $i \in \{0, 1, ..., m-1\}$, $2r_i < \min\{\delta, r_{i+1}\}$ to meet the conditions for Lemma 4.33 in (4.23a) and (4.24) and the condition for Lemma 4.35 in (4.28). In addition, it is required that $r_m < \delta$ as per (4.23b);

D3) For each $i \in \{0, 1, ..., m-1\}$, the condition $K_x^g r_i < r_{i+1}$ is essential to comply with the prerequisites for Lemma 4.35 in (4.29), where K_x^g comes from Assumption 4.27 and is fixed;

D4) $\sum_{i=0}^{m} r_i < \delta$, thereby ensuring, as per Proposition B.10, that the concatenation of all the ϕ_{new}'s, which are returned by the function call `new_state` in Section 4.5.4 and satisfy the properties introduced in probabilistic events in Lemmas 4.33 and 4.35, remains within the clearance δ.

To satisfy D1 - D3, the common ratio q is selected as any positive number satisfying the following inequality:

$$q > \max\left\{\frac{5}{2}, 2, K_x^g\right\} = \max\left\{\frac{5}{2}, K_x^g\right\}$$

which inherently implies $q > \frac{5}{2}$. Given that $q > \frac{5}{2} > 1$, the sequence $\mathbb{D}$ is, therefore, monotonically increasing. With the selected q, the initial term in $\mathbb{D}$ is selected as

$$r_0 = \frac{\delta}{q^{m+1}},$$

ensuring that, for each $i \in \{0, 1, ..., m-1\}$, $2r_i < 2r_{m-1} = \frac{2\delta}{q^2} < \frac{8\delta}{25} < \delta$ and $r_m = \frac{\delta}{q} < \frac{2\delta}{5} < \delta$. Furthermore, the sum[4] of $\mathbb{D}$, given by

$$\sum_{i=0}^{m} r_i = \frac{\delta}{q^{m+1}(1-q)} - \frac{\delta}{1-q} = \frac{\delta}{q-1}\left(1 - \frac{1}{q^{m+1}}\right) < \frac{\delta}{q-1}$$

where $q > \frac{5}{2}$, demonstrates that $\sum_{i=0}^{m} r_i < \frac{\delta}{\frac{5}{2}-1} = \frac{2\delta}{3} < \delta$, thereby satisfying D4. Therefore, this selection method satisfies D1 - D4 for each $m \in \mathbb{R}_{\geq 0}\backslash\{0\}$, with the specific value of m to be selected subsequently.

The sequence $\mathbb{T}$ is constructed such that for each $i \in \{0, 1, ..., m-1\}$, (T_i, J_i) and (T_{i+1}, J_{i+1}) satisfy one of the following two conditions:

T1) $T_i = T_{i+1}$ and $J_{i+1} = J_i + 1$, indicating a jump from (T_i, J_i) to (T_{i+1}, J_{i+1});

T2) $T_{i+1} - T_i = \Delta t'$ and $J_{i+1} = J_i$ for some $\Delta t' = \frac{\Delta t}{k}$, implying a purely continuous evolution from (T_i, J_i) to (T_{i+1}, J_{i+1}), where Δt is the resolution of the piecewise-constant input as in Definition 4.16, T_m is the upper bound for the time duration of the input signals in the complete $\mathcal{U}_C$ as outlined in Definition 4.20, and $k \in \mathbb{R}_{\geq 0}\backslash\{0\}$ is chosen large enough such that $\Delta t' = \frac{\Delta t}{k}$ is small enough to satisfy $\Delta t' \leq \frac{\ln \frac{q}{2}}{K_x^f}$, thereby satisfying (4.26), and $\Delta t' \leq T_m$ to establish a positive lower bound in Lemma 4.33.

[4]The closed-form of the sum of a geometric sequence is $\sum_{i=0}^{m} r_i = \frac{r_0(1-q^{m+1})}{1-q}$

Since υ is piecewise-constant, as per Definition 4.16, for any $k \in \mathbb{R}_{\geq 0} \backslash \{0\}$, $\Delta t' = \frac{\Delta t}{k}$ is also a resolution of the piecewise-constant input υ. Furthermore, Definition 4.17 implies that if $(t, j) \in \operatorname{dom} \upsilon$, $(t, j+1) \notin \operatorname{dom} \upsilon$ and $t + \Delta t' \leq \max_t \operatorname{dom} \upsilon$, it follows that $(t + \Delta t', j) \in \operatorname{dom} (\phi, \upsilon)$; see (2.5) for the definition of $\max_t$. Therefore, it is indeed feasible to construct $\mathbb{T}$ by sweeping through the hybrid time domain of (ϕ, υ) as follows: Starting from the initial assignment $i \leftarrow 0$ and $(T_i, J_i) \leftarrow (0, 0)$,

1. if $(T_i, J_i + 1) \in \operatorname{dom} (\phi, \upsilon)$, it follows that $(T_{i+1}, J_{i+1}) \leftarrow (T_i, J_i + 1)$, which satisfies T1;

2. if $(T_i, J_i + 1) \notin \operatorname{dom} (\phi, \upsilon)$, it follows that $(T_{i+1}, J_{i+1}) \leftarrow (T_i + \Delta t', J_i)$, which satisfies T2.

This process is continued by incrementing $i \leftarrow i + 1$ until $(T_i, J_i) = \max \operatorname{dom} (\phi, \upsilon)$. This construction of $\mathbb{T}$ results in

$$m = \frac{T}{\Delta t'} + J + 1$$

where $(T, J) = \max \operatorname{dom} (\phi, \upsilon)$.

With this construction, the motion plan between (T_i, J_i) and (T_{i+1}, J_{i+1}) is either purely continuous or purely discrete, which allows the use of Lemmas 4.33 and 4.35 in each respective regime. Therefore, the construction of $\mathbb{T}$ and $\mathbb{D}$ adheres to all the conditions required to apply Lemmas 4.31, 4.33, and 4.35.

Next, we partition the compact motion plan into a finite number of segments. By demonstrating that the probability of HyRRT generating ϕ_{new} close to each segment

is positive, we establish that the probability of HyRRT failing to find a motion plan converges to zero as the number of iterations approaches infinity. Using the truncation and translation operations defined in Definition B.11, for each $i \in \{0, 1, ..., m-1\}$, let (ϕ_i, υ_i) represent the truncation of (ϕ, υ) between (T_i, J_i) and (T_{i+1}, J_{i+1}) following the translation by (T_i, J_i). Given that (T_i, J_i) and (T_{i+1}, J_{i+1}) satisfy T1 or T2, it follows that each element in $\{\phi_i\}_{i=0}^{m-1}$ is either purely continuous with a constant input or purely discrete with a single jump. Then, with the search tree denoted $\mathcal{T} = (V, E)$, the proof proceeds by showing that the probability of each of the following probabilistic events is positive:

P1) The function call $\mathcal{T}.\texttt{init}(X_0)$ in Section 4.5.1 adds a vertex associated with some $x_0 \in \phi(T_0, J_0) + \frac{2}{5} r_0 \mathbb{B}$ to $\mathcal{T}$.

P2) For each $i \in \{0, 1, , m-1\}$, given the existence of a vertex $v \in V$ such that $x_v \in \phi(T_i, J_i) + \frac{2}{5} r_i \mathbb{B}$, the function call `extend` in Algorithm 3 adds a new vertex to $\mathcal{T}$ associated with x_{new}, along with a corresponding new edge to $\mathcal{T}$ associated with $\psi_{new} = (\phi_{new}, \upsilon_{new})$, such that

PE1) $x_{new} \in \phi(T_{i+1}, J_{i+1}) + \frac{2}{5} r_{i+1} \mathbb{B}$;

PE2) ϕ_{new} is $(\overline{\tau}, r_{i+1})$-close to ϕ_i, where[5]

$$\overline{\tau} = \begin{cases} \max\{\max_t \text{dom } \phi_{new}, \max_t \text{dom } \phi_i\} & \text{if } \phi_i \text{ is purely continuous} \\ 1 & \text{if } \phi_i \text{ is purely discrete.} \end{cases}$$

[5]Given that ϕ_i is purely continuous, by T2 and Definition B.11, it follows that $\max_t \text{dom } \phi_i = \Delta t'$

The process of HyRRT in finding a motion plan can be viewed as a Bernoulli trial[6] (P1) following repeated Bernoulli trials (P2). Denote the probability that the probabilistic event P1 occurs as p_{init} and that P2 occurs as p_{extend}. Next, we show that p_{init} and p_{extend} are positive.

1. Given that HyRRT is executed uniformly (see Definition 4.18), by (4.21) and $\mu(\phi(T_0, J_0)+\frac{2}{5}r_0\mathbb{B}) > 0$, where $\mu(\phi(T_0, J_0)+\frac{2}{5}r_0\mathbb{B})$ represents the Lebesgue measure of the ball $\phi(T_0, J_0) + \frac{2}{5}r_0\mathbb{B}$, it follows that $p_{init} > 0$.

2. The function calls `nearest_neighbor` and `new_state` are executed in the function call `extend`. We first show the probability that each of `nearest_neighbor` and `nearest_neighbor` contributes to an output of `extend` satisfying PE1 and PE2 is positive:

   (a) Given the existence of a vertex $v \in V$ such that $x_v \in \phi(T_i, J_i) + \frac{2}{5}r_i\mathbb{B}$, Lemma 4.31 implies that the probability that the function call `nearest_neighbor` returns a vertex v_{cur} such that $\overline{x}_{v_{cur}} \in \phi(T_i, J_i) + r_i\mathbb{B}$, denoted p_{near}, is positive.

   (b) Under the condition that $\overline{x}_{v_{cur}} \in \phi(T_i, J_i) + r_i\mathbb{B}$ and $r_i < \frac{2}{5}r_{i+1}$ ensured by D1, the probability that the function call `new_state` generates a new solution pair $\psi_{new} = (\phi_{new}, \upsilon_{new})$ satisfying PE1 and PE2, denoted p_{new}, is positive. This property is guaranteed by Lemma 4.33, if ϕ_i is purely continuous and by Lemma 4.35 if ϕ_i is purely discrete.

[6] In the theory of probability and statistics, a Bernoulli trial is a random experiment with exactly two possible outcomes, success and failure.

Therefore, since p_{near} and p_{new} are positive, it follows that $p_{extend} := p_{near}p_{new}$ is positive.

Given that the probability of P2 occurring is positive, and denoting the total number of occurrences of P2 trials as X_n, where $n \in \mathbb{N}\backslash\{0\}$, we characterize the probability of HyRRT failing to identify a motion plan, which is equivalent to P2 occurring fewer than m times, as follows:

$$\lim_{n\to\infty} \text{Prob}[X_n < m] = \lim_{n\to\infty} \sum_{i=0}^{m-1} \binom{n}{i} p_{extend}^{i}(1-p_{extend})^{n-i}. \tag{4.32}$$

Since $i \in \{0, 1, ..., m-1\}$, we have $0 \leq i \leq m-1$. As $n \to \infty$, there exists an $N \in \mathbb{N}$ such that for all $n \geq N$, we have $0 \leq i \leq m-1 \leq \frac{n}{2}$. Such an N can be chosen as $2(m-1)$. Due to the monotonicity of $\binom{n}{k}$ for k from 0 to $\frac{n}{2}$ and $0 \leq i \leq m-1 \leq \frac{n}{2}$, we have $\binom{n}{i} \leq \binom{n}{m-1}$. Hence,

$$\begin{aligned} \lim_{n\to\infty} \text{Prob}[X_n < m] &\leq \lim_{n\to\infty} \sum_{i=0}^{m-1} \binom{n}{m-1} p_{extend}^{i}(1-p_{extend})^{n-i} \\ &\leq \lim_{n\to\infty} \binom{n}{m-1} \sum_{i=0}^{m-1} p_{extend}^{i}(1-p_{extend})^{n-i}. \end{aligned} \tag{4.33}$$

Define $p := \max\{p_{extend}, 1 - p_{extend}\} \in (0, 1)$. It follows that

$$\begin{aligned} \lim_{n\to\infty} \text{Prob}[X_n < m] &\leq \lim_{n\to\infty} \begin{pmatrix} n \\ \\ m-1 \end{pmatrix} \sum_{i=0}^{m-1} p^i p^{n-i} \\ &\leq \lim_{n\to\infty} \begin{pmatrix} n \\ \\ m-1 \end{pmatrix} \sum_{i=0}^{m-1} p^n. \end{aligned} \tag{4.34}$$

Since p^n is not related to i, we have $\sum_{i=0}^{m-1} p^n = mp^n$. It follows that

$$\begin{aligned} \lim_{n\to\infty} \text{Prob}[X_n < m] &\leq \lim_{n\to\infty} \begin{pmatrix} n \\ \\ m-1 \end{pmatrix} mp^n \\ &\leq \lim_{n\to\infty} \frac{n!}{(m-1)!(n-m+1)!} mp^n. \end{aligned} \tag{4.35}$$

Since $\frac{n!}{(n-m+1)!} = \Pi_{i=n-m+2}^{n} i \leq \Pi_{i=n-m+2}^{n} n = n^{m-1}$, it follows that

$$\lim_{n\to\infty} \text{Prob}[X_n < m] \leq \lim_{n\to\infty} \frac{n^{m-1}}{(m-1)!} mp^n. \tag{4.36}$$

Since $p \in (0, 1)$ and m is finite, then $\lim_{n\to\infty} \frac{n^{m-1}}{(m-1)!} mp^n = 0$. Therefore, the probability that HyRRT fails to find a motion plan is converging to zero as the number of iterations approaches infinity.

In addition, Proposition B.10 establishes that the concatenation of all the ϕ_{new}'s satisfying PE2, which is returned by HyRRT as ϕ', is $(\tilde{\tau}, \sum_{i=0}^{m} r_i)$-close to $\phi = \phi_0|\phi_1|...|\phi_m$, where $\sum_{i=0}^{m} r_i < \delta$ is guaranteed by D4.

□

4.6.6 Proof of Theorem 4.30

Given that ψ is a motion plan to $\mathcal{P}$ with safety clearance $\delta_s > 0$ and that $\delta_f > 0$, Lemma 4.15 establishes that ψ is a motion plan to $\mathcal{P}_\delta$ with clearance $\delta = \min\{\delta_s, \delta_f\} > 0$. Furthermore, by Proposition 4.37, it follows that the probability that HyRRT fails to find $\psi' = (\phi', \upsilon')$ is at most ae^{-bk} and the generated ϕ' is $(\tilde{\tau}, \tilde{\delta})$-close to ϕ where $\tilde{\tau} = \max\{T + J, T' + J'\}$, $(T, J) = \max \operatorname{dom} \psi$, $(T', J') = \max \operatorname{dom} \psi'$, and $\tilde{\delta} = \min\{\delta_s, \delta_f\}$.

4.7 HyRRT Software Tool for Motion Planning for Hybrid Dynamical Systems and Examples

Algorithm 2 leads to a software tool[7] to solve the motion planning problems for hybrid dynamical systems. This software only requires the motion planning problem data $(X_0, X_f, X_u, (C, f, D, g))$, an input library $(\mathcal{U}_C, \mathcal{U}_D)$, a tunable parameter $p_n \in (0, 1)$, an upper bound K over the iteration number and two constraint sets X_c and X_d. The tool is illustrated in Examples 3.2 and 3.3.

Example 4.38 (Actuated bouncing ball system in Example 3.2, revisited)**.** *This example serves to demonstrate that the HyRRT algorithm is proficient in solving the specific instance of the motion planning problem as illustrated in Example 3.2. The inputs fed to the proposed algorithm are given as follows:*

1. *The tune parameter p_n is set as* 0.5.

[7] Code at https://github.com/HybridSystemsLab/hybridRRT.

2. *The upper bound K is set to* 1000.

3. *The construction of the input library* ($\mathcal{U}_C$,$\mathcal{U}_D$) *adheres to the methodology delineated in* **Input Library Construction Procedure**, *as referenced in Section 4.2. Here, the upper bound T_m is explicitly configured to* 0.1. *From the setting of X_u, It is imperative to note that each input falling outside the interval* $(0, 5)$ *is unsafe. As a result, both U_C^s and U_D are restricted to* $(0, 5)$. *Consequently, both $\mathcal{U}_C$ and $\mathcal{U}_D$ are constructed in compliance with the specifications outlined in the* **Input Library Construction Procedure**.

4. *The constraint sets X_c and X_d are set as $X_c = C'$ and $X_d = D'$.*

5. *The tolerance ϵ in (4.11) is set to* 0.2.

The simulation result is shown in Figure 4.3. The simulation is implemented in MATLAB software and processed by a 2.2 *GHz Intel Core i7 processor. On average, over the course of* 20 *runs, the simulation is observed to require approximately* 0.72 *seconds for completion and results in the generation of* 34.2 *vertices during the propagation phase.*

Example 4.39 (Walking robot system in Example 3.3, revisited)**.** *The settings for HyRRT planner are given as follows:*

1) The tune parameter p_n is set as 0.9 *to encourage the flow regime.*

2) The upper bound K is set as 2000.

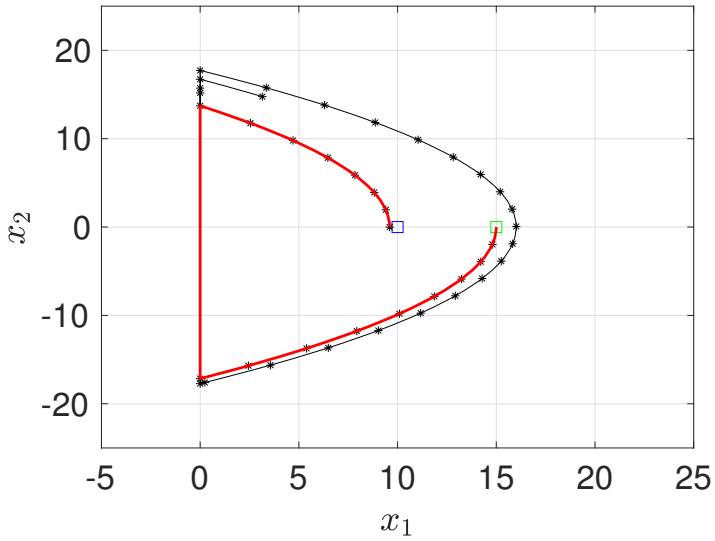


Figure 4.3: The above shows the search results of HyRRT algorithm to solve the sample motion planning problem in Example 3.2. Green square denotes X_0 and blue square denotes X_f. The states represented by vertices in the search tree are denoted by $\star$'s. The lines between $\star$'s denote the state trajectories of the solution pairs associated with edges in the search tree. The red trajectory denotes the state trajectory of a motion plan to the given motion planning problem.

3) The input library $(\mathcal{U}_C, \mathcal{U}_D)$ *is formulated in accordance with* **Input Library Construction Procedure**. *Specifically, the upper bound* T_m *is set to* 0.4. *Given the constraints imposed by* X_u, *each input deviating from the set* $(-3,3) \times (-3,3) \times (-0.2,0.2)$ *is classified as unsafe. Consequently, both* U_C^s *and* U_D *are restricted to* $(-3,3) \times (-3,3) \times (-0.2,0.2)$.

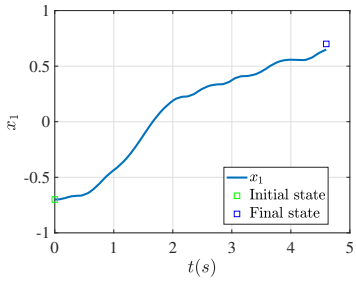
4) The constraint set X_c *is chosen as* $\{(x,a) \in \mathbb{R}^6 \times \mathbb{R}^3 : h(x) \geq -s\}$ *and* X_d *as* $\{(x,a) \in \mathbb{R}^6 \times \mathbb{R}^3 : h(x) = 0, \omega_p \geq -s\}$ *with a tunable parameter* s *set to* 0, 0.3, 0.5, 1, *and* 2, *such that* $C = X_c|_{s=0} \subsetneq X_c|_{s=0.3} \subsetneq X_c|_{s=0.5} \subsetneq X_c|_{s=1} \subsetneq X_c|_{s=2}$ *and* $D = X_d|_{s=0} \subsetneq X_d|_{s=0.3} \subsetneq X_d|_{s=0.5} \subsetneq X_d|_{s=1} \subsetneq X_d|_{s=2}$.

5) The input library $(\mathcal{U}_C, \mathcal{U}_D)$ *is constructed as follows. In this illustration,* U_C^s *is constructed as* $U_C^s = \{-2.0, -1.0, 0.0, 1.0, 2.0\} \times \{-2.0, -1.0, 0.0, 1.0, 2.0\} \times \{-0.15, -0.0875, -0.0250, 0.0375, 0.10\}$. *There are* 125 *elements in* U_C^s. *For each* $u^s \in U_C^s$, *an input signal* $[0, 0.2] \to \{u^s\}$ *is constructed and added to* $\mathcal{U}_C$. *In the biped system, since input has no effect on the jump, then* $\mathcal{U}_D$ *is constructed as* $\{(0,0,0)\}$.

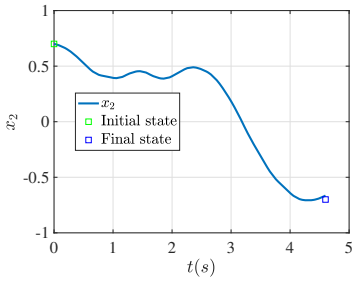
6) The tolerance ϵ *in (4.11) is set to* 0.3

The simulation result in Figure 4.4 shows that HyRRT is able to solve the instance of motion planning problem for the walking robot. The simulation is implemented in MATLAB and processed by a 3.5 *GHz Intel Core i5 processor. Over the course of* 20 *runs, on average, the simulation takes* $63.26/78.3/100.4/183.7/280.8$ *seconds with* s *set to* $0/0.3/0.5/1.0/2.0$, *respectively. Among all the runs conducted, the simulation takes a minimum of* 31.2 *seconds to complete. In contrast, when employing the forward/backward*

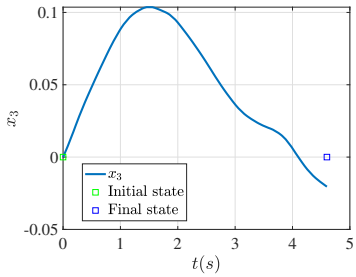
propagation algorithm based on breadth-first search operating under identical settings, the motion planner takes 1608.2 *seconds to solve the same problem. The results demonstrate a noteworthy improvement provided by the rapid exploration technique, with average and fastest run computation time improvements of* 96.1% *and* 98.1%*, respectively. It is also observed that as the sets* X_c *and* X_d *grow, HyRRT considers more vertices in solving Problem 4.5 leading to higher computation time.*



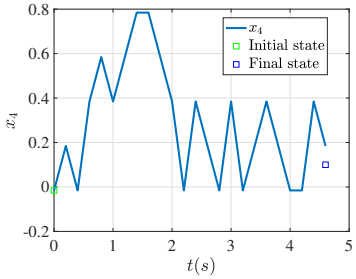
(a) The trajectory of x_1 component of the generated motion plan.



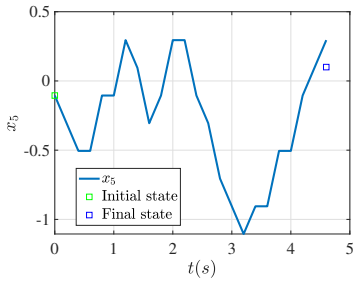
(b) The trajectory of x_2 component of the generated motion plan.



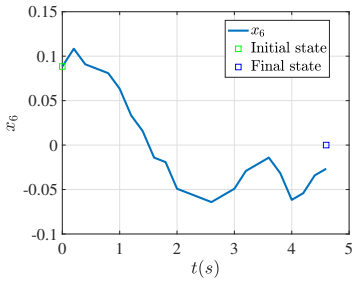
(c) The trajectory of x_3 component of the generated motion plan.



(d) The trajectory of x_4 component of the generated motion plan.



(e) The trajectory of x_5 component of the generated motion plan.



(f) The trajectory of x_6 component of the generated motion plan.

Figure 4.4: The above shows the trajectories of each state components of the generated motion plan to the sample motion planning problem for biped system. In each figure in Figure 4.4, green square denotes the corresponding component of X_0 and blue square denotes the corresponding component of X_f.

Chapter 5

A Bidirectional Sampling-based Motion Planning Approach for Hybrid Dynamical Systems

5.1 Overview

In this chapter, a bidirectional RRT-type motion planning algorithm for hybrid systems, called HyRRT-Connect, is proposed. HyRRT-Connect searches for a motion plan by incrementally constructing two search trees: one starts from the initial state set and propagates forward in hybrid time, while the other starts from the final state set and propagates backward in hybrid time. Upon detecting overlaps between the two search trees, a connection is established, subsequently yielding a motion plan, which is elaborated in Section 5.3.

The proposed HyRRT-Connect algorithm requires a library of possible inputs

to construct $\mathcal{T}^{\text{fw}}$, denoted $\mathcal{U}^{\text{fw}} = (\mathcal{U}_C^{\text{fw}}, \mathcal{U}_D^{\text{fw}})$, and to construct $\mathcal{T}^{\text{bw}}$, denoted $\mathcal{U}^{\text{bw}} = (\mathcal{U}_C^{\text{bw}}, \mathcal{U}_D^{\text{bw}})$. The input library $\mathcal{U}^{\text{fw}}$ (respectively, $\mathcal{U}^{\text{bw}}$) includes the input signals for the flows of $\mathcal{H}^{\text{fw}}$ (respectively, $\mathcal{H}^{\text{bw}}$), collected in $\mathcal{U}_C^{\text{fw}}$ (respectively, $\mathcal{U}_C^{\text{bw}}$), and the input values for the jumps of $\mathcal{H}^{\text{fw}}$ (respectively, $\mathcal{H}^{\text{bw}}$), collected in $\mathcal{U}_D^{\text{fw}}$ (respectively, $\mathcal{U}_D^{\text{bw}}$).

HyRRT-Connect addresses the motion planning problem $\mathcal{P} = (X_0, X_f, X_u, (C^{\text{fw}}, f^{\text{fw}}, D^{\text{fw}}, g^{\text{fw}}))$ using input libraries $\mathcal{U}^{\text{fw}}$ and $\mathcal{U}^{\text{bw}}$ through the following steps:

Step 1: Sample a finite number of points from X_0 (respectively, X_f) and initialize a search tree $\mathcal{T}^{\text{fw}} = (V^{\text{fw}}, E^{\text{fw}})$ (respectively, $\mathcal{T}^{\text{bw}} = (V^{\text{bw}}, E^{\text{bw}})$) by adding vertices associated with each sampling point.

Step 2: Incrementally construct $\mathcal{T}^{\text{fw}}$ forward in hybrid time and $\mathcal{T}^{\text{bw}}$ backward in hybrid time, executing both procedures in an interleaved manner[1].

Step 3: If an appropriate overlap between $\mathcal{T}^{\text{fw}}$ and $\mathcal{T}^{\text{bw}}$ is found, reverse the solution pair in $\mathcal{T}^{\text{bw}}$, concatenate it to the solution pair in $\mathcal{T}^{\text{fw}}$ and return the concatenation result.

An appropriate overlap between $\mathcal{T}^{\text{fw}}$ and $\mathcal{T}^{\text{bw}}$ implies the paths in both trees that can collectively be used to construct a motion plan to $\mathcal{P}$. Details on these overlaps are discussed in Section 5.3.

[1]It is imperative to underscore that the information used in forward propagation remains unaffected by the backward propagation, and vice versa. This independence allows for the potential of executing both forward and backward propagation simultaneously, as opposed to an interleaved approach. Nonetheless, the nature of motion planning algorithm, which involves frequently joining the results computed in parallel to check overlaps between the forward and backward search trees, may prevent the computation improvement associated with parallel computation. More discussion and experiment results can be found in the forthcoming Section 5.5.

5.2 HyRRT-Connect Algorithm

The proposed algorithm is given in Algorithm 4. The inputs of Algorithm 4 are the problem $\mathcal{P} = (X_0, X_f, X_u, (C^{\text{fw}}, f^{\text{fw}}, D^{\text{fw}}, g^{\text{fw}}))$, the backward-in-time hybrid dynamical system $\mathcal{H}^{\text{bw}}$ obtained from (3.6), the input libraries $\mathcal{U}^{\text{fw}}$ and $\mathcal{U}^{\text{bw}}$, two parameters $p_n^{\text{fw}} \in (0,1)$ and $p_n^{\text{bw}} \in (0,1)$, which tune the probability of proceeding with the flow regime or the jump regime during the forward and backward construction, respectively, an upper bound $K \in \mathbb{N}_{>0}$ for the number of iterations to execute, and four tunable sets $X_c^{\text{fw}} \supset \overline{C^{\text{fw}'}}$, $X_d^{\text{fw}} \supset \overline{D^{\text{fw}'}}$, $X_c^{\text{bw}} \supset \overline{C^{\text{bw}'}}$ and $X_d^{\text{bw}} \supset \overline{D^{\text{bw}'}}$ where $C^{\text{fw}'}$, $C^{\text{bw}'}$, $D^{\text{fw}'}$ and $D^{\text{bw}'}$ are defined as in (2.3) and (2.4), which act as constraints in finding a closest vertex to x_{rand}. **Step** 1 in Section 5.1 corresponds to the function calls $\mathcal{T}^{\text{fw}}$.`init` and $\mathcal{T}^{\text{bw}}$.`init` in line 1 of Algorithm 4. The construction of $\mathcal{T}^{\text{fw}}$ in **Step** 2 is implemented in lines 3 - 10. The construction of $\mathcal{T}^{\text{bw}}$ in **Step** 2 is implemented in lines 11 - 18. The solution checking in **Step** 3 is executed depending on the return of the function call `extend` and will be further discussed in[2] Section 5.3. Each function in Algorithm 4 is defined next.

5.2.1 $\mathcal{T}$.`init`(X)

The $\mathcal{T}$.`init` function initializes the search tree $\mathcal{T} = (V, E)$ by randomly selecting points from set X, which can be either X_0 or X_f. For each sampling point $x_0 \in X$, a corresponding vertex v_0 is added to V, while no edges are added to E at this stage.

[2]The solution checking process in **Step** 3 is not reflected in Algorithm 4. This omission is intentional, as practitioners might devise varied termination conditions for the HyRRT-Connect algorithm.

Algorithm 4 HyRRT-Connect algorithm

Input: $X_0, X_f, X_u, \mathcal{H}^{\mathrm{fw}} = (C^{\mathrm{fw}}, f^{\mathrm{fw}}, D^{\mathrm{fw}}, g^{\mathrm{fw}}), \mathcal{H}^{\mathrm{bw}}$ $= (C^{\mathrm{bw}}, f^{\mathrm{bw}}, D^{\mathrm{bw}}, g^{\mathrm{bw}}), (\mathcal{U}_C, \mathcal{U}_D), p_n^{\mathrm{fw}}, p_n^{\mathrm{bw}} \in (0,1)$, $K \in \mathbb{N}_{>0}$, $X_c^{\mathrm{fw}} \supset \overline{C^{\mathrm{fw}'}}$, $X_d^{\mathrm{fw}} \supset \overline{D^{\mathrm{fw}'}}$, $X_c^{\mathrm{bw}} \supset \overline{C^{\mathrm{bw}'}}$ and $X_d^{\mathrm{bw}} \supset \overline{D^{\mathrm{bw}'}}$.

$\mathcal{T}^{\mathrm{fw}}$.`init`($X_0$), $\mathcal{T}^{\mathrm{bw}}$.`init`($X_f$)
for $k = 1$ to K **do**
  randomly select a real number r^{fw} from $[0, 1]$.
  if $r^{\mathrm{fw}} \leq p_n^{\mathrm{fw}}$ **then**
    $x_{rand}^{\mathrm{fw}} \leftarrow$ `random_state`($\overline{C^{\mathrm{fw}'}}$).
    `extend`($\mathcal{T}^{\mathrm{fw}}, x_{rand}^{\mathrm{fw}}, (\mathcal{U}_C^{\mathrm{fw}}, \mathcal{U}_D^{\mathrm{fw}}), \mathcal{H}^{\mathrm{fw}}, X_u, X_c^{\mathrm{fw}}$).
  else
    $x_{rand}^{\mathrm{fw}} \leftarrow$ `random_state`($D^{\mathrm{fw}'}$).
    `extend`($\mathcal{T}^{\mathrm{fw}}, x_{rand}^{\mathrm{fw}}, (\mathcal{U}_C^{\mathrm{fw}}, \mathcal{U}_D^{\mathrm{fw}}), \mathcal{H}^{\mathrm{fw}}, X_u, X_d^{\mathrm{fw}}$).
  end if
  randomly select a real number r^{bw} from $[0, 1]$.
  if $r^{\mathrm{bw}} \leq p_n^{\mathrm{bw}}$ **then**
    $x_{rand}^{\mathrm{bw}} \leftarrow$ `random_state`($\overline{C^{\mathrm{bw}'}}$).
    `extend`($\mathcal{T}^{\mathrm{bw}}, x_{rand}^{\mathrm{bw}}, (\mathcal{U}_C^{\mathrm{bw}}, \mathcal{U}_D^{\mathrm{bw}}), \mathcal{H}^{\mathrm{bw}}, X_u, X_c^{\mathrm{bw}}$).
  else
    $x_{rand}^{\mathrm{bw}} \leftarrow$ `random_state`($D^{\mathrm{bw}'}$).
    `extend`($\mathcal{T}^{\mathrm{bw}}, x_{rand}^{\mathrm{bw}}, (\mathcal{U}_C^{\mathrm{bw}}, \mathcal{U}_D^{\mathrm{bw}}), \mathcal{H}^{\mathrm{bw}}, X_u, X_d^{\mathrm{bw}}$).
  end if
end for

Algorithm 5 Extend function

1: **function** EXTEND($(\mathcal{T}, x, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u, X_*)$)

2: $v_{cur} \leftarrow$ `nearest_neighbor`$(x, \mathcal{T}, \mathcal{H}, X_*)$;

3: $(\texttt{is\_a\_new\_vertex\_generated}, x_{new}, \psi_{new}) \leftarrow$ `new_state`$(v_{cur}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u)$

4: **if** `is_a_new_vertex_generated` $= true$ **then**

5: $v_{new} \leftarrow \mathcal{T}$.`add_vertex`$(x_{new})$;

6: $\mathcal{T}$.`add_edge`$(v_{cur}, v_{new}, \psi_{new})$;

7: **return** `Advanced`;

8: **end if**

9: **return** `Trapped`;

10: **end function**

5.2.1.1 $x_{rand} \leftarrow$`random_state`(S)

The function call `random_state` randomly selects a point from the set $S \subset \mathbb{R}^n$. Rather than to select from $\overline{C'} \cup D'$, it is designed to select points from $\overline{C'}$ and D' separately depending on the value of r. The reason is that if $\overline{C'}$ (or D') has zero measure while D' (respectively, $\overline{C'}$) does not, the probability that the point selected from $\overline{C'} \cup D'$ lies in $\overline{C'}$ (respectively, D') is zero, which would prevent establishing probabilistic completeness.

5.2.2 $v_{cur} \leftarrow$`nearest_neighbor`$(x_{rand}, \mathcal{T}, \mathcal{H}, X_*^{\Delta})$

The function call `nearest_neighbor` searches for a vertex v_{cur} in the search tree $\mathcal{T} = (V, E)$ such that its associated state value has minimal distance to x_{rand}. This

function is implemented as solving the following optimization problem over $X_\star^\Delta$, where $\star$ is either c or d and Δ is either `fw` or `bw`.

Problem 5.1. *Given a hybrid dynamical system $\mathcal{H} = (C, f, D, g)$, $x_{rand} \in \mathbb{R}^n$, and a search tree $\mathcal{T} = (V, E)$, solve*

$$\begin{aligned} \arg\min_{v \in V} \quad & |\overline{x}_v - x_{rand}| \\ s.t. \quad & \overline{x}_v \in X_\star^\Delta. \end{aligned}$$

The data of Problem 5.1 comes from the arguments of the `nearest_neighbor` function call. This optimization problem can be solved by traversing all the vertices in V.

5.2.3 $(\texttt{is\_a\_new\_vertex\_generated}, x_{new}, \psi_{new}) \leftarrow \texttt{new\_state}(v_{cur}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H} = (C, f, D, g), X_u)$

If $\overline{x}_{v_{cur}} \in \overline{C'} \backslash D'$ ($\overline{x}_{v_{cur}} \in D' \backslash \overline{C'}$), the function call `new_state` generates a new solution pair ψ_{new} to hybrid dynamical system $\mathcal{H}$ starting from $\overline{x}_{v_{cur}}$ by applying a input signal $\tilde{u}$ (respectively, an input value u_D) randomly selected from $\mathcal{U}_C$ (respectively, $\mathcal{U}_D$). If $\overline{x}_{v_{cur}} \in \overline{C'} \cap D'$, then this function generates ψ_{new} by randomly selecting flows or jump. The final state of ψ_{new} is denoted as x_{new}. After ψ_{new} and x_{new} are generated, the function `new_state` evaluates if ψ_{new} is trivial. If it is, indicating no exploration of new space, addition to $\mathcal{T}$ is unnecessary, setting `is_a_new_vertex_generated` $\leftarrow$ `false`. The function call `new_state` then returns. Else, the function `new_state` finds $(t, j) \in \text{dom } \psi_{new}$ such that $\psi_{new}(t, j) \in X_u$, implying intersection with the unsafe

set, then it sets `is_a_new_vertex_generated` $\leftarrow$ `false`. If neither condition is met, `is_a_new_vertex_generated` $\leftarrow$ `true`.

5.2.4 $v_{new} \leftarrow \mathcal{T}$.add_vertex$(x_{new})$ and $\mathcal{T}$.add_edge$(v_{cur}, v_{new}, \psi_{new})$

The function call $\mathcal{T}$.add_vertex(x_{new}) adds a new vertex v_{new} associated with x_{new} to $\mathcal{T}$ and returns v_{new}. The function call $\mathcal{T}$.add_edge$(v_{cur}, v_{new}, \psi_{new})$ adds a new edge $e_{new} = (v_{cur}, v_{new})$ associated with ψ_{new} to $\mathcal{T}$.

5.3 Motion Plan Identification and Reconstruction

The following two scenarios are identified where a motion plan can be constructed by utilizing one path from $\mathcal{T}^{\text{fw}}$ and another from $\mathcal{T}^{\text{bw}}$:

S1) A vertex in $\mathcal{T}^{\text{fw}}$ is associated with the same state in the flow set as some vertex in $\mathcal{T}^{\text{bw}}$.

S2) A vertex in $\mathcal{T}^{\text{fw}}$ is associated with a state such that a forward-in-hybrid time jump from such state results in the state associated with some vertex in $\mathcal{T}^{\text{bw}}$, or conversely, a vertex in $\mathcal{T}^{\text{bw}}$ is associated with a state such that a backward-in-hybrid time jump from such state results in the state associated with some vertex in $\mathcal{T}^{\text{fw}}$.

These scenarios provide solution pairs that can be used to construct a motion plan by reversing a solution pair in $\mathcal{T}^{\text{bw}}$ and concatenating its reversal to a solution pair in $\mathcal{T}^{\text{fw}}$, as guaranteed by Lemma 3.17. In the HyRRT-Connect algorithm, each of these

scenarios is evaluated whenever an `Advanced` signal is returned by the `extend` function. However, the random selection of the inputs prevents the exact satisfaction of S1 in Assumption 3.15. Neglecting approximation errors due to numerical computation, it is typically possible to solve for an exact input at a jump from one state to an other, as required in S2. However, due to the random selection of the inputs and the family of signals used, satisfying S1 is not typically possible. This may lead to a discontinuity along the flow in the resulting motion plan. A reconstruction process is introduced below to address this issue. This process propagates forward in hybrid time from the state associated with the vertex identified in $\mathcal{T}^{\mathrm{fw}}$. Next, we present an implementation of identifying two paths from $\mathcal{T}^{\mathrm{fw}}$ and $\mathcal{T}^{\mathrm{bw}}$ in S1 and S2, respectively.

5.3.1 Same State Associated with Vertices in $\mathcal{T}^{\mathrm{fw}}$ and $\mathcal{T}^{\mathrm{bw}}$

In S1, HyRRT-Connect identifies if there exists a path

$$p^{\mathrm{fw}} := ((v_0^{\mathrm{fw}}, v_1^{\mathrm{fw}}), (v_1^{\mathrm{fw}}, v_2^{\mathrm{fw}}), ..., (v_{m-1}^{\mathrm{fw}}, v_m^{\mathrm{fw}})) =: (e_0^{\mathrm{fw}}, e_1^{\mathrm{fw}}, ..., e_{m-1}^{\mathrm{fw}}) \tag{5.1}$$

in $\mathcal{T}^{\mathrm{fw}}$, where $m \in \mathbb{N}$, and a path

$$p^{\mathrm{bw}} := ((v_0^{\mathrm{bw}}, v_1^{\mathrm{bw}}), (v_1^{\mathrm{bw}}, v_2^{\mathrm{bw}}), ..., (v_{n-1}^{\mathrm{bw}}, v_n^{\mathrm{bw}})) =: (e_0^{\mathrm{bw}}, e_1^{\mathrm{bw}}, ..., e_{n-1}^{\mathrm{bw}}) \tag{5.2}$$

in $\mathcal{T}^{\mathrm{bw}}$, where $n \in \mathbb{N}$, satisfying the following conditions:

C1) $\overline{x}_{v_0^{\mathrm{fw}}} \in X_0$,

C2) for $i \in \{0, 1, ..., m-2\}$, if $\overline{\psi}_{e_i^{\mathrm{fw}}}$ and $\overline{\psi}_{e_{i+1}^{\mathrm{fw}}}$ are both purely continuous, then $\overline{\psi}_{e_{i+1}^{\mathrm{fw}}}(0,0) \in C^{\mathrm{fw}}$,

C3) $\overline{x}_{v_0^{\text{bw}}} \in X_f$,

C4) for $i \in \{0, 1, ..., n-2\}$, if $\overline{\psi}_{e_i^{\text{bw}}}$ and $\overline{\psi}_{e_{i+1}^{\text{bw}}}$ are both purely continuous, then $\overline{\psi}_{e_{i+1}^{\text{bw}}}(0,0) \in C^{\text{bw}}$,

C5) $\overline{x}_{v_m^{\text{fw}}} = \overline{x}_{v_n^{\text{bw}}}$,

C6) if $\overline{\psi}_{e_{m-1}^{\text{fw}}}$ and $\overline{\psi}_{e_{n-1}^{\text{bw}}}$ are both purely continuous, then $\overline{\psi}_{e_{n-1}^{\text{bw}}}(T^{\text{bw}}, 0) \in C^{\text{fw}}$ where $(T^{\text{bw}}, 0) = \max \text{dom } \overline{\psi}_{e_{n-1}^{\text{bw}}}$.

If HyRRT-Connect is able to find a path p^{fw} in $\mathcal{T}^{\text{fw}}$ and a path p^{bw} in $\mathcal{T}^{\text{bw}}$ satisfying C1-C6, then a motion plan to $\mathcal{P}$ can be constructed by $\psi^{\text{fw}}|\psi^{\text{bw}'}$, where, for notation simplicity, $\psi^{\text{fw}} = (\phi^{\text{fw}}, \upsilon^{\text{fw}}) := \tilde{\psi}_{p^{\text{fw}}}$ denotes the solution pair associated with the path p^{fw} in (5.1) and is referred to as *forward partial motion plan*, $\psi^{\text{bw}} = (\phi^{\text{bw}}, \upsilon^{\text{bw}}) := \tilde{\psi}_{p^{\text{bw}}}$ denotes the solution pair associated with the path p^{bw} in (5.2) and is referred to as *backward partial motion plan*, and $\psi^{\text{bw}'}$ denotes the reversal of ψ^{bw}. The result $\psi^{\text{fw}}|\psi^{\text{bw}'}$ is guaranteed to satisfy each item in Problem 3.1 as follows:

1. By C1, it follows that $\psi^{\text{fw}}|\psi^{\text{bw}'}$ starts from X_0. Namely, item 1 in Problem 3.1 is satisfied.

2. Due to C2 (respectively, C4), by iterative applying Proposition 3.13 to each pair of $\overline{\psi}_{e_i^{\text{fw}}}$ and $\overline{\psi}_{e_{i+1}^{\text{fw}}}$ (respectively, $\overline{\psi}_{e_i^{\text{bw}}}$ and $\overline{\psi}_{e_{i+1}^{\text{bw}}}$) where $i \in \{0, 1, ..., m-2\}$ (respectively, $i \in \{0, 1, ..., n-2\}$), it follows that ψ^{fw} (respectively, ψ^{bw}) is a solution pair to $\mathcal{H}^{\text{fw}}$ (respectively, $\mathcal{H}^{\text{bw}}$). Furthermore, given C5 and C6, Lemma 3.17 establishes that $\psi^{\text{fw}}|\psi^{\text{bw}'}$ is a solution pair to $\mathcal{H}^{\text{fw}}$.

3. C3 ensures that $\psi^{\mathrm{fw}}|\psi^{\mathrm{bw}'}$ ends within X_f. This confirms the satisfaction of item 3 in Problem 3.1.

4. For any edge $e \in p^{\mathrm{fw}} \cup p^{\mathrm{bw}}$, the trajectory $\overline{\psi}_e$ avoids intersecting the unsafe set as a result of the exclusion of solution pairs that intersect the unsafe set in the function call `new_state`. Therefore, item 4 in Problem 3.1 is satisfied.

Since each requirement in Problem 3.1 is satisfied, it is established that $\psi^{\mathrm{fw}}|\psi^{\mathrm{bw}'}$ is a motion plan to $\mathcal{P}$.

In practice, as guaranteeing C5 above is not possible in most hybrid dynamical systems, given $\delta > 0$ representing the tolerance associated with this condition, we implement C5 as

$$|\overline{x}_{\upsilon_m^{\mathrm{fw}}} - \overline{x}_{\upsilon_n^{\mathrm{bw}}}| \leq \delta \tag{5.3}$$

leading to a potential discontinuity during the flow.

5.3.2 Reconstruction Process

To smoothen and control the discontinuity associated with (5.3), we propose a reconstruction process. Given the hybrid input υ^{bw} of ψ^{bw} identified in S1, which is backward in hybrid time, the reconstruction process involves simulating a hybrid arc, denoted ϕ^{r}, such that it starts from the final state of ϕ^{fw}, flows when $\upsilon^{\mathrm{bw}'}$ flows, jumps when $\upsilon^{\mathrm{bw}'}$ jumps, and applies the input $(t, j) \mapsto \upsilon^{\mathrm{bw}'}(t, j)$ where $\upsilon^{\mathrm{bw}'}$ denotes the reversal of υ^{bw}; see item 2 in Definition 3.4 for the reversal of a hybrid input. We generate ϕ^{r} via the following hybrid dynamical system, denoted $\mathcal{H}_{\upsilon^{\mathrm{bw}'}}$, with state $x \in \mathbb{R}^n$

and dynamics:

$$\mathcal{H}_{\upsilon^{\mathrm{bw}'}}: \begin{cases} \dot{x} = f_{\upsilon^{\mathrm{bw}'}}(x, \upsilon^{\mathrm{bw}'}(t,j)) & (t,j) \in C_{\upsilon^{\mathrm{bw}'}} \\ x^+ = g_{\upsilon^{\mathrm{bw}'}}(x, \upsilon^{\mathrm{bw}'}(t,j)) & (t,j) \in D_{\upsilon^{\mathrm{bw}'}} \end{cases} \tag{5.4}$$

where

1. $D_{\upsilon^{\mathrm{bw}'}} := \{(t,j) \in \operatorname{dom} \upsilon^{\mathrm{bw}'} : (t, j+1) \in \operatorname{dom} \upsilon^{\mathrm{bw}'}\}$;

2. $C_{\upsilon^{\mathrm{bw}'}} := \overline{\operatorname{dom} \upsilon^{\mathrm{bw}'} \backslash D_{\upsilon^{\mathrm{bw}'}}}$;

3. $g_{\upsilon^{\mathrm{bw}'}}(x,u) := g(x,u)$ for all[3] $(x,u) \in \mathbb{R}^n \times \mathbb{R}^m$;

4. $f_{\upsilon^{\mathrm{bw}'}}(x,u) := f(x,u)$ for all $(x,u) \in \mathbb{R}^n \times \mathbb{R}^m$.

In addition to satisfying the hybrid dynamics in (5.4), we also require that the reconstruction result ϕ^{r} satisfies the following conditions:

R1) $\phi^{\mathrm{r}}(0,0) = \phi^{\mathrm{fw}}(T^{\mathrm{fw}}, J^{\mathrm{fw}})$, where ϕ^{fw} is the state trajectory of ψ^{fw} identified in S1 and $(T^{\mathrm{fw}}, J^{\mathrm{fw}}) = \max \operatorname{dom} \phi^{\mathrm{fw}}$;

R2) ϕ^{r} is a maximal solution to $\mathcal{H}_{\upsilon^{\mathrm{bw}'}}$ such that $\operatorname{dom} \phi^{\mathrm{r}} = \operatorname{dom} \upsilon^{\mathrm{bw}'}$.

Remark 5.2. *The definitions of $C_{\upsilon^{\mathrm{bw}'}}$ and $D_{\upsilon^{\mathrm{bw}'}}$ indicate that ϕ^{r} follows the flow or jump of $\upsilon^{\mathrm{bw}'}$. R1 ensures that the reconstructed motion plan begins at the final state of the forward partial motion plan, effectively eliminating any discontinuity. Given that R2 ensures that ϕ^{r} is maximal, it follows that*

$$dom\ \phi^{\mathrm{r}} = dom\ \upsilon^{\mathrm{bw}'} = dom\ \phi^{\mathrm{bw}'}. \tag{5.5}$$

[3]The flow map f and the jump map g in (2.1) are defined on the domain $\mathbb{R}^n \times \mathbb{R}^m$.

5.3.2.1 Convergence of ϕ^{r} to X_f

We first show the dependency between the difference $|\phi^{\mathrm{r}}(T^{\mathrm{r}}, J^{\mathrm{r}}) - \phi^{\mathrm{bw}}(0,0)|$ and the tolerance δ in (4.11) where $(T^{\mathrm{r}}, J^{\mathrm{r}}) = \max \mathrm{dom}\ \phi^{\mathrm{r}}$. The following lemma is a slight modification of [27, Lemma 2].

Lemma 5.3. *Given a hybrid dynamical system $\mathcal{H}$ satisfying Assumption 4.24, if there exists a purely continuous solution pair $\psi^{\mathrm{bw}} = (\phi^{\mathrm{bw}}, \upsilon^{\mathrm{bw}})$ to $\mathcal{H}^{\mathrm{bw}}$, then the reconstructed solution ϕ^{r}, which is a solution to $\mathcal{H}_{\upsilon^{\mathrm{bw}'}}$ satisfying R1 and R2 where $\upsilon^{\mathrm{bw}'}$ is the reversal of υ^{bw}, satisfies the following properties:*

P1) dom ϕ^{r} = dom $\upsilon^{\mathrm{bw}'}$; thus, ϕ^{r} is purely continuous;

P2) if $|\phi^{\mathrm{r}}(0,0) - \phi^{\mathrm{bw}'}(0,0)| \leq \delta$, then $|\phi^{\mathrm{r}}(T,0) - \phi^{\mathrm{bw}'}(T,0)| \leq \exp(K_x^f T)\delta$, where $(T,0) = \max$ dom $\phi^{\mathrm{r}} = \max$ dom $\phi^{\mathrm{bw}'}$, and $\max$ dom $\phi^{\mathrm{r}} = \max$ dom $\phi^{\mathrm{bw}'}$ is valid because dom ϕ^{r} = dom $\phi^{\mathrm{bw}'}$ in P1.

Proof. This proof is in Appendix C.1. □

Lemma 5.4. *Given a hybrid dynamical system $\mathcal{H}$ satisfying Assumption 4.27, if there exists a purely discrete solution pair $\psi^{\mathrm{bw}} = (\phi^{\mathrm{bw}}, \upsilon^{\mathrm{bw}})$ to $\mathcal{H}^{\mathrm{bw}}$, then the reconstructed solution ϕ^{r}, which is a solution to $\mathcal{H}_{\upsilon^{\mathrm{bw}'}}$ satisfying R1 and R2 where $\upsilon^{\mathrm{bw}'}$ is the reversal of υ^{bw}, satisfies the following properties:*

P1) dom ϕ^{r} = dom $\upsilon^{\mathrm{bw}'}$; thus, ϕ^{r} is purely discrete;

P2) if $|\phi^{\mathrm{r}}(0,0) - \phi^{\mathrm{bw}'}(0,0)| \leq \delta$, then $|\phi^{\mathrm{r}}(0,J) - \phi^{\mathrm{bw}'}(0,J)| \leq \exp(J \ln(K_x^g))\delta$, where $(0,J) = \max$ dom $\phi^{\mathrm{r}} = \max$ dom $\phi^{\mathrm{bw}'}$, and $\max$ dom $\phi^{\mathrm{r}} = \max$ dom $\phi^{\mathrm{bw}'}$ is valid

because dom $\phi^{\mathrm{r}} = dom\ \phi^{\mathrm{bw}'}$ *in P1.*

Proof. This proof is in Appendix C.2. □

Next, we show that the final state of the reconstructed motion plan ϕ^{r} converges to $\phi^{\mathrm{bw}}(0,0) \in X_f$ as the tolerance δ in (5.3) approaches zero.

Theorem 5.5. *Suppose Assumptions 4.24 and 4.27 are satisfied, and there exist a solution pair* $\psi^{\mathrm{fw}} = (\phi^{\mathrm{fw}}, \upsilon^{\mathrm{fw}})$ *to* $\mathcal{H}^{\mathrm{fw}}$ *and a solution pair* $\psi^{\mathrm{bw}} = (\phi^{\mathrm{bw}}, \upsilon^{\mathrm{bw}})$ *to* $\mathcal{H}^{\mathrm{bw}}$ *identified in S1. For each* $\epsilon > 0$*, there exists a tolerance* $\delta > 0$ *in (5.3) such that* $|\phi^{\mathrm{fw}}(T^{\mathrm{fw}}, J^{\mathrm{fw}}) - \phi^{\mathrm{bw}}(T^{\mathrm{bw}}, J^{\mathrm{bw}})| \leq \delta$ *leads to* $|\phi^{\mathrm{r}}(T^{\mathrm{r}}, J^{\mathrm{r}}) - \phi^{\mathrm{bw}}(0,0)| \leq \epsilon$ *where* $(T^{\mathrm{fw}}, J^{\mathrm{fw}}) = \max dom\ \phi^{\mathrm{fw}}$*,* $(T^{\mathrm{bw}}, J^{\mathrm{bw}}) = \max dom\ \phi^{\mathrm{bw}}$*,* ϕ^{r} *is a solution to* $\mathcal{H}_{\upsilon^{\mathrm{bw}'}}$ *following R1 and R2, and* $(T^{\mathrm{r}}, J^{\mathrm{r}}) = \max dom\ \phi^{\mathrm{r}}$*.*

Proof. Given that $\psi^{\mathrm{bw}} = (\phi^{\mathrm{bw}}, \upsilon^{\mathrm{bw}})$ is a solution pair to $\mathcal{H}^{\mathrm{bw}}$, Proposition 3.9 guarantees that $\psi^{\mathrm{bw}'} = (\phi^{\mathrm{bw}'}, \upsilon^{\mathrm{bw}'})$ is a solution pair to $\mathcal{H}^{\mathrm{fw}}$. By (5.5), it follows that dom $\phi^{\mathrm{r}} =$ dom $\phi^{\mathrm{bw}'}$. Since $|\phi^{\mathrm{fw}}(T^{\mathrm{fw}}, J^{\mathrm{fw}}) - \phi^{\mathrm{bw}}(T^{\mathrm{bw}}, J^{\mathrm{bw}})| = |\phi^{\mathrm{r}}(0,0) - \phi^{\mathrm{bw}'}(0,0)| \leq \delta$, by iteratively applying Lemma 5.3 and Lemma 5.4 throughout dom $\phi^{\mathrm{r}} =$ dom $\phi^{\mathrm{bw}'}$, it follows that

$$|\phi^{\mathrm{r}}(T^{\mathrm{r}}, J^{\mathrm{r}}) - \phi^{\mathrm{bw}}(0,0)| = |\phi^{\mathrm{r}}(T^{\mathrm{r}}, J^{\mathrm{r}}) - \phi^{\mathrm{bw}'}(T^{\mathrm{r}}, J^{\mathrm{r}})| \leq \exp(K_x^f T^{\mathrm{r}} + J^{\mathrm{r}} \ln(K_x^f))\delta,$$

which establishes the existence of $\delta > 0$. In particular, δ can be taken to be equal to $\frac{\epsilon}{\exp(K_x^f T^{\mathrm{r}} + J^{\mathrm{r}} \ln(K_x^f))}$. □

Furthermore, if $\phi^{\mathrm{bw}}(0,0)$ is not on the boundary of X_f, the following result shows there is a tolerance ensuring that ϕ^{r} concludes within X_f.

Corollary 5.6. *Suppose Assumptions 4.24 and 4.27 are satisfied, and there exist a solution pair* $\psi^{\mathrm{fw}} = (\phi^{\mathrm{fw}}, \upsilon^{\mathrm{fw}})$ *to* $\mathcal{H}^{\mathrm{fw}}$*, and a solution pair* $\psi^{\mathrm{bw}} = (\phi^{\mathrm{bw}}, \upsilon^{\mathrm{bw}})$ *to* $\mathcal{H}^{\mathrm{bw}}$ *identified in S1, and some* $\epsilon' > 0$ *such that* $\phi^{\mathrm{bw}}(0,0) + \epsilon'\mathbb{B} \subset X_f$*. Then, there exists a tolerance* $\delta > 0$ *in (5.3) such that* $|\phi^{\mathrm{fw}}(T^{\mathrm{fw}}, J^{\mathrm{fw}}) - \phi^{\mathrm{bw}}(T^{\mathrm{bw}}, J^{\mathrm{bw}})| \leq \delta$ *leads to* $\phi^{\mathrm{r}}(T^{\mathrm{r}}, J^{\mathrm{r}}) \in X_f$ *where* $(T^{\mathrm{fw}}, J^{\mathrm{fw}}) = \max dom\ \phi^{\mathrm{fw}}$*,* $(T^{\mathrm{bw}}, J^{\mathrm{bw}}) = \max dom\ \phi^{\mathrm{bw}}$*,* ϕ^{r} *is a solution to* $\mathcal{H}_{\upsilon^{\mathrm{bw}\prime}}$ *following R1 and R2, and* $(T^{\mathrm{r}}, J^{\mathrm{r}}) = \max dom\ \phi^{\mathrm{r}}$*.*

Proof. By selecting $\epsilon = \epsilon'$, Theorem 5.5 ensures the existence of some $\delta > 0$ such that $|\phi^{\mathrm{r}}(T^{\mathrm{r}}, J^{\mathrm{r}}) - \phi^{\mathrm{bw}}(0,0)| \leq \epsilon'$. By $\phi^{\mathrm{bw}}(0,0) + \epsilon'\mathbb{B} \subset X_f$, it is established that $\phi^{\mathrm{r}}(T^{\mathrm{r}}, J^{\mathrm{r}}) \in X_f$. □

Then, by replacing ϕ^{bw} with ϕ^{r} and concatenating the reconstructed pair $\psi^{\mathrm{r}} := (\phi^{\mathrm{r}}, \upsilon^{\mathrm{bw}\prime})$ to $\psi^{\mathrm{fw}} = (\phi^{\mathrm{fw}}, \upsilon^{\mathrm{fw}})$, HyRRT-Connect generates the motion plan $\psi^{\mathrm{fw}}|\psi^{\mathrm{r}}$, where the discontinuity associated with (5.3) is removed. Note that the tolerance δ in (5.3) is adjustable. Setting δ to a smaller value brings the endpoint of ϕ^{r} closer to X_f, However, it also reduces the possibility of finding a motion plan, thereby increasing the time expected to find forward and backward partial motion plans.

Remark 5.7. *Connecting two points via flow typically involves solving a two point boundary value problem constrained by the flow set. Solving such problems is difficult. This is the reason why we do not consider to actively connect two paths in* $\mathcal{T}^{\mathrm{fw}}$ *and* $\mathcal{T}^{\mathrm{bw}}$ *via flow.*

5.3.3 Connecting Forward and Backward Search Trees via Jump

In S2, HyRRT-Connect checks the existence of p^{fw} in (5.1) and p^{bw} in (5.2) which, in addition to meeting C1-C4 in Section 5.3.1, results in a solution to the following constrained equation, denoted u^*, provided one exists[4]:

$$\overline{x}_{v_n^{\text{bw}}} = g(\overline{x}_{v_m^{\text{fw}}}, u^*), \quad (\overline{x}_{v_m^{\text{fw}}}, u^*) \in D^{\text{fw}}. \tag{5.6}$$

The constrained equation above can be solved analytically for certain hybrid dynamical systems such as the one in Example 3.2 and numerically in general using numerical techniques like root-finding methods [8] or optimization methods [5]. A solution to (5.6) implies that $\overline{x}_{v_m^{\text{fw}}}$ and $\overline{x}_{v_n^{\text{bw}}}$ can be connected by applying u^* at a jump from $\overline{x}_{v_m^{\text{fw}}}$ to $\overline{x}_{v_n^{\text{bw}}}$. Hence, a motion plan is constructed by concatenating ψ^{fw}, a single jump from $\overline{x}_{v_m^{\text{fw}}}$ to $\overline{x}_{v_n^{\text{bw}}}$, and $\psi^{\text{bw}'}$.

Remark 5.8. *This approach enables the construction of a motion plan prior to the detection of any overlaps between $\mathcal{T}^{\text{fw}}$ and $\mathcal{T}^{\text{bw}}$, potentially leading to improved computational efficiency, as illustrated in the forthcoming Section 5.4. Furthermore, as this connection is established through a jump, it prevents the discontinuity during the flow introduced in (5.3).*

[4]It is indeed possible that all the motion plans are purely continuous. In this case, no solution to (5.6) would be found since no jumps exist in every motion plan.

5.4 Software Tool and Simulation Results

Algorithm 4 leads to a software tool[5] to solve Problem 3.1.This software only requires the inputs listed in Algorithm 4. Next, we illustrate the HyRRT-Connect algorithm and this tool in Example 3.2 and Example 3.3.

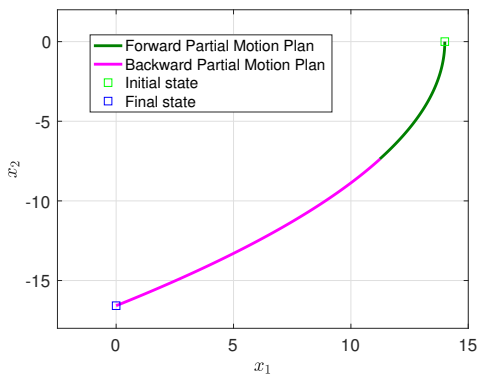
Example 5.9 (Actuated bouncing ball system in Example 3.2, revisited)**.** *We initially showcase the simulation results of the HyRRT-Connect algorithm without the functionality of connecting via jumps discussed in Section 5.3.3. We consider the case where HyRRT-Connect precisely connects the forward and backward partial motion plans. This is demonstrated by deliberately setting the initial state set as* $X_0 = \{(14, 0)\}$ *and the final state set as* $X_f = \{(0, -16.58)\}$*. In this case, no tolerance is applied, and thus, no reconstruction process is employed. Instead, strict equality in C5 in Section 5.3.1 is used to identify the motion plan. The motion plan detected under these settings is depicted in Figure 5.1(a), where the forward and backward partial motion plans identified in S1 are depicted by the green and magenta lines, respectively. However, for most scenarios, such as* $X_0 = \{(14, 0)\}$ *and* $X_f = \{(10, 0)\}$ *in Example 3.2, if we require strict equality without allowing any tolerance, then HyRRT-Connect fails to return a motion plans in almost all the runs. This demonstrates the necessity of allowing a certain degree of tolerance in HyRRT-Connect. The simulation results, allowing a tolerance of* $\delta = 0.2$*, are shown in Figure 5.1(b). A discontinuity during the flow between the forward and backward partial motion plans is observed, as depicted in the red circle in Figure 5.1(b).*

[5]Code at https://github.com/HybridSystemsLab/HyRRTConnect.git.

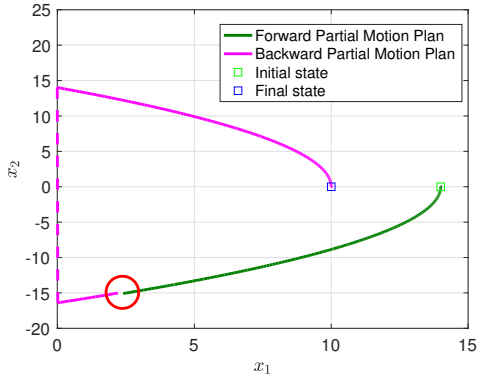
This discontinuity is addressed through the reconstruction process, as is shown in Figure 5.1(c). A deviation between the endpoint of the reconstructed motion plan and the final state set is also observed in Figure 5.1(c), which, according to Theorem 5.5, is bounded.

Next, we proceed to perform simulation results of HyRRT-Connect showcasing its full functionalities, including the ability to connect partial motion plans via jumps. Figure 6.3(a) shows this situation. This feature enables HyRRT-Connect to avoid discontinuities during the flow, as it computes exact solutions at jumps to connect forward and backward partial motion plans. Furthermore, we compare the computational performance of the proposed HyRRT-Connect algorithm, its variant Bi-HyRRT (where the function to connect partial motion plans via jumps is disabled), and HyRRT given in [56, 61]. Conducted on a $3.5GHz$ *Intel Core i7 processor using MATLAB, each algorithm is run* 20 *times on the same problem. HyRRT-Connect on average creates* 78.8 *vertices in* 0.27 *seconds, Bi-HyRRT* 186.5 *vertices in* 0.76 *seconds, and HyRRT* 457.4 *vertices in* 3.93 *seconds. Compared to HyRRT, both HyRRT-Connect and Bi-HyRRT show considerable improvements in computational efficiency. Notably, HyRRT-Connect, with its jump-connecting capability, achieves a* 64.5% *reduction in computation time and* 57.7% *fewer vertices than Bi-HyRRT, demonstrating the benefits of jump connections.*

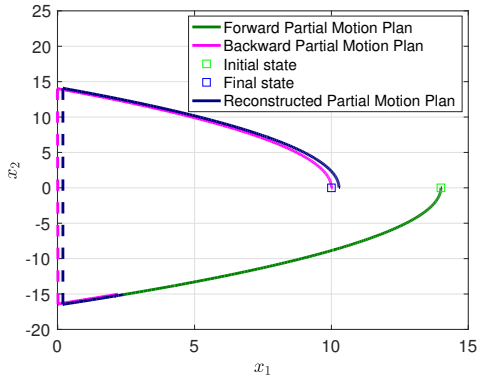
Example 5.10 (Walking robot system in Example 3.3, revisited)**.** *The simulation results demonstrate that HyRRT-Connect successfully finds a motion plan for the high-dimensional walking robot system with a tolerance* δ *of* 0.3*. The forward search tree* $\mathcal{T}^{\text{fw}}$*, with its partial motion plan shown in green, is displayed in Figure 5.2. Similarly,*



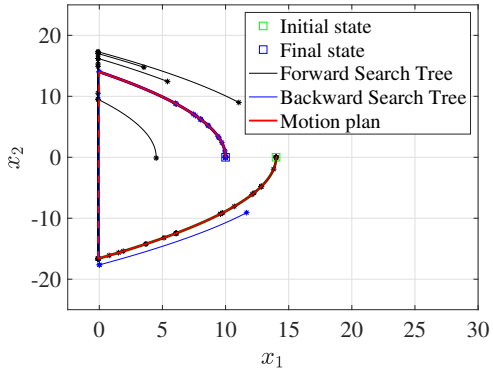
(a) Precise connection during the flow is achieved.



(b) A discontinuity during the flow in red circle.



(c) The backward partial motion plan is reconstructed.



(d) HyRRT-Connect

Figure 5.1: Motion plans for the actuated bouncing ball example.

the backward search tree $\mathcal{T}^{\text{bw}}$*, with its partial motion plan in magenta, is shown in Figure 5.3. These simulations were performed in MATLAB on a 3.5 GHz Intel Core i7 processor. Running HyRRT-Connect and HyRRT* 20 *times each for the same problem, HyRRT-Connect generates* 470.2 *vertices and takes* 19.8 *seconds, while HyRRT generates* 2357.1 *vertices and takes* 71.5 *seconds. This indicates a significant* 72.3% *improvement in computation time and* 80.1% *in vertex creation for HyRRT-Connect compared to HyRRT, highlighting the efficiency of bidirectional exploration.*

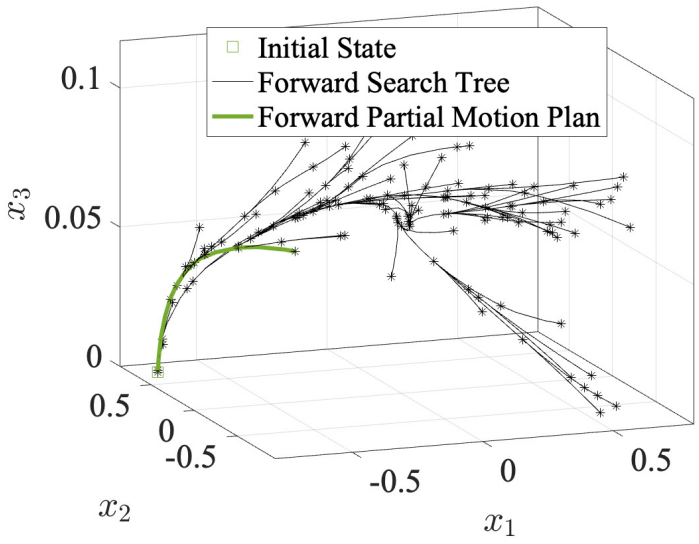


Figure 5.2: Forward search tree and the forward partial motion plan.

5.5 Discussion on Parallel Implementation

In this section, we discuss the computation performance on the parallel implementation of the HyRRT-Connect algorithm. We have ascertained that when im-

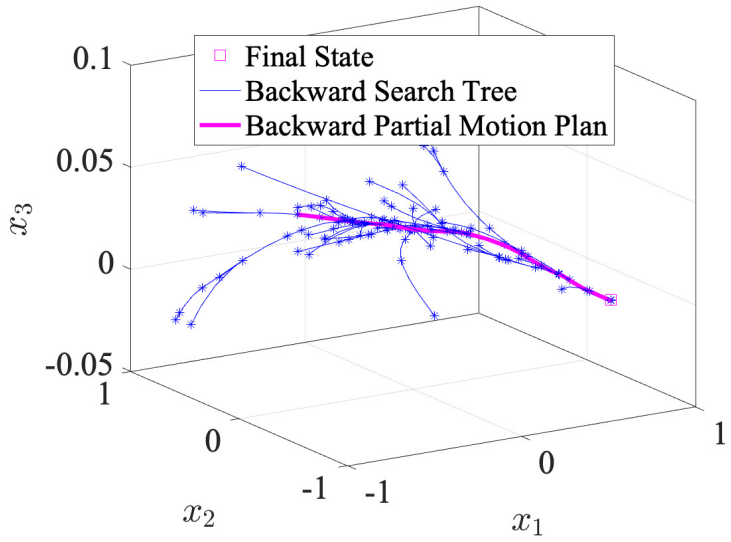


Figure 5.3: Backward search tree and the backward partial motion plan.

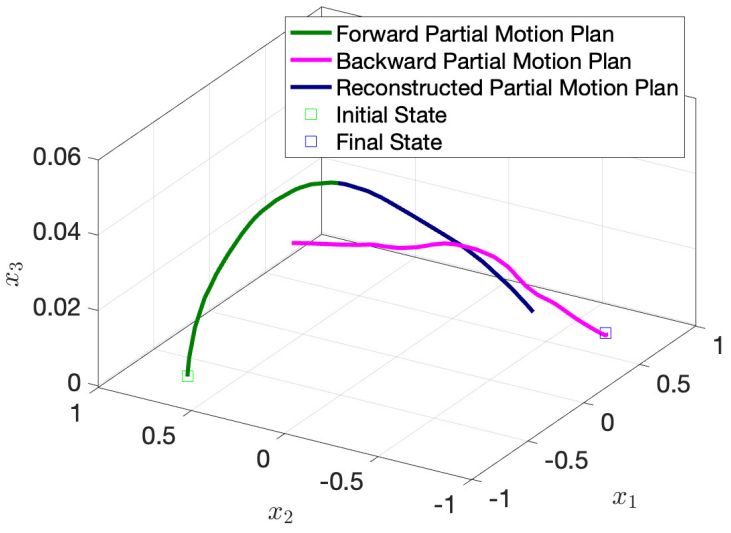


Figure 5.4: Selected states of the forward and backward partial motion plan generated by HyRRT-Connect for the walking robot system.

plementing HyRRT-Connect using MATLAB's internal parallel computation toolbox, `parpool`, there is no improvement in computational performance compared to the interleaved implementation.This observation is based on the results obtained from solving the two example problems discussed. On average, the parallel implementation of HyRRT-Connect takes approximately 2.47 seconds to compute a motion plan for the actuated bouncing ball system. This time cost is significantly longer than the time required by the interleaved implementation, which averages around 0.27 second. A similar conclusion is observed in the case of the walking robot system. The parallel computation approach requires 167.8 seconds to complete, in contrast to the interleaved implementation, which completes the task in significantly less time, taking only 19.8 seconds. It is important to note that the time required for halting and restarting parallel computation is contingent upon factors like the specific parallel computation software toolbox used, the hardware platform, and other implementation details. Consequently, the conclusions regarding performance may differ with varying implementations.

5.6 Discussion on Probabilistic Completeness

The HyRRT-Connect algorithm renders probabilistic completeness, implying that as the number of samples approaches infinity, the probability of failure to find a motion plan converges to zero if one exists. This property extends from the probabilistic completeness of the HyRRT algorithm shown in [56], which propagates in a forward direction. By truncating an existing motion plan into two segments, HyRRT maintains

probabilistic completeness in finding each segment, thereby ensuring the probabilistic completeness of HyRRT-Connect.

Chapter 6

Sampling-based Optimal Motion Planning for Hybrid Dynamical Systems

6.1 Problem Statement

The formulation of the feasible motion planning problem for hybrid dynamical systems can be found in Problem 3.1 and is denoted as $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$, where the initial state set is denoted as $X_0 \subset \mathbb{R}^n$, the final state set is denoted as $X_f \subset \mathbb{R}^n$, and the unsafe set is denoted as $X_u \subset \mathbb{R}^n \times \mathbb{R}^m$. Let $\hat{\mathcal{S}}_{\mathcal{H}}$ denote the set of all solution pairs to $\mathcal{H}$. Let $\hat{\mathcal{S}}^{\phi}_{\mathcal{H}}$ denote the set of state trajectories of all the solution pairs in $\hat{\mathcal{S}}_{\mathcal{H}}$. The optimal motion planning problem for hybrid dynamical systems consists of finding a feasible motion plan with minimum cost [24, Problem 3].

Problem 6.1 (Optimal motion planning problem for hybrid dynamical systems)**.** *Given a motion planning problem* $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$ *and a cost functional* $c : \hat{\mathcal{S}}^{\phi}_{\mathcal{H}} \rightarrow$

$\mathbb{R}_{\geq 0}$, *find a feasible motion plan* (ϕ^*, u^*) *to* $\mathcal{P}$ *such that* $(\phi^*, u^*) = \arg\min_{(\phi,u)\in\hat{\mathcal{S}}_{\mathcal{H}}} c(\phi)$. *Given sets* X_0, X_f, *and* X_u, *a hybrid dynamical system* $\mathcal{H}$ *with data* (C, f, D, g), *and a cost functional* c, *an optimal motion planning problem* $\mathcal{P}^*$ *is formulated as* $\mathcal{P}^* = (X_0, X_f, X_u, (C, f, D, g), c)$.

Example 6.2 (Actuated bouncing ball system in Example 3.2, revisited)**.** *Consider a ball bouncing on a fixed horizontal surface in Example 3.2. Given the initial state set* $X_0 = \{(15, 0)\}$, *the final state set* $X_f = \{(10, 0)\}$, *and the unsafe set* $X_u = \{(x, u) \in \mathbb{R}^2 \times \mathbb{R} : x_1 \in [20, \infty), u \in [5, \infty)\}$, *an instance of the optimal motion planning problem for the actuated bouncing ball system is to find a motion plan that has minimal hybrid time. To capture the hybrid time domain information, an auxiliary state* $\tau \in \mathbb{R}_{\geq 0}$ *representing the normal time and an auxiliary state* $k \in \mathbb{N}$ *representing the jump numbers are imported. An auxiliary hybrid dynamical system* $\overline{\mathcal{H}} := (\overline{C}, \overline{f}, \overline{D}, \overline{g})$ *with state* $\overline{x} := (x, \tau, k) \in \mathbb{R}^2 \times \mathbb{R}_{\geq 0} \times \mathbb{N}$ *is constructed as follows*

$$\overline{C} := \left\{(\overline{x}, u) \in \mathbb{R}^2_{\geq 0} \times \mathbb{R}_{\geq 0} \times \mathbb{N} \times \mathbb{R} : (x, u) \in C\right\} \tag{6.1}$$

$$\overline{f}(\overline{x}, u) := \begin{bmatrix} f(x, u) \\ 1 \\ 0 \end{bmatrix} \quad \forall (\overline{x}, u) \in \overline{C} \tag{6.2}$$

$$\overline{D} := \left\{(\overline{x}, u) \in \mathbb{R}^2 \times \mathbb{R}_{\geq 0} \times \mathbb{N} \times \mathbb{R}_{\geq 0} : (x, u) \in D\right\} \tag{6.3}$$

$$\overline{g}(\overline{x}, u) := \begin{bmatrix} g(x, u) \\ \tau \\ k + 1 \end{bmatrix} \quad \forall (\overline{x}, u) \in \overline{D} \tag{6.4}$$

with the sets X_0, X_f, and X_u extended to

$$\overline{X}_0 := X_0 \times (0,0) \tag{6.5}$$

$$\overline{X}_f := X_f \times \mathbb{R}_{\geq 0} \times \mathbb{N} \tag{6.6}$$

$$\overline{X}_u := X_u \times \mathbb{R}_{\geq 0} \times \mathbb{N}. \tag{6.7}$$

Then the cost functional c can be defined as

$$c(\overline{\phi}) = c(\phi, \tau, k) := \tau(T, J) + k(T, J) \tag{6.8}$$

where $\overline{\phi} = (\phi, \tau, k)$ denotes a state trajectory of the solution pair to $\overline{\mathcal{H}}$, and $(T, J) = \max dom\, \overline{\phi}$. Then the example optimal motion planning problem for the bouncing ball is defined as $\mathcal{P}^* = (\overline{X}_0, \overline{X}_f, \overline{X}_u, (\overline{C}, \overline{f}, \overline{D}, \overline{g}), c)$.

Example 6.3 (Collision-resilient tensegrity multicopter system [66, 67])**.** *Consider a planar collision-resilient tensegrity multicopter that is resilient to collisions with a wall. The state of the multicopter involves the position vector $p := (p_x, p_y) \in \mathbb{R}^2$, the velocity vector $v := (v_x, v_y) \in \mathbb{R}^2$, and the acceleration vector $a := (a_x, a_y) \in \mathbb{R}^2$, where, respectively, p_x and p_y denote the position, v_x and v_y denote the velocity, and a_x and a_y denote the acceleration along the x-axis and y-axis. The state of the system is $x := (p, v, a) \in \mathbb{R}^6$ and its input is $u := (u_x, u_y) \in \mathbb{R}^2$ which represents the effect of the torque. The environment is assumed to be known. Define the walls as the region $\mathcal{W} \subset \mathbb{R}^2$, which is a closed set represented by the blue rectangles in Figure 6.1. Flow is allowed when the multicopter is in*

$$C := (\mathbb{R}^2 \backslash \mathcal{W}) \times \mathbb{R}^4 \times \mathbb{R}^2, \tag{6.9}$$

which defines the flow set. The dynamics of the multicopter when no collision occurs is

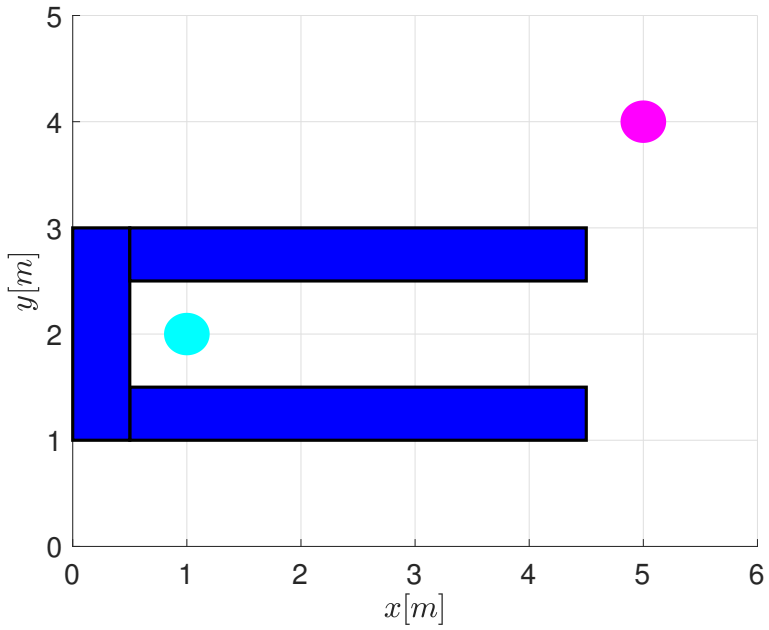


Figure 6.1: The environment around the collision-resilient tensegrity multicopter. In the figure above, the blue rectangles denote the obstacles that may cause the collision with the multicopter. The cyan circle denotes the initial state and the magnet circle denotes the final state set.

captured as

$$\dot{x} = \begin{bmatrix} v \\ a \\ u \end{bmatrix} =: f(x,u), (x,u) \in C. \tag{6.10}$$

At collisions, the position is assumed to remain constant. To model the change of v, denote the velocity component of $v = (v_x, v_y)$ that is normal to the wall as v_N and the velocity component that is tangential to the wall as v_T. Then, the velocity component v_N after the jump is modeled as

$$v_N{}^+ = -\lambda v_N =: \tilde{g}_N(v) \tag{6.11}$$

where $\lambda \in (0,1)$ *is the coefficient of restitution. The velocity component* v_T *after the jump is modeled as*

$$v_T{}^+ = v_T + \kappa(-\lambda - 1)\arctan\frac{v_T}{v_N}v_N =: \tilde{g}_T(v), \tag{6.12}$$

where $\kappa \in \mathbb{R}$ *is a constant; see [67]. Denoting the projection of the updated vector* (v_N^+, v_T^+) *onto the* x*-axis as* $\Pi_x(v_N^+, v_T^+)$ *and the projection of the updated vector* (v_N^+, v_T^+) *onto the* y*-axis as* $\Pi_y(v_N^+, v_T^+)$*, we have* $v^+ = (\Pi_x(\tilde{g}_N(v), \tilde{g}_T(v)), \Pi_y(\tilde{g}_N(v), \tilde{g}_T(v))) =: \tilde{g}(v)$*. We assume that* $a^+ = 0$*, which, through a post-impact hovering maneuver, can be mitigated in the control layer. The discrete dynamics capturing the collision process is modeled as*

$$x^+ = \begin{bmatrix} p \\ \tilde{g}(v) \\ 0 \end{bmatrix} =: g(x,u)\ (x,u) \in D. \tag{6.13}$$

Jumps are allowed when the multicopter is on the wall surface with positive velocity towards the wall. Hence, the jump set is

$$D := \{((p,v,a),u) \in \mathbb{R}^6 \times \mathbb{R}^2 : p \in \partial\mathcal{W}, v_N \leq 0\}. \tag{6.14}$$

The hybrid model of the collision-resilient tensegrity multicopter system is given by (2.1) where the flow map f *is given in (6.10), the flow set* C *is given in (6.9), the jump map* g *is given in (6.13), and the jump set* D *is given in (6.14). Given the initial state set as* $X_0 = \{(1,2,0,0,0,0)\}$*, the final state set as* $X_f = \{(5,4)\} \times \mathbb{R}^4$*, and the unsafe set as* $X_u = \{(x,u) \in \mathbb{R}^6 \times \mathbb{R}^2 : \sqrt{(p_x-5)^2 + (p_y-3)^2}| \leq 0.3\}$ *which represents the green ball in Figure 6.4 that is forbidden to fly into or collide with, an instance of the optimal*

motion planning problem for the collision-resilient tensegrity multicopter system is to find the motion plan with minimal hybrid time. To capture the hybrid time domain information, an auxiliary state $\tau \in \mathbb{R}_{\geq 0}$ *representing the ordinary time and an auxiliary state* $k \in \mathbb{N}$ *representing the number of jumps associated to collisions are included. The resulting hybrid dynamical system* $\overline{\mathcal{H}} := (\overline{C}, \overline{f}, \overline{D}, \overline{g})$ *with state* $\overline{x} := (x, \tau, k) \in \mathbb{R}^2 \times \mathbb{R}_{\geq 0} \times \mathbb{N}$, *input* $u \in \mathbb{R}^2$, *and data* $\overline{C} := \big\{(\overline{x}, u) \in \mathbb{R}^2 \times \mathbb{R}_{\geq 0} \times \mathbb{N} \times \mathbb{R} : (x, u) \in C\big\}$; $\overline{f}(\overline{x}, u) := (f(x, u), 1, 0)$ *for each* $(\overline{x}, u) \in \overline{C}$; $\overline{D} := \big\{(\overline{x}, u) \in \mathbb{R}^2 \times \mathbb{R}_{\geq 0} \times \mathbb{N} \times \mathbb{R} : (x, u) \in D\big\}$; $\overline{g}(\overline{x}, u) := (g(x, u), \tau, k + 1)$ *for each* $(\overline{x}, u) \in \overline{D}$ *with the* X_0, X_f, *and* X_u *extended as* $\overline{X}_0 := X_0 \times \{0\} \times \{0\}$, $\overline{X}_f := X_f \times \mathbb{R}_{\geq 0} \times \mathbb{N}$, $\overline{X}_u := X_u \times \mathbb{R}_{\geq 0} \times \mathbb{N}$. *Then, with* $\overline{\phi} = (\phi, \tau, k)$ *being a state trajectory of the solution pair to* $\overline{\mathcal{H}}$, *the cost functional* c *can be defined as*

$$c(\overline{\phi}) := \tau(T, J) + k(T, J), \tag{6.15}$$

where $(T, J) = \max dom\, \overline{\phi}$. *The resulting optimal motion planning problem is defined as* $\mathcal{P}^* = (\overline{X}_0, \overline{X}_f, \overline{X}_u, (\overline{C}, \overline{f}, \overline{D}, \overline{g}), c)$.

In the forthcoming Example 6.10 and Example 6.11, we employ HySST to solve these motion planning problems.

6.2 Overview

HySST searches for the optimal motion plan by incrementally constructing a search tree. Each vertex $v \in V$ in the search tree $\mathcal{T} = (V, E)$ is associated with a state value of $\mathcal{H}$, denoted $\overline{x}_v$, and a cost value that, via addition, compounds the cost

from the root vertex up to the vertex v, denoted $\overline{c}_v$. Each edge $e \in E$ in the search tree $\mathcal{T} = (V, E)$ is associated with a solution pair to $\mathcal{H}$, denoted $\overline{\psi}_e$. HySST requires a library of possible inputs. The input library ($\mathcal{U}_C$,$\mathcal{U}_D$) includes the input signals that can be applied during flows (collected in $\mathcal{U}_C$) and the input values that can be applied at jumps (collected in $\mathcal{U}_D$).

HySST selects the vertex associated with the lowest cost within the vicinity of a randomly selected state. This vicinity is referred to as *random state neighborhood* and defined by a ball of radius $\delta_{BN} \in \mathbb{R}_{>0}$. Then, HySST employs a pruning process to decrease the number of vertices in the search tree. It is not required to keep all the vertices in the search tree because some of the vertices may be close to a vertex with much lower cost. This observation allows for a pruning operation to ignore some vertices generated during the search. This pruning operation is implemented by maintaining a *witness state set*, denoted S, such that all the vertices within the vicinity of the witnesses are deleted except the ones with lowest cost. This vicinity is referred to as *closest witness neighborhood* and defined by a ball of radius $\delta_s \in \mathbb{R}_{>0}$, For every witness s kept in S, a single vertex in the tree represents that witness. Such a vertex is stored in $s.rep$ for each witness $s \in S$. Note that a vertex, say, v_a, may be associated with a higher cost than other vertices within the same witness's neighborhood, but has a child vertex, say, v_b, associated with the lowest cost compared with other vertices in the same witness's neighborhood. In this case, v_a should not be removed from the search tree because, if it is removed, then all of its child vertices, including v_b with the lowest cost, are consequently removed. However, even v_a is not removed, it will not be selected, and,

therefore, will be kept in a separate set called *inactive vertex set*, denoted $V_{inactive}$. On the other hand, the vertices that are not pruned are stored in a set called the *active vertex set*, denoted V_{active}.

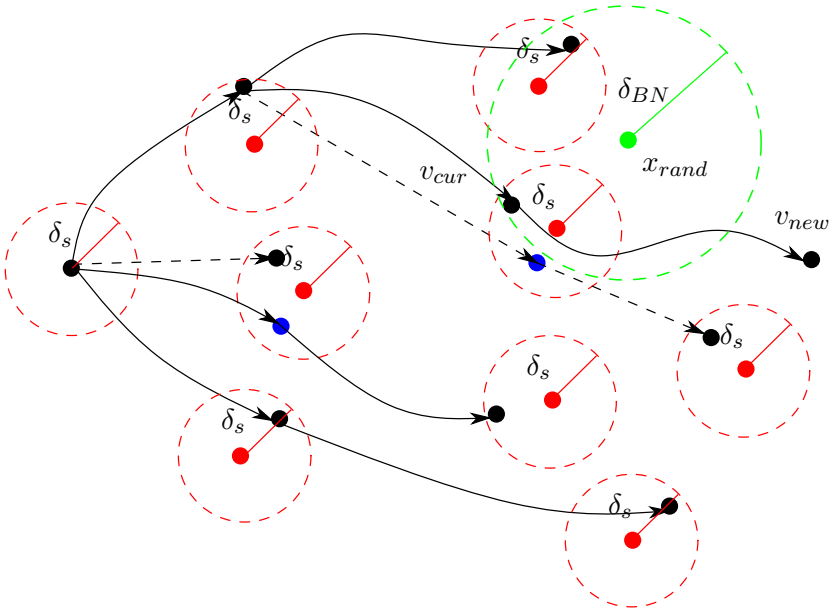


Figure 6.2: The search tree witnessed by a witness set. In this figure, the black dots and lines denote the active vertices and the edges in the search tree. The blue dots denote the inactive vertices. The red dots denote the witness states. The green dot denotes the randomly selected state x_{rand}. Any single witness has an active vertex as its representative.

Next, we introduce the main steps executed by HySST. Given the optimal motion planning problem $\mathcal{P}^* = (X_0, X_f, X_u, (C, f, D, g), c)$ and the input library $(\mathcal{U}_C, \mathcal{U}_D)$, HySST performs the following steps:

Step 1: Initialize a search tree $\mathcal{T} = (V, E)$ by sampling a finite number of points from X_0. For each sampling point x_0, add a vertex v_0 and assign $\overline{x}_{v_0} \leftarrow x_0$. Initialize E by $E \leftarrow \emptyset$. Initialize the witness state set $S \subset \mathbb{R}^n$ by $S \leftarrow \emptyset$. For each $v \in V$

such that $|\overline{x}_v - \overline{x}_{v'}| > \delta_s$ for all $v' \in V \backslash v$, add the witness state $s = \overline{x}_v$ to S and set the representative of s as $s.rep \leftarrow v$. Initialize the active vertices set V_{active} by $V_{active} \leftarrow \{s.rep \in V : s \in S\}$. Initialize the inactive vertices set $V_{inactive}$ by $V_{inactive} \leftarrow \emptyset$.

Step 2: Randomly select flow regime or jump regime for the evolution of $\mathcal{H}$.

Step 3: Randomly select a point x_{rand} from C' (D') if the flow (respectively, jump) regime is selected in **Step 2**.

Step 4: Find all the vertices in V_{active} associated with the state values that are within δ_{BN} to x_{rand} and collect them in the set V_{BN}. Then, find vertex in V_{BN} that has minimal cost, denoted v_{cur}. If no vertex is collected in V_{BN}, then find vertex in the search tree that has minimal distance to x_{rand} and assign it to v_{cur}.

Step 5: Randomly select an input signal (respectively, value) from $\mathcal{U}_C$ (respectively, $\mathcal{U}_D$) if $\overline{x}_{v_{cur}} \in C' \backslash D'$ (respectively, $\overline{x}_{v_{cur}} \in D' \backslash C'$). Then, compute a solution pair denoted $\psi_{new} = (\phi_{new}, u_{new})$ starting from $\overline{x}_{v_{cur}}$ with the selected input applied via flow (respectively, jump). If $\overline{x}_{v_{cur}} \in D' \cap C'$, a random process is employed to decide whether to proceed the computation with flow or jump. Denote the final state of ϕ_{new} as x_{new}. Compute the cost at x_{new}, denoted c_{new}, by $c_{new} \leftarrow \overline{c}_{v_{cur}} + c(\phi_{new})$. If ψ_{new} intersects with X_u, then go to **Step 2**.

Step 6: Find the witness in S that is closest to x_{new}, denoted s_{near}, and proceed as

follows:

- If x_{new} is not in the closest witness neighborhood of s_{near}, namely, $|s_{near} - x_{new}| > \delta_s$, then add a vertex v_{new} associated with x_{new} to V_{active} and an edge (v_{cur}, v_{new}) associated with ψ_{new} to E. Add a new witness $s_{new} \leftarrow x_{new}$ to S and set its representative as v_{new}. Then, go to **Step 2**.
- If $|s_{near} - x_{new}| \leq \delta_s$,
  - if $\overline{c}_{s_{near}.rep} > c_{new}$, add a vertex v_{new} associated with x_{new} to V_{active} and an edge (v_{cur}, v_{new}) associated with ψ_{new} to E. Then, update the representative of s_{near} with v_{new} and prune the vertex, say, v_{pre_near} which is previously witnessed by s_{near}. If v_{pre_near} is an active vertex, then add v_{pre_near} to $V_{inactive}$. Otherwise, remove v_{pre_near} and all its child vertices from the search tree. Then, go to **Step 2**.
  - if $\overline{c}_{s_{near}.rep} \leq c_{new}$, go to **Step 2** directly.

The steps above are illustrated in Figure 6.2. First, the algorithm initializes the search tree and the witness state set from the data in $\mathcal{P}^*$ as **Steps 1 - 2** show. Then, HySST randomly selects between flow regime and jump regime, followed by choosing a random state, which is denoted x_{rand} and represented by a green dot in Figure 6.2 following **Steps 3 - 4**. The green circle in Figure 6.2 denotes the random state neighborhood of x_{rand} defined by δ_{BN} in **Step 5**. Then, among all the active vertices represented by black dots in Figure 6.2, HySST finds one within the ball of radius δ_{BN} associated to lowest cost, denoted v_{cur} as is described in **Step 5**. The inactive vertices represented

by blue dots are ignored at this step. Next, HySST propagates forward by applying a randomly selected input to generate v_{new}, shown in Figure 6.2, as is described in **Step 6**. Note that in Figure 6.2, v_{new} is not in the closest witness neighborhood of any existing witness, represented by the red circles. In this case, v_{new} is added to V_{active}, no vertices need to be pruned, and, therefore, $V_{inactive}$ is not updated. A witness that equals $\overline{x}_{v_{new}}$ and whose representative is v_{new} is added to the witness state set as is presented in **Step 7**.

6.3 HySST Algorithm

Following the overview above, the proposed algorithm is given in Algorithm 6. The inputs of Algorithm 6 are the problem $\mathcal{P}^* = (X_0, X_f, X_u, (C, f, D, g), c)$, the input library ($\mathcal{U}_C$,$\mathcal{U}_D$), a parameter $p_n \in (0, 1)$, which tunes the probability of evolving with the flow regime or the jump regime, an upper bound $K \in \mathbb{N}_{>0}$ for the number of iterations to execute, and two tunable sets $X_c \supset \overline{C'}$ and $X_d \supset D'$, which act as constraints in finding a closest vertex to x_{rand}. In addition, HySST requires parameters δ_{BN} and δ_s to tune the radius of random state neighborhood and closest witness neighborhood, respectively. Each function in Algorithm 2 is defined next.

6.3.1 $\mathcal{T}$.`init`(X_0)

The function call $\mathcal{T}$.`init` is used to initialize a search tree $\mathcal{T} = (V, E)$. It randomly selects a finite number of points from X_0. For each sampling point x_0, a vertex v_0 associated with x_0 is added to V. At this step, no edge is added to E.

6.3.2 return ← is_vertex_locally_the_best($x, cost, S, \delta_s$)

The function call `is_vertex_locally_the_best` describes the conditions under which the state x is considered for addition to the search tree as is shown in Algorithm 7. First, this function searches for the closest witness s_{new} to x from the witness set S (line 1). If the closest witness distance to x is larger than δ_s, a new witness is added to S (lines 2 - 6). If s_{new} is just added as a witness or *cost* is less than the cost of the closest witness's representatives (line 7), then the state x with the cost *cost* is locally optimal and a `true` signal is returned (line 8). Otherwise, a `false` signal is returned.

6.3.3 $(S, V_{active}, V_{inactive}, E)$ ← prune_dominated_vertices(v, S, V_{active}, $V_{inactive}, E$)

The function call `prune_dominated_vertices` describes the pruning process as in Algorithm 8. First, this function searches for the witnesses s_{new} that are closest to $\overline{x}_v$ and their representatives v_{peer} (lines 1 - 2). Then, v_{peer} is moved from V_{active} to $V_{inactive}$ (lines 4 - 5) and, consequently, v replaces v_{peer} as the representative of s_{new} (line 7). If v_{peer} is a leaf vertex, then it can also safely be removed from the search tree (lines 8 - 13). The removal of v_{peer} may cause a cascading effect for its parents, if they have already been in the inactive set $V_{inactive}$ and the only reason they were maintained in the search tree was because they were leading to v_{peer}. Here, the function call `isleaf`(v_{peer}) returns `true` signal if v_{peer} is a leaf vertex, which means v_{peer} have no child vertices (line 8). The function call `parent`(v_{peer}) returns the parent vertex of v_{peer} (line 9).

Algorithm 6 HySST algorithm

Input: $X_0, X_f, X_u, c, \mathcal{H} = (C, f, D, g), (\mathcal{U}_C, \mathcal{U}_D), p_n \in (0,1)$, $K \in \mathbb{N}$, X_c, X_d, δ_{BN} and δ_s

$\mathcal{T}$.`init`(X_0);

$V_{active} \leftarrow V$, $V_{inactive} \leftarrow \emptyset$, $S \leftarrow \emptyset$;

for all $v_0 \in V$ **do**

  if `is_vertex_locally_the_best`($\overline{x}_{v_0}, 0, S, \delta_s$) **then**

    $(S, V_{active}, V_{inactive}, E) \leftarrow$ `prune_dominated_vertices`($v_0, S, V_{active}, V_{inactive}, E$)

  end if

end for

for $k = 1$ to K **do**

  randomly select a real number r from $[0, 1]$;

  if $r \leq p_n$ **then**

    $x_{rand} \leftarrow$ `random_state`($\overline{C'}$);

    $v_{cur} \leftarrow$ `best_near_selection`(x_{rand}, V_{active},δ_{BN}, X_c);

  else

    $x_{rand} \leftarrow$ `random_state`(D');

    $v_{cur} \leftarrow$ `best_near_selection`($x_{rand}, V_{active}, \delta_{BN}, X_d$);

  end if

  (`is_a_new_vertex_generated`, $x_{new}, \psi_{new}, cost_{new}$) $\leftarrow$ `new_state`($v_{cur}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u$)

  if `is_a_new_vertex_generated` & `is_vertex_locally_the_best`($x_{new}, cost_{new}, S, \delta_s$) **then**

    $v_{new} \leftarrow V_{active}$.`add_vertex`($x_{new}, cost_{new}$);

    E.`add_edge`($v_{cur}, v_{new}, \psi_{new}$);

    $(S, V_{active}, V_{inactive}, E) \leftarrow$ `prune_dominated_vertices`($v_{new}, S, V_{active}, V_{inactive}, E$);

  end if

end for

return $\mathcal{T}$;

Algorithm 7 `is_vertex_locally_the_best`$(x, cost, S, \delta_s)$

1: $s_{new} \leftarrow$ `nearest`(S, x);
2: **if** $|x - s_{new}| > \delta_s$ **then**
3: $\quad s_{new} \leftarrow x$
4: $\quad s_{new}.rep \leftarrow NULL$
5: $\quad S \leftarrow S \cup \{s_{new}\}$;
6: **end if**
7: **if** $s_{new}.rep == NULL$ or $cost < \overline{c}_{s_{new}.rep}$ **then**
8: $\quad$ **return** true;
9: **end if**
10: **return** false;

6.3.4 $x_{rand} \leftarrow$ `random_state`(S)

The function call `random_state` randomly selects a point from $S \subset \mathbb{R}^n$.

6.3.5 $v_{cur} \leftarrow$ `best_near_selection`$(x_{rand}, V_{active}, \delta_{BN}, X_\star)$

The function call `best_near_selection` searches for a vertex v_{cur} in the active vertex set V_{active} such that its associated state value is in the intersection between the set $X_\star$ and $x_{rand} + \delta_{BN}\mathbb{B}$, and has minimal cost, where $\star$ is either c or d. This function is implemented by solving the following optimization problem.

Problem 6.4. *Given $x_{rand} \in \mathbb{R}^n$, a radius $\delta_{BN} > 0$ of the random state neighborhood,*

Algorithm 8 $(S, V_{active}, V_{inactive}, E) \leftarrow$ `prune_dominated_vertices`$(v, S, V_{active}, V_{inactive}, E)$

1: $s_{new} \leftarrow$ `nearest`$(S, \overline{x}_v)$;

2: $v_{peer} \leftarrow s_{new}.rep$;

3: **if** $v_{peer} != NULL$ **then**

4: $\quad V_{active} \leftarrow V_{active} \backslash \{v_{peer}\}$;

5: $\quad V_{inactive} \leftarrow V_{inactive} \cup \{v_{peer}\}$;

6: **end if**

7: $s_{new}.rep \leftarrow v$;

8: **while** `isleaf`(v_{peer}) and $v_{peer} \in V_{inactive}$ **do**

9: $\quad v_{parent} \leftarrow$ `parent`(v_{peer});

10: $\quad E \leftarrow E \backslash \{(v_{parent}, v_{peer})\}$;

11: $\quad V_{inactive} \leftarrow V_{inactive} \backslash v_{peer}$;

12: $\quad v_{peer} \leftarrow v_{parent}$;

13: **end while**

a tunable state constraint set $X_\star$, and an active vertex set V_{active}, solve

$$\begin{aligned} \underset{v \in V_{active}}{\arg\min} \quad & \overline{c}_v \\ s.t. \quad & |\overline{x}_v - x_{rand}| \leq \delta_{BN} \\ & \overline{x}_v \in X_\star. \end{aligned}$$

Data of Problem 4.5 comes from the arguments of `best_near_selection` function call. This optimization problem is solved by traversing all the vertices in V_{active}.

6.3.6 $(\texttt{is\_a\_new\_vertex\_generated}, x_{new}, \psi_{new}, cost_{new}) \leftarrow \texttt{new\_state}(v_{cur}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u)$

If $\overline{x}_{v_{cur}} \in \overline{C'} \backslash D'$ (respectively, $\overline{x}_{v_{cur}} \in D' \backslash \overline{C'}$), the function call *new_state* generates a new solution pair ψ_{new} to the hybrid dynamical system $\mathcal{H}$ starting from $\overline{x}_{v_{cur}}$ by applying an input signal $\tilde{u}$ (respectively, an input value u_D) randomly selected from $\mathcal{U}_C$ (respectively, $\mathcal{U}_D$). If $\overline{x}_{v_{cur}} \in \overline{C'} \cap D'$, then this function generates ψ_{new} by randomly selecting flow or jump. The final state of $\psi_{new} = (\phi_{new}, u_{new})$ is denoted as x_{new}. The cost $cost_{new}$ at x_{new} is computed by $cost_{new} \leftarrow \overline{c}_{v_{cur}} + c(\phi_{new})$. After ψ_{new} and x_{new} are generated, the function `new_state` checks if there exists $(t, j) \in \text{dom } \psi_{new}$ such that $\psi_{new}(t, j) \in X_u$. If so, we have `is_a_new_vertex_generated` $\leftarrow$ `false`. Otherwise, we have `is_a_new_vertex_generated` $\leftarrow$ `true`.

6.3.7 $v_{new} \leftarrow V_{active}.\texttt{add\_vertex}(x_{new}, cost_{new})$ **and** $E.\texttt{add\_edge}(v_{cur}, v_{new}, \psi_{new})$

The function call V_{active}.`add_vertex` adds a new vertex v_{new} to V_{active} such that $\overline{x}_{v_{new}} \leftarrow x_{new}$ and $\overline{c}_{v_{new}} \leftarrow cost_{new}$ and, consequently, returns v_{new}. The function

call E.`add_edge` adds a new edge $e_{new} = (v_{cur}, v_{new})$ associated with ψ_{new} to E.

6.4 Asymptotic Near-optimality Analysis

This section analyzes the asymptotic optimality property of HySST algorithm. The following assumption assumes that the cost functional is Lipchitz continuous along the purely continuous solution pairs, locally bounded at jumps, and satisfies additivity, monotonicity, and non-degeneracy.

Assumption 6.5. *The cost functional $c : \hat{\mathcal{S}}^{\phi}_{\mathcal{H}} \to \mathbb{R}_{\geq 0}$ satisfies the following:*

1. *It is Lipschitz continuous for all continuous solution pairs (ϕ_0, u_0) and (ϕ_1, u_1) to $\mathcal{H}$ such that $\phi_0(0,0) = \phi_1(0,0)$; specifically, there exists $K_c > 0$ such that $|c(\phi_0) - c(\phi_1)| \leq K_c \sup_{(t,0) \in dom\ \phi_0 \cap dom\ \phi_1} \{|\phi_0(t,0) - \phi_1(t,0)|\}$.*

2. *For each pair of purely discrete solution pairs (ϕ_0, u_0) and (ϕ_1, u_1) to $\mathcal{H}$ such that $dom\ \phi_0 = dom\ \phi_1 = \{0\} \times \{0,1\}$ and $\phi_0(0,0) = \phi_1(0,0)$, there exists $K_d > 0$ such that $|c(\phi_0) - c(\phi_1)| \leq K_d \sup_{j \in \{0,1\}} \{|\phi_0(0,j) - \phi_1(0,j)|\}$.*

3. *Consider two solution pairs $\psi_0 = (\phi_0, u_0)$ and $\psi_1 = (\phi_1, u_1)$, and let their concatenation be $\psi_0 | \psi_1$. The following hold:*

   (a) $c(\phi_0 | \phi_1) = c(\phi_0) + c(\phi_1)$ (additivity);

   (b) $c(\phi_1) \leq c(\phi_0 | \phi_1)$ (monotonicity);

   (c) For each $t_2 > t_1 \geq 0$ such that $(t_1, j) \in dom\ \psi_0$ and $(t_2, j) \in dom\ \psi_0$ for some $j \in \mathbb{N}$, there exists $M_c > 0$ such that $t_2 - t_1 \leq M_c |c(\phi_0(t_2, j)) - c(\phi_0(t_1, j))|$

(non-degeneracy during flows).

(d) For each $j_1, j_2 \in \mathbb{N}$ *such that* $j_2 > j_1$, $(t, j_1) \in \text{dom } \psi_0$ *and* $(t, j_2) \in \text{dom } \psi_0$ *for some* $t \in \mathbb{R}_{\geq 0}$, *there exists* $M_d > 0$ *such that* $j_2 - j_1 \leq M_d |c(\phi_0(t, j_2)) - c(\psi_0(t, j_1))|$ *(non-degeneracy at jumps).*

Remark 6.6. *Items 1) and 2) above guarantee that the cost of the nearby solution pairs are bounded by the distance between the solutions. Item 3) above guarantees that the cost of the solution pairs can be computed incrementally and that the global minimum of the cost functional can be found by the optimal motion planning problem.*

Assumption 6.7. *The optimal motion plan to the optimal motion planning problem has positive safety clearance.*

The following assumption relating the safety clearance δ_s of the optimal motion plan and the inflation parameter δ_f with the algorithm parameters δ_{BN} and δ_s guarantees that the pruning process maintains at least one vertex close to the optimal motion planning if such vertex has been generated.

Assumption 6.8. *The parameters* δ_{BN} *and* δ_s *need to satisfy* $\delta_{BN} + 2\delta_s < \min\{\delta_s, \delta_f\}$.

We are ready to provide our main result, which states that, by feeding the inflation $\mathcal{H}_{\delta_f}$, HySST returns a motion plan with cost that is close to the minimal cost regardless of the positive dynamics clearance. Note that Lemmas D.3, 4.33, and 4.35 provide lower bounds over the probability of executing correct selection in `best_near_selection` and correct propagation in `new_state`, and Lemma D.1 guarantees that the pruning

process will not prune any vertex that help decrease the cost of the generated motion plan.

Theorem 6.9. *Given an optimal motion planning problem* $\mathcal{P}^* = (X_0, X_f, X_u, (C, f, D, g), c)$, *suppose Assumptions 6.5, 6.8, 4.24, and 4.27 are satisfied and that there exists an optimal motion plan* $\psi^* = (\phi^*, u^*)$ *to* $\mathcal{P}^*$ *satisfying Assumption 6.7 for some* $\delta_s > 0$. *Then, using a complete input library and when executed uniformly (as defined in Definition 4.18) to solve the motion planning problem* $\mathcal{P}^*_{\delta_f} = (X_0, X_f, X_u, (C_{\delta_f}, f_{\delta_f}, D_{\delta_f}, g_{\delta_f}), c)$ *where, for some* $\delta_f > 0$, $(C_{\delta_f}, f_{\delta_f}, D_{\delta_f}, g_{\delta_f})$ *denotes* δ_f*-inflation of* (C, f, D, g), *the probability that HySST finds a motion plan* $\psi = (\phi, u)$ *such that* $c(\phi) \leq (1 + \alpha\delta)c(\phi^*)$ *converges to one as the number of iterations* k *approaches infinity, where* $\alpha \geq 0$ *and* $\delta = \min\{\delta_s, \delta_f\}$.

Proof. See Appendix D.2. □

6.5 HySST Software Tool for Optimal Motion Planning Problems for Hybrid Dynamical Systems

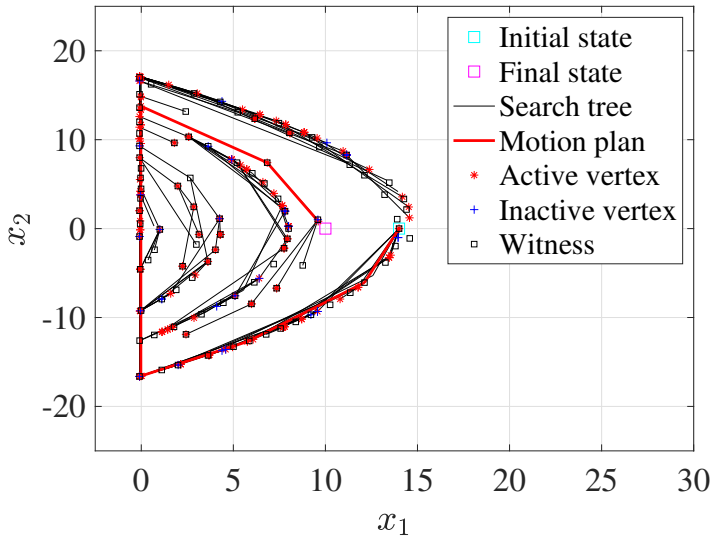
Algorithm 6 has been implemented in a software tool[1] to solve the optimal motion planning problems for hybrid dynamical systems. This software only requires the inputs listed in Algorithm 6. Next, the HySST algorithm and this tool are illustrated in Example 6.2 and Example 6.3.

Example 6.10 (Actuated bouncing ball system in Example 6.2, revisited)**.** *The simulation result in Figure 6.3 shows that HySST is able to find a motion plan for the instance*

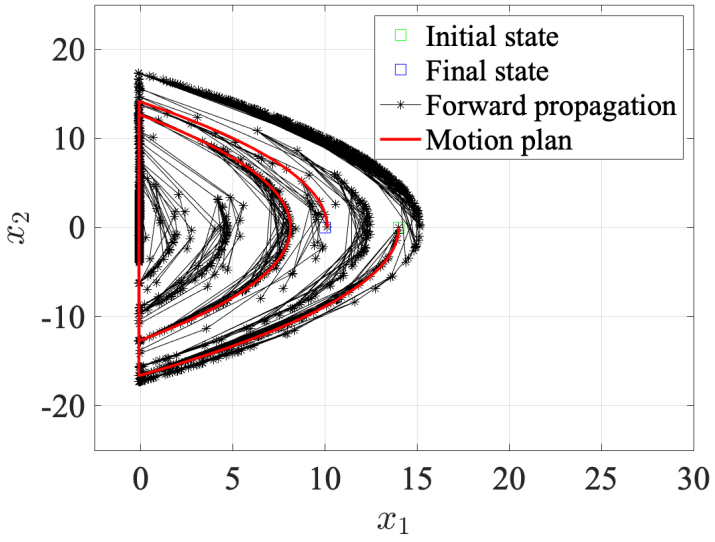
[1] Code at https://github.com/HybridSystemsLab/hybridSST.

of optimal motion planning problem for the actuated bouncing ball system. The simulation is implemented in MATLAB and processed by a 3.5 *GHz Intel Core i7 processor. Both HySST and HyRRT are run for* 20 *times to solve the same problem. The HySST creates* 154 *active vertices and* 35 *inactive vertices and takes* 3.30 *seconds, while HyRRT creates* 660 *vertices in total and takes* 18.4 *seconds on average. As is shown in Figure 6.3(a), only one jump occurs in the motion plans generated by HySST. Compared to the motion plans generated by HyRRT in Figure 6.3(b) where multiple jumps occur, the motion plan generated by HySST takes less hybrid time.*

Example 6.11. *(Collision-resilient tensegrity multicopter in Example 6.3, revisited) In this example, the restitution coefficient in 6.11 is set as* 0.43 *and the constant* κ *in (6.12) is set as* 0.20*. The simulation result in Figure 6.4 shows that HySST is able to ultilize the collision with the wall to decrease the hybrid time of the motion plan for multicopter. The simulation for this problem takes* 54.7 *seconds and creates* 2094 *active vertices on average.*



(a) HySST



(b) HyRRT

Figure 6.3: Motion plans for actuated bouncing ball example solved by HySST and HyRRT in [56].

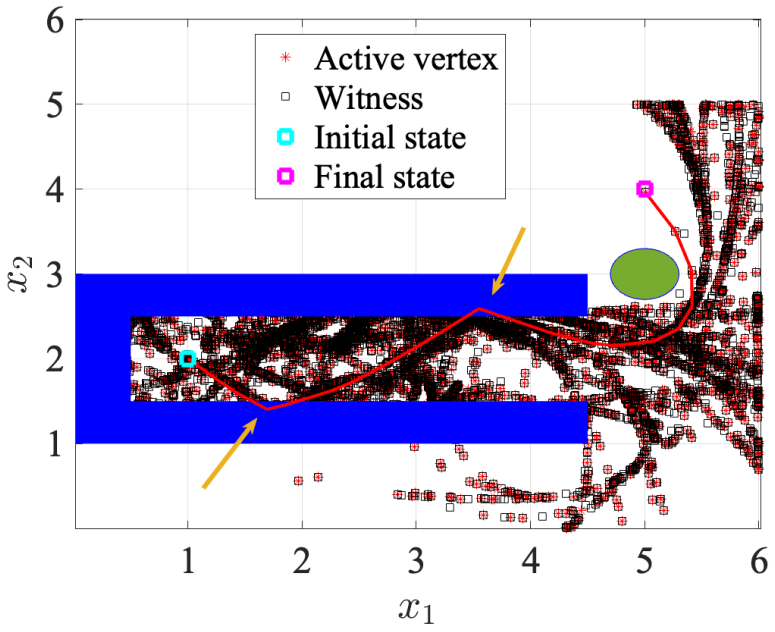


Figure 6.4: The motion plan generated by HySST for the collision-resilient tensegrity multicopter in Example 6.3. The blue rectangles denote the walls where collisions potentially occur. The green circle denotes the forbidden zone. The yellow arrows point to the location where collisions occur.

Chapter 7

Conclusion and Future Work

In this dissertation, we proposed algorithms for motion planning and tracking control for hybrid dynamical systems. In this chapter, we present a summary of the major contributions and describe several potential future research directions.

7.1 Summary

In chapter 3, we first define the motion planning problem for hybrid dynamical systems using the hybrid equation framework, which is general to capture most hybrid dynamical systems. To overcome the lack of the systematic analysis on the propagation, reversal, concatenation, and truncation operations, which are used in almost all motion planning algorithms, on the solutions to hybrid dynamical systems, we formalized the definitions of those operations for the hybrid dynamical systems and validated them theoretically. We proposed a bidirectional propagation algorithm template that describes a general framework using the aforementioned operations to solve the motion planning

problem for hybrid dynamical systems.

In chapter 4, we designed an RRT implementation of the proposed algorithm template. At each iteration, the proposed algorithm, called HyRRT, randomly picks a state sample and extends the search tree by flow or jump, which is also chosen randomly when both regimes are possible. Through a definition of concatenation of functions defined on hybrid time domains, we showed that HyRRT is probabilistically complete, namely, the probability of failing to find a motion plan approaches zero as the number of iterations of the algorithm increases. This property is guaranteed under mild conditions on the data defining the motion plan, which include a relaxation of the usual positive clearance assumption imposed in the literature of classical systems. The motion plan is computed through the solution of two optimization problems, one associated with the flow and the other with the jumps of the system. The proposed algorithm is applied to an actuated bouncing ball system and a walking robot system so as to highlight its generality and computational features.

In chapter 5, we designed a bidirectional RRT-type algorithm to solve the motion planning problem for hybrid dynamical systems. The proposed algorithm, called HyRRT-Connect, propagates in both forward and backward directions in hybrid time until an overlap between the forward and backward propagation results is detected. Then, HyRRT-Connect constructs a motion plan through the reversal and concatenation of functions defined on hybrid time domains, ensuring the motion plan thoroughly satisfies the given hybrid dynamics. To address the potential discontinuity along the flow caused by (5.3), we reconstruct the backward partial motion plan by a forward-in-hybrid-time

simulation from the final state of the forward partial motion plan. By applying the reversed input of the backward partial motion plan, the reconstruction process effectively eliminates the discontinuity and ensures that as the tolerance distance decreases to zero, the distance between the endpoint of the reconstructed motion plan and the final state set approaches zero. The proposed algorithm is applied to an actuated bouncing ball example and a walking robot example so as to highlight its generality and computational improvement.

In chapter 6, we proposed a SST-type algorithm to solve the optimal motion planning problem for hybrid dynamical systems. At each iteration, the proposed algorithm, called HySST, selects a vertex with minimal cost among all the vertices within the neighborhood of a random sample, subsequently extending the search tree through flow or jump, which is also chosen randomly when both regimes are possible. In addition, HySST maintains a static set of witness points where all vertices within each witness's neighborhood are pruned, except for the ones with lowest cost. We show that HySST is asymptotically near-optimal, namely, the probability of failing to find a motion plan with cost close to the optimal approaches zero as the number of iterations of the algorithm increases to infinity. The proposed algorithm is applied to an actuated bouncing ball system and a collision-resilient tensegrity multicopter system so as to highlight its generality and computational features.

7.2 Future Directions

The following research directions arise from the results in this dissertation.

- **Motion planning for hybrid dynamical systems under uncertainty:** While the algorithms in this dissertation operate under the assumption of a fully deterministic hybrid dynamical system, it is essential to recognize the prevalence of uncertainty in robotics applications. Uncertainty permeates various aspects, including perception, sensor measurements, numerical computations in motion planning and control, control actuation on the plant, and system dynamics. Analyzing uncertainty within hybrid dynamics poses additional challenges, particularly regarding uncertainties related to flow and jump sets, potentially resulting in probability distributions across numerous possible hybrid time domains. This research holds significant practical value; for instance, in scenarios like walking robots and collision-resilient aerial vehicles, accurately estimating the time to collide (jump) and the state before the jump is exceedingly difficult. This uncertainty can lead to entirely different hybrid time domain structures and potentially different states after the jump. Incorporating uncertainty into motion planning and minimizing its impact would greatly benefit practical applications.

- **Reactive motion planning for hybrid dynamical systems:** When multiple robotics agents with hybrid dynamics collaborate to accomplish a task, a centralized sampling-based algorithm may not be optimal. In such scenarios, distributed reactive planning methods could be more suitable. Approaches guided by Control

Lyapunov Functions (CLF) and Control Barrier Functions (CBF) show promise in this regard, as foundational research has already been conducted in these areas. These methods enable decentralized decision-making and coordination among multiple agents, making them well-suited for multi-agent hybrid dynamical systems.

- **Feedback planning for hybrid dynamical systems:** In this dissertation, our focus lies on an open-loop hybrid dynamical system as the plant. However, to execute the motion plan effectively, an additional tracking controller must be designed to close this loop. Alternatively, closed-loop hybrid dynamical systems with an external reference can be explored, where stability is inherently achieved, obviating the need for specific tracking controller designs. Additionally, leveraging constrained control technology in feedback hybrid dynamical systems offers the benefit of bypassing the time-consuming collision checking process.

Appendix A

Proof for Results in Chapter 3

A.1 Proof of Proposition 3.9

Given the solution pair $\psi = (\phi, \upsilon)$ to a hybrid dynamical system $\mathcal{H}$, we need to show that the reversal $\psi' = (\phi', \upsilon')$ of ψ is a compact solution pair to its backward-in-time hybrid dynamical system $\mathcal{H}^{\text{bw}}$. The following items are showing that ψ' satisfies each condition in Definition 2.5.

- The first item is to prove that the domain dom ϕ' equals dom υ' and dom $\psi' :=$ dom $\phi' =$ dom υ' is a compact hybrid time domain.

  Since $\psi = (\phi, \upsilon)$ is a solution pair to $\mathcal{H}$, then

  $$\text{dom } \phi = \text{dom } \upsilon.$$

  Due to Definition 3.4, dom $\phi' = \{(T, J)\} -$ dom ϕ and dom $\upsilon' = \{(T, J)\} -$ dom υ

where $(T, J) = \max \text{dom } \psi$. Therefore,

$$\text{dom } \phi' = \{(T, J)\} - \text{dom } \phi = \{(T, J)\} - \text{dom } \upsilon = \text{dom } \upsilon'.$$

Thus, dom ϕ' = dom υ' is proved.

Then we want to show that dom ψ' = dom ϕ' = dom υ' is a hybrid time domain. Since ψ is a compact solution pair, then dom ψ is a compact hybrid time domain. Therefore,

$$\text{dom } \psi = \cup_{j=0}^{J}([t_j, t_{j+1}], j) \tag{A.1}$$

holds for some finite sequence of times,

$$0 = t_0 \leq t_1 \leq t_2 \leq ... \leq t_{J+1} = T \tag{A.2}$$

where $(T, J) = \max \text{dom } \psi$.

Since dom ψ' = $\{(T, J)\}$ − dom ψ, then

$$\begin{aligned} \text{dom } \psi' &= \{(T, J)\} - \text{dom } \psi \\ &= \{(T, J)\} - \cup_{j=0}^{J}([t_j, t_{j+1}], j) \\ &= \cup_{j=0}^{J}([T - t_{j+1}, T - t_j], J - j). \end{aligned} \tag{A.3}$$

Let $t'_j = T - t_{J+1-j}$, where t_j is the sequence of time in (A.2). Since (A.2) holds, then

$$0 = t'_0 \leq t'_1 \leq t'_2 \leq ... \leq t'_{J+1} = T.$$

Therefore, (A.3) can be written as

$$\text{dom } \psi' = \cup_{j=0}^{J}([t'_j, t'_{j+1}], j). \tag{A.4}$$

Hence, dom ψ' is a compact hybrid time domain.

- The second item is to prove that $\psi'(0,0) \in \overline{C^{\text{bw}}} \cup D^{\text{bw}}$. Since ψ is compact, the hybrid time $(T, J) \in \text{dom } \psi$, where $(T, J) = \max \text{dom } \psi$. The state $\phi(T, J)$ is either reached during a flow or at a jump.

  1. Consider the case when $\phi(T, J)$ is reached during a flow. In this case, $I_\psi^J = I_{\psi'}^0$ has nonempty interior.

     Definition 3.4 suggests that

     $$\phi'(0,0) = \phi(T, J)$$

     and $\upsilon'(0,0)$ is such that $(\phi'(0,0), \upsilon'(0,0)) \in \overline{C} = \overline{C^{\text{bw}}}$. Therefore, $\psi'(0,0) = (\phi'(0,0), \upsilon'(0,0)) \in \overline{C^{\text{bw}}} \subset \overline{C^{\text{bw}}} \cup D^{\text{bw}}$.

  2. Consider the case when $\psi(T, J)$ is reached at a jump. In this case,

     $$(T, J-1) \in \text{dom } \psi \quad (T, J) \in \text{dom } \psi.$$

     According to Definition 2.5, the state input pair

     $$\psi(T, J-1) = (\phi(T, J-1), \upsilon(T, J-1)) \in D$$

     $$\phi(T, J) = g(\phi(T, J-1), u(T, J-1)).$$

     According to Definition 3.4,

     $$\phi'(0,0) = \phi(T, J) \quad \upsilon'(0,0) = \upsilon'(T, J-1).$$

     Therefore,

     $$(\phi'(0,1), \upsilon'(0,0)) \in D \quad \phi'(0,0) = g(\phi'(0,1), \upsilon'(0,0)).$$

Since D^{bw} is defined as

$$D^{\mathrm{bw}} = \{(x,u) : \exists z \in g^{\mathrm{bw}}(x,u) : (z,u) \in D\},$$

where g^{bw} is defined as

$$g^{\mathrm{bw}}(x,u) = \{z : x = g(z,u), (z,u) \in D\},$$

therefore,

$$\phi'(0,1) \in g^{\mathrm{bw}}(\phi'(0,0), \upsilon'(0,0))$$

and

$$\psi'(0,0) = (\phi'(0,0), \upsilon'(0,0)) \in D^{\mathrm{bw}}.$$

Thus, $\psi'(0,0) = (\phi'(0,0), \upsilon'(0,0)) \in D^{\mathrm{bw}} \subset \overline{C^{\mathrm{bw}}} \cup D^{\mathrm{bw}}$.

In conclusion, $\psi'(0,0) = (\phi'(0,0), \upsilon'(0,0)) \in \overline{C^{\mathrm{bw}}} \cup D^{\mathrm{bw}}$.

- This item is to prove that ψ' satisfies the conditions in the first item in Definition 2.5. The following items are to prove each of these conditions is satisfied.

  (a) This item is to prove that for all $j \in \mathbb{N}$ such that $I^j_{\psi'}$ has nonempty interior, ϕ' is absolutely continuous on each $I^j_{\phi'} = \{t : (t,j) \in \mathrm{dom}\ \phi'\}$ with nonempty interior.

  Due to $\mathrm{dom}\ \psi' = \{(T,J)\} - \mathrm{dom}\ \psi$, then $(t,j) \in \mathrm{dom}\ \psi'$ implies $(T-t, J-j) \in \mathrm{dom}\ \psi$. Hence,

  $$I^j_{\psi'} = \{T\} - I^{J-j}_{\psi}. \tag{A.5}$$

  Therefore, $\mathrm{int}\, I^j_{\psi'} \neq \emptyset$ implies that $\mathrm{int}\, I^{J-j}_{\psi} \neq \emptyset$.

Since $\psi = (\phi, \upsilon)$ is a solution pair to $\mathcal{H}$, the function $t \mapsto \phi(t, j)$ is locally absolutely continuous for each I_ψ^j with nonempty interior. Therefore, $t \mapsto \phi(T - t, J - j)$ is locally absolutely continuous for each $\{T\} - I_\psi^{J-j}$ with nonempty interior.

According to Definition 3.4,

$$\phi'(t, j) = \phi(T - t, J - j)$$

for all $(t, j) \in \operatorname{dom} \phi'$. Therefore, the function $t \mapsto \phi'(t, j) = \phi(T - t, J - j)$ is locally absolutely continuous for each $I_{\psi'}^j = \{T\} - I_\psi^{J-j}$ with nonempty interior.

(b) This item is to prove that for all $j \in \mathbb{N}$ such that $I_{\psi'}^j$ has nonempty interior, $\psi'(t, j) = (\phi'(t, j), \upsilon'(t, j)) \in C^{\text{bw}}$ for all $t \in \operatorname{int} I_{\psi'}^j$.

Since $I_{\psi'}^j = \{T\} - I_\psi^{J-j}$ and $I_{\psi'}^j$ has nonempty interior, then $\{T\} - I_\psi^{J-j}$ has nonempty interior. Because $\psi = (\phi, \upsilon)$ is a solution pair to $\mathcal{H}$, according to Definition 2.5, then $\psi(t, j) \in C$ for all $t \in \operatorname{int} I_\psi^j$, where I_ψ^j has nonempty interior.

Since $\{T\} - I_\psi^{J-j}$ has nonempty interior, then

$$\psi(T - t, J - j) = (\phi(T - t, J - j), \upsilon(T - t, J - j)) \in C$$

for all $T - t \in I_\psi^{J-j}$.

According to Definition 3.4, since $I_{\psi'}^j$ has nonempty interior, then

$$\phi'(t, j) = \phi(T - t, J - j) \quad \upsilon'(t, j) = \upsilon(T - t, J - j).$$

Hence, state input pair $\psi'(t,j) = (\phi'(t,j), \upsilon'(t,j)) \in C = C^{\text{bw}}$ for all $t \in \operatorname{int} I^j_{\psi'}$, where $I^j_{\psi'}$ has nonempty interior.

(c) This item is to prove that for all $j \in \mathbb{N}$ such that $I^j_{\psi'}$ has nonempty interior, the function $t \mapsto \upsilon'(t,j)$ is Lebesgue measurable and locally bounded.

– This item is to show that $t \mapsto \upsilon'(t,j)$ is Lebesgue measurable for all $j \in \mathbb{N}$ such that $I^j_{\upsilon'}$ has nonempty interior.

If $I^j_{\upsilon'}$ has nonempty interior, then $I^{J-j}_{\upsilon} = \{T\} - I^j_{\upsilon'}$ has nonempty interior. Therefore, $t \mapsto u(t, J-j)$ is Lebesgue measurable. Let υ_i denote the ith component of υ. For all $a \in \mathbb{R}$ and $i \in \{1, 2, ..., m\}$, since $t \mapsto \upsilon(t, J-j)$ is Lebesgue measurable, $S_i := \{t \in I^{J-j}_{\upsilon_i} : \upsilon_i(t, J-j) > a\}$ is Lebesgue measurable.

Note that

$$\begin{aligned}
S_i &= \{t \in I^{J-j}_{\upsilon_i} : \upsilon_i(t, J-j) > a\} \\
&= \{t \in \{T\} - I^{J-j}_{\upsilon_i} : \upsilon_i(T-t, J-j) > a\} \\
&= \{t \in I^j_{\upsilon'} : \upsilon_i(T-t, J-j) > a\} \\
&= \{t \in \operatorname{int} I^j_{\upsilon'} : \upsilon_i(T-t, J-j) > a\} \\
&\cup \{t \in \partial I^j_{\upsilon'} : \upsilon_i(T-t, J-j) > a\} \\
&= \{t \in \operatorname{int} I^j_{\upsilon'} : \upsilon'_i(t,j) > a\} \cup \{t \in \partial I^j_{\upsilon'} : \upsilon_i(T-t, J-j) > a\} \\
&=: S_i^{\text{int}} \cup S_i^{\partial}.
\end{aligned} \tag{A.6}$$

Therefore, $S_i^{\text{int}} = S_i^{\text{int}} \cup S_i^{\partial}$ is Lebesgue measurable. Let $I^j_{\upsilon'} = [T_1, T_2]$,

then S_i^{∂} can be one of the following:

$$\emptyset, \{T_1\}, \{T_2\}, \{T_1, T_2\}.$$

Since all of the above are Lebesgue measurable, then S_i^{∂} is Lebesgue measurable. Since S_i^{∂} is measurable, then its complement $\mathbb{R}\backslash S_i^{\partial}$ is Lebesgue measurable. Hence,

$$S_i^{\text{int}} = S_i \cap (\mathbb{R}\backslash S_i^{\partial})$$

is Lebesgue measurable.

Since we want to show that $t \mapsto v'(t, j)$ is Lebesgue measurable for all $j \in \mathbb{N}$ such that $I_{v'}^j$ has nonempty interior, it is equivalent to show that $S_i' := \{t \in I_{v'}^j : v_i'(t, j) > a\}$ is Lebesgue measurable for all $a \in \mathbb{R}$, $i \in \{1, 2, ..., m\}$ and $j \in \mathbb{N}$ such that $I_{v'}^j$ has nonempty interior.

Note that

$$\begin{aligned} S_i' &= \{t \in \operatorname{int} I_{v'}^j : v_i'(t, j) > a\} \cup \{t \in \partial I_{v'}^j : v_i'(t, j) > a\} \\ &= S_i^{\text{int}} \cup \{t \in \partial I_{v'}^j : v_i'(t, j) > a\} \\ &=: S_i^{\text{int}} \cup S_i^{\partial'} \end{aligned} \tag{A.7}$$

and S_i^{int} is Lebesgue measurable. Hence, the Lebesgue measurability of S_i' depends on that of set $S_i^{\partial'}$. Similarly, let $I_{v'}^j = [T_1', T_2']$, then $S_i^{\partial'}$ can be one of the following:

$$\emptyset, \{T_1'\}, \{T_2'\}, \{T_1', T_2'\}.$$

Since all of the above are Lebesgue measurable, then $S_i^{\partial'}$ is Lebesgue measurable. Therefore, $S_i' = S_i^{\text{int}} \cup S_i^{\partial'}$ is Lebesgue measurable for all

$a \in \mathbb{R}$, $i \in \{1, 2, ..., m\}$ and $j \in \mathbb{N}$ such that $I^j_{v'}$ has nonempty interior. Hence, $t \mapsto v'(t, j)$ is Lebesgue measurable for all $j \in \mathbb{N}$ such that $I^j_{v'}$ has nonempty interior.

– This item is to show that $t \mapsto v'(t, j)$ is locally bounded for all $j \in \mathbb{N}$ such that $I^j_{v'} := \{t \in \mathbb{R}_{\geq 0} : (t, j) \in \text{dom } v'\}$. In other words, we want to show that for all $j \in \mathbb{N}$ such that $I^j_{v'}$ has nonempty interior, $i \in \{1, 2, ..., m\}$ and any $t_0 \in I^j_{v'}$, there exists a neighborhood A' of t_0 such that for some number $M' > 0$, one has

$$|v'_i(t, j)| \leq M'$$

for all $t \in A'$.

Note that for all the $j \in \mathbb{N}$ such that I^j_v has nonempty interior, $t \mapsto v(t, j)$ is locally bounded. Therefore, for all $i \in \{1, 2, ..., m\}$ and any $t_0 \in I^j_v$, there exists a neighborhood A of t_0 such that for some number $M > 0$ one has

$$|v_i(t, j)| \leq M$$

for all $t \in A$, where v_i denotes the ith component of v.

Note that dom $v' = (T, J) - \text{dom } v$. Hence, $I^j_{v'} = \{T\} - I^{J-j}_v$. If $I^j_{v'}$ has nonempty interior, then $\{T\} - I^{J-j}_v$ has nonempty interior. Since $\{T\} - I^{J-j}_v$ has nonempty interior, $t \mapsto u(T - t, J - j)$ is locally bounded. Therefore, for all $i \in \{1, 2, ..., m\}$ and any $t_0 \in \{T\} - I^{J-j}_v = I^j_{v'}$, there

exists a neighborhood A of t_0 such that for some number $M > 0$ one has

$$|\upsilon_i(T - t, J - j)| \leq M$$

for all $t \in A$.

Note that $A = (A \cap \operatorname{int} I^j_{\upsilon'}) \cup (A \cap \partial I^j_{\upsilon'})$. For all the $t \in (A \cap \operatorname{int} I^j_{\upsilon'})$,

$$\upsilon(T - t, J - j) = \upsilon'(t, j).$$

Therefore, $|\upsilon'_i(t, j)| \leq M$ for all $t \in A \cap \operatorname{int} I^j_{\upsilon'}$.

Let $I^j_{\upsilon'} = [T_1, T_2]$. Since $I^j_{\upsilon'}$ has nonempty interior, then $T_1 < T_2$. Hence, $\partial I^j_{\upsilon'} = \{T_1, T_2\}$. Therefore, $A \cap \partial I^j_{\upsilon'} \subset \{T_1, T_2\}$.

Therefore, for all $t \in A$,

$$\upsilon'_i(t, j) \leq \max\{M, \upsilon'(T_1, j), \upsilon'(T_2, j)\}.$$

We can select $A' = A$ and $M' = \max\{M, \upsilon'(T_1, j), \upsilon'(T_2, j)\}$. Hence, the local boundness of $t \mapsto \upsilon'(t, j)$ is proved.

(d) This item is to prove that for all $j \in \mathbb{N}$ such that $I^j_{\psi'}$ has nonempty interior, for almost all $t \in I^j_{\psi'}$,

$$\dot{\phi}'(t, j) = f^{\mathrm{bw}}(\phi'(t, j), \upsilon'(t, j)). \tag{A.8}$$

For almost all $t \in I^j_{\psi'}$, the following holds:

$$\begin{aligned}\dot{\phi}'(t,j) = \dot{\phi}(T-t, J-j) &= \frac{\mathrm{d}\phi(T-t, J-j)}{\mathrm{d}(T-t)} \\ &= -f(\phi(T-t, J-j), \upsilon(T-t, J-j)) \\ &= -f(\phi'(t,j), \upsilon'(t,j)) \\ &= f^{\mathrm{bw}}(\phi'(t,j), \upsilon'(t,j)).\end{aligned} \tag{A.9}$$

Therefore,

$$\dot{\phi}'(t,j) = f^{\mathrm{bw}}(\phi'(t,j), \upsilon'(t,j)) \qquad \text{for almost all } t \in I^j_{\psi'}. \tag{A.10}$$

- The last item is to prove that for all $(t,j) \in \mathrm{dom}\ \psi'$ such that $(t, j+1) \in \mathrm{dom}\ \psi'$,

$$\begin{aligned}&(\phi'(t,j), \upsilon'(t,j)) \in D^{\mathrm{bw}} \\ &\phi'(t, j+1) = g^{\mathrm{bw}}(\phi'(t,j), \upsilon'(t,j)).\end{aligned} \tag{A.11}$$

Since the hybrid time domain $\mathrm{dom}\ \psi' = \{T, J\} - \mathrm{dom}\ \psi$, for all $(t,j) \in \mathrm{dom}\ \psi'$ such that $(t, j+1) \in \mathrm{dom}\ \psi'$, the hybrid times $(T-t, J-j-1) \in \mathrm{dom}\ \psi$ and $(T-t, J-j) \in \mathrm{dom}\ \psi$. Since $(T-t, J-j-1) \in \mathrm{dom}\ \psi$ and $(T-t, J-j) \in \mathrm{dom}\ \psi$, then

$$\begin{aligned}&\psi(T-t, J-j-1) \in D \\ &\phi(T-t, J-j) = g(\phi(T-t, J-j-1), \upsilon(T-t, J-j-1)).\end{aligned} \tag{A.12}$$

According to Definition 3.4,

$$\begin{aligned}\phi'(t,j) &= \phi(T-t, J-j) \\ \phi'(t, j+1) &= \phi(T-t, J-j-1) \\ \upsilon'(t,j) &= \upsilon(T-t, J-j-1).\end{aligned} \tag{A.13}$$

Since the state $\phi(T-t, J-j) = g(\phi(T-t, J-j-1), \upsilon(T-t, J-j-1))$, then

$$\begin{aligned}\phi'(t, j+1) &= \phi(T-t, J-j-1) \\ &\in g^{\text{bw}}(\phi(T-t, J-j), \upsilon(T-t, J-j-1)) \\ &\in g^{\text{bw}}(\phi'(t, j), \upsilon'(t, j)).\end{aligned} \tag{A.14}$$

Since the state input pair $\psi(T-t, J-j-1) = (\phi(T-t, J-j-1), \upsilon(T-t, J-j-1)) \in D$, then

$$(\phi'(t, j+1), \upsilon'(t, j)) \in D. \tag{A.15}$$

Since $\phi'(t, j+1) \in g^{\text{bw}}(\phi'(t, j), \upsilon'(t, j))$ and the jump set of the backward-in-time hybrid dynamical system is defined as (3.10), the state input pair $(\phi'(t, j), \upsilon'(t, j)) \in D^{\text{bw}}$.

Because all the conditions in Definition 2.5 are satisfied, $\psi' = (\phi', \upsilon')$ is a solution pair to $\mathcal{H}^{\text{bw}}$.

A.2 Proof of Proposition 3.13

The proof is to show that $\psi = (\phi, \upsilon) = (\phi_1|\phi_2, \upsilon_1|\upsilon_2)$ satisfies the conditions in Definition 2.5. The following items are to show that ψ satisfies each of the conditions.

- The first item is to prove that dom ϕ = dom υ and the domain dom ψ := dom ϕ = dom υ is a hybrid time domain.

  Since ψ_1 and ψ_2 are solution pairs to $\mathcal{H}$, according to Definition 2.5,

  $$\text{dom } \psi_1 = \text{dom } \phi_1 = \text{dom } \upsilon_1,$$

$$\text{dom } \psi_2 = \text{dom } \phi_2 = \text{dom } \upsilon_2$$

and dom ψ_1 and dom ψ_2 are both hybrid time domains.

Due to dom ϕ = dom $\phi_1 \cup$ (dom $\phi_2 + \{(T, J)\}$) and dom υ = dom $\upsilon_1 \cup$ (dom $\upsilon_2 +$ $\{(T, J)\}$), therefore,

$$\text{dom } \phi = \text{dom } \phi_1 \cup (\text{dom } \phi_2 + \{(T, J)\}) = \text{dom } \upsilon_1 \cup (\text{dom } \upsilon_2 + \{(T, J)\}) = \text{dom } \upsilon.$$

Because dom ψ_1 and dom ψ_2 are hybrid time domains, then for each $(T_1, J_1) \in$ dom ψ_1 and $(T_2, J_2) \in$ dom ψ_2, the intersection can be written as

$$\text{dom } \psi_1 \cap ([0, T_1] \times \{0, 1, ..., J_1\}) = \cup_{j=0}^{J_1}([t_j^1, t_{j+1}^1], j) \tag{A.16}$$

for some finite sequence of times $0 = t_0^1 \leq t_1^1 \leq t_2^1 \leq ... \leq t_{J_1+1}^1 = T_1$ and

$$\text{dom } \psi_2 \cap ([0, T_2] \times \{0, 1, ..., J_2\}) = \cup_{j=0}^{J_2}([t_j^2, t_{j+1}^2], j) \tag{A.17}$$

for some finite sequence of times $0 = t_0^2 \leq t_1^2 \leq t_2^2 \leq ... \leq t_{J_2+1}^2 = T_2$. Since dom ψ = dom $\psi_1 \cup$ (dom $\psi_2 + \{(T, J)\}$), then for each $(T', J') \in$ dom ψ,

$$\begin{aligned} &\text{dom } \psi \cap ([0, T'] \times \{0, 1, ..., J'\}) \\ &\quad = \text{dom } \psi_1 \cup (\text{dom } \psi_2 + \{(T, J)\}) \cap ([0, T'] \times \{0, 1, ..., J'\}) \\ &\quad = (\text{dom } \psi_1 \cap ([0, T'] \times \{0, 1, ..., J'\})) \cup \\ &\quad (\text{dom } \psi_2 + \{(T, J)\}) \cap ([0, T'] \times \{0, 1, ..., J'\}) \end{aligned} \tag{A.18}$$

If $T' \leq T$ and $J' \leq J$, then the intersection

$$\begin{aligned}\text{dom } \psi \cap ([0,T'] \times \{0,1,...,J'\}) &= (\text{dom } \psi_1 \cap ([0,T'] \times \{0,1,...,J'\})) \cup \\ &\quad (\text{dom } \psi_2 + \{(T,J)\}) \cap ([0,T'] \times \{0,1,...,J'\}) \\ &= \cup_{j=0}^{J'}([t_j^1, t_{j+1}^1], j) \cup \emptyset = \cup_{j=0}^{J'}([t_j^1, t_{j+1}^1], j)\end{aligned} \tag{A.19}$$

for some finite sequence of times $0 = t_0^1 \leq t_1^1 \leq t_2^1 \leq ... \leq t_{J'+1}^1 = T'$.

If $T' \geq T$ and $J' \geq J$, then the intersection

$$\begin{aligned}&\text{dom } \psi \cap ([0,T'] \times \{0,1,...,J'\}) \\ &\quad = (\text{dom } \psi_1 \cap ([0,T'] \times \{0,1,...,J'\})) \cup \\ &\quad (\text{dom } \psi_2 + \{(T,J)\}) \cap ([0,T'] \times \{0,1,...,J'\}) \\ &\quad = (\text{dom } \psi_1 \cap ([0,T] \times \{0,1,...,J\})) \cup \\ &\quad (\text{dom } \psi_2 + \{(T,J)\}) \cap ([T,T'] \times \{J,J+1,...,J'\}) \\ &\quad = (\text{dom } \psi_1 \cap ([0,T] \times \{0,1,...,J\})) \cup \\ &\quad (\text{dom } \psi_2 \cap ([0,T'-T] \times \{0,1,...,J'-J\}) + \{(T,J)\}) \\ &\quad = \cup_{j=0}^{J}([t_j^1, t_{j+1}^1], j) \cup (\cup_{j=0}^{J'-J}([t_j^2, t_{j+1}^2], j) + \{(T,J)\}) \\ &\quad = \cup_{j=0}^{J}([t_j^1, t_{j+1}^1], j) \cup (\cup_{j=0}^{J'-J}([t_j^2 + T, t_{j+1}^2 + T], j + J))\end{aligned} \tag{A.20}$$

for some finite sequence of times $0 = t_0^1 \leq t_1^1 \leq t_2^1 \leq ... \leq t_{J+1}^1 = T$ and $T = t_0^2 + T \leq t_1^2 + T \leq t_2^2 + T \leq ... \leq t_{J'-J+1}^2 + T = T'$. Hence, a finite sequence $\{t_i\}_{i=0}^{J'}$ can be constructed as follows such that $0 = t_0 \leq t_1 \leq t_2 \leq ... \leq t_{J'} = T'$:

$$t_i = \begin{cases} t_i^1 & i \leq J \\ t_{i-J}^2 + T & J+1 \leq i \leq J'. \end{cases} \tag{A.21}$$

Therefore,

$$\text{dom } \psi \cap ([0, T'] \times \{0, 1, ..., J'\}) = \cup_{j=0}^{J'}([t_j, t_{j+1}], j) \tag{A.22}$$

for a finite sequence of $\{t_i\}_{i=0}^{J'}$. Hence, dom ϕ = dom υ and the domain dom $\psi :=$ dom ϕ = dom υ is a hybrid time domain.

- This item is to prove that $\psi(0,0) = (\phi(0,0), u(0,0)) \in \overline{C} \cup D$.

  Since ψ_1 is a solution pair to $\mathcal{H}$, $\psi_1(0,0) \in \overline{C} \cup D$. According to Definition 3.5, $\psi(t,j) = \psi_1(t,j)$ for all $(t,j) \in$ dom ψ_1. Since $(0,0) \in$ dom ψ_1, $\psi(0,0) = \psi_1(0,0) \in \overline{C} \cup D$.

- This item is to prove that for all $j \in \mathbb{N}$ such that I_ψ^j has nonempty interior, ψ satisfies conditions in the first item in Definition 2.5. The following items prove that ψ satisfies each of the conditions.

  (a) This item proves that for all $j \in \mathbb{N}$ such that I_ψ^j has nonempty interior, the function $t \mapsto \phi(t,j)$ is locally absolutely continuous.

  Since dom ψ = dom $\psi_1 \cup$ (dom $\psi_2 + \{(T,J)\}$), then

  $$(t,j) \in \text{dom } \psi = \text{dom } \psi_1 \cup (\text{dom } \psi_2 + \{(T,J)\})$$

  implies that

  $$(t,j) \in \text{dom } \psi_1$$

  or

  $$(t,j) \in (\text{dom } \psi_2 + \{(T,J)\}).$$

Hence, $t \in I^j_\psi$ implies that

$$t \in I^j_{\psi_1}$$

when $j < J$ and

$$t \in I^{j-J}_{\psi_2} + \{T\}$$

when $j > J$. Therefore, the interval $I^j_\psi = I^j_{\psi_1}$ if $j < J$ and $I^j_\psi = I^{j-J}_{\psi_2} + \{T\}$ if $j > J$.

When it comes to $j = J$, we want to show that the interval $I^J_\psi = I^J_{\psi_1} \cup (I^0_{\psi_2} + \{T\})$. The two cases for time $t \in I^J_\psi$ are $(t, J) \in \text{dom } \psi_1$ or $(t, J) \in \text{dom } \psi_2 + \{(T, J)\}$. Therefore, $t \in I^J_\psi$ leads to that $t \in I^J_{\psi_1}$ or $t \in I^0_{\psi_2} + \{T\}$ which indicates that

$$I^J_\psi = I^J_{\psi_1} \cup (I^0_{\psi_2} + \{T\}).$$

The following cases are considered to prove that ϕ is absolutely continuous.

I Consider the case of $j < J$ such that $I^j_\psi = I^j_{\psi_1}$ has nonempty interior. Since ψ_1 is a solution pair to $\mathcal{H}$, according to Definition 2.5, $t \mapsto \phi_1(t, j)$ is locally absolutely continuous for $I^j_{\psi_1}$ with nonempty interior. Because $\phi(t, j) = \phi_1(t, j)$ for all $(t, j) \in \text{dom } \psi_1 \backslash (T, J)$ and $\phi(T, J) = \phi_2(0, 0) = \phi_1(T, J)$, then $\phi(t, j) = \phi_1(t, j)$ for all $t \in I^j_{\psi_1}$.

Since $I^j_\psi = I^j_{\psi_1}$, when I^j_ψ has nonempty interior, it implies that $I^j_{\psi_1}$ has nonempty interior. Therefore, $t \mapsto \phi(t, j) = \phi_1(t, j)$ is locally absolutely continuous for all I^j_ψ with nonempty interior.

II Consider the case of all $j > J$ such that $I^j_\psi = I^{j-J}_{\psi_2} + \{T\}$ has nonempty

interior. Since ψ_2 is a solution pair to $\mathcal{H}$, according to Definition 2.5, then $t \mapsto \phi_2(t, j)$ is locally absolutely continuous for interval $I^j_{\psi_2}$ with nonempty interior.

Since $I^j_\psi = I^{j-J}_{\psi_2} + \{T\}$, when I^j_ψ has nonempty interior, it implies that $I^{j-J}_{\psi_2} + \{T\}$ has nonempty interior. Due to Definition 3.5, $\phi(t, j) = \phi_2(t - T, j - J)$ for all $t \in I^{j-J}_{\psi_2} + \{T\}$. Therefore, $t \mapsto \phi(t, j) = \phi_2(t - T, j - J)$ is locally absolutely continuous for all $I^j_\psi = I^{j-J}_{\psi_2} + \{T\}$ with nonempty interior.

III If $j = J$ and $I^J_\psi = I^J_{\psi_1} \cup (I^0_{\psi_2} + \{T\})$ has nonempty interior, there are three cases to consider.

i. If $I^J_{\psi_1}$ has nonempty interior and $(I^0_{\psi_2} + \{T\})$ has empty interior, then the interval $I^J_\psi = I^J_{\psi_1}$. Since $\phi(t, J) = \phi_1(t, J)$ for all $t \in I^J_{\psi_1}$ and $t \mapsto \phi_1(t, J)$ is locally absolutely continuous for $I^J_{\psi_1}$ with nonempty interior , the function $t \mapsto \phi(t, J) = \phi_1(t, J)$ is locally absolutely continuous for $I^J_\psi = I^J_{\psi_1}$.

ii. If $I^J_{\psi_1}$ has empty interior and $(I^0_{\psi_2} + \{T\})$ has nonempty interior, then the interval $I^J_\psi = (I^0_{\psi_2} + \{T\})$. Since $\phi(t, J) = \phi_2(t - T, 0)$ for all $t \in (I^0_{\psi_2} + \{T\})$ and $t \mapsto \phi_2(t, 0)$ is locally absolutely continuous for $I^0_{\psi_2}$, the function $t \mapsto \phi(t, J) = \phi_2(t - T, 0)$ is locally absolutely continuous for $I^J_\psi = (I^0_{\psi_2} + \{T\})$.

iii. If both $I^J_{\psi_1}$ and $(I^0_{\psi_2} + \{T\})$ has nonempty interior, since $t \mapsto \phi(t, J)$ is locally absolutely continuous for $I^J_{\psi_1}$ and $(I^0_{\psi_2} + \{T\})$, respectively.

Because $\phi_1(T, J) = \phi_2(0, 0) = \phi(T, J)$, then $t \mapsto \phi(t, j)$ is locally absolutely continuous for $I^J_\psi = I^J_{\psi_1} \cup (I^0_{\psi_2} + \{T\})$.

Therefore, the function $t \mapsto \phi(t, j)$ is locally absolutely continuous for all $j \in \mathbb{N}$ such that I^j_ψ has nonempty interior.

(b) This item proves that for all $j \in \mathbb{N}$ such that I^j_ψ has nonempty interior, $\psi(t, j) \in C$ for all $t \in \operatorname{int} I^j_\psi$.

I Consider the case of all $j < J$ such that $I^j_{\psi_1}$ has nonempty interior. Since ψ_1 is a solution pair to $\mathcal{H}$, according to Definition 2.5,

$$\psi_1(t, j) \in C$$

for all

$$t \in \operatorname{int} I^j_{\psi_1}.$$

Because $\psi(t, j) = \psi_1(t, j)$ for all $t \in \operatorname{int} I^j_{\psi_1}$, then $\psi(t, j) = \psi_1(t, j) \in C$ for all $t \in \operatorname{int} I^j_\psi = I^j_{\psi_1}$.

II Consider the case of all $j > J$ such that $I^{j-J}_{\psi_2} + \{T\}$ has nonempty interior. Since ψ_2 is a solution pair to $\mathcal{H}$, according to Definition 2.5, then

$$\psi_2(t, j) \in C$$

for all

$$t \in \operatorname{int} I^j_{\psi_2}.$$

Therefore,

$$\psi_2(t - T, j - J) \in C$$

for all

$$t \in I_{\psi_2}^{j-J} + \{T\}.$$

Because $\psi(t,j) = \psi_2(t-T, j-J)$ for all $t \in I_{\psi_2}^{j-J} + \{T\}$, then

$$\psi(t,j) = \psi_2(t-T, j-J) \in C$$

holds for all $t \in \operatorname{int} I_\psi^j = I_{\psi_2}^{j-J} + \{T\}$.

III If $j = J$ and $I_\psi^J = I_{\psi_1}^J \cup (I_{\psi_2}^0 + \{T\})$ has nonempty interior, there are three cases to consider.

i. if $I_{\psi_1}^J$ has nonempty interior and $(I_{\psi_2}^0 + \{T\})$ has empty interior, then the interval $I_\psi^J = I_{\psi_1}^J$. And $\psi(t,j) = \psi_1(t,j) \in C$ holds for all $t \in \operatorname{int} I_\psi^J = I_{\psi_1}^J$.

ii. if $I_{\psi_1}^J$ has empty interior and $I_{\psi_2}^0 + \{T\}$ has nonempty interior, then the interval $I_\psi^J = I_{\psi_2}^0 + \{T\}$.Then $\psi(t,j) = \psi_2(t-T, j-J) \in C$ holds for all $t \in \operatorname{int} I_\psi^J = I_{\psi_2}^0 + \{T\}$.

iii. if both $I_{\psi_1}^J$ and $(I_{\psi_2}^0 + \{T\})$ has nonempty interior, since we have had

$$\psi(t,j) = \psi_1(t,j) \in C \quad \forall t \in \operatorname{int} I_{\psi_1}^J$$

and

$$\psi(t,j) = \psi_2(t-T, j-J) \in C \quad \forall t \in \operatorname{int} I_{\psi_2}^0 + \{T\},$$

then $\psi(t,j) \in C$ for all $t \in \operatorname{int} I_\psi^j \cup (\operatorname{int} I_{\psi_2}^0 + \{T\})$. Because $\psi_2(0,0) = \psi(T,J) \in C$, then $\psi(t,j) \in C$ holds for all $t \in \operatorname{int} I_\psi^J = \operatorname{int}(I_{\psi_1}^J \cup (I_{\psi_2}^0 + \{T\}))$.

Therefore, $\psi(t,j) \in C$ holds for all $t \in \operatorname{int} I_\psi^j$ for all $j \in \mathbb{N}$ such that I_ψ^j has nonempty interior.

(c) This item proves that for all $j \in \mathbb{N}$ such that I_ψ^j has nonempty interior, the function $t \mapsto \upsilon(t,j)$ is Lebesgue measurable and locally bounded.

I Consider the case of all $j < J$ such that $I_\psi^j = I_{\psi_1}^j$ has nonempty interior. Since ψ_1 is solution pairs to $\mathcal{H}$, the functions $t \mapsto \upsilon_1(t,j)$ is Lebesgue measurable and locally bounded for all the $j \in \mathbb{N}$ such that $I_{\psi_1}^j$ has nonempty interior.

Note that $\upsilon(t,j) = \upsilon_1(t,j)$ holds for all $t \in I_\psi^j$ such that $j < J$. Then $t \mapsto \upsilon(t,j) = \upsilon_1(t,j)$ is Lebesgue measurable and locally bounded for all $j < J$ such that $I_\psi^j = I_{\psi_1}^j$ has nonempty interior.

II Consider the case of all $j > J$ such that $I_\psi^j = I_{\psi_2}^{j-J} + \{T\}$ has nonempty interior. Since ψ_2 is a solution pair, then $t \mapsto \upsilon_2(t,j)$ is Lebesgue measurable and locally bounded for $I_{\psi_2}^j$ with nonempty interior.

Note that $\upsilon(t,j) = \upsilon_2(t-T, j-J)$ for all $t \in I_\psi^j$ such that $j > J$. Then $t \mapsto \upsilon(t,j) = \upsilon_2(t-T, j-J)$ is Lebesgue measurable and locally bounded for all $j > J$ such that $I_\psi^j = I_{\psi_2}^{j-J} + \{T\}$ has nonempty interior.

III If $j = J$ and $I_\psi^J = I_{\psi_1}^J \cup (I_{\psi_2}^0 + \{T\})$ has nonempty interior, then according to Definition 3.5,

$$\upsilon(t,j) = \begin{cases} \upsilon_1(t,J) & t \in I_{\psi_1}^J \backslash \{T\} \\ \upsilon_2(t-T,0) & t \in I_{\psi_2}^0 + \{T\}. \end{cases} \tag{A.23}$$

Note that $t \mapsto \upsilon_1(t, J)$ is Lebesgue measurable if $I^J_{\psi_1}$ has nonempty interior and $t \mapsto \upsilon_2(t, 0)$ is Lebesgue measurable if $I^0_{\psi_2}$ has nonempty interior. Therefore, for all $a \in \mathbb{R}$ and $i \in \{1, 2, ..., m\}$, if $I^J_{\psi_1}$ has nonempty interior, set

$$S^i_1 := \{t \in I^J_{\psi_1} : \upsilon^i_1(t, J) > a\}$$

is Lebesgue measurable, where υ^i_1 denotes the ith component of υ_1. For all $a \in \mathbb{R}$ and $i \in \{1, 2, ..., m\}$, if $I^0_{\psi_2}$ has nonempty interior, set

$$S^i_2 := \{t \in I^0_{\psi_2} : \upsilon^i_2(t, 0) > a\}$$

is Lebesgue measurable, where υ^i_2 denotes the ith component of υ_2.

In order to show that $t \mapsto \upsilon(t, J)$ is Lebesgue measurable, we want to show that for all $a \in \mathbb{R}$ and $i \in \{1, 2, ..., m\}$,

$$S^i := \{t \in I^J_{\psi} : \upsilon^i(t, J) > a\}$$

is Lebesgue measurable. Note that for all $a \in \mathbb{R}$ and $i \in \{1, 2, ..., m\}$,

$$S^i = (S^i_1 \backslash \{T\}) \cup (S^i_2 + \{T\}) = (S^i_1 \cap (\mathbb{R} \backslash \{T\})) \cup (S^i_2 + \{T\})$$

and both S^i_1 and S^i_2 are Lebesgue measurable. $(\mathbb{R} \backslash \{T\})$ is Lebesgue measurable because $\{T\}$ is Lebesgue measurable and so is its complement. Then, S^i is Lebesgue measurable. Therefore, $t \mapsto \upsilon(t, J)$ is Lebesgue measurable, if I^J_{ψ} has nonempty interior.

Then we want to show that $t \mapsto \upsilon(t, J)$ is locally bounded. Note that $t \mapsto \upsilon_1(t, j)$ is locally bounded, for all the $j \in \mathbb{N}$ such that $I^j_{\psi_1}$ has nonempty

interior. Therefore, if $I^J_{\psi_1}$ has nonempty interior, for all $i \in \{1, 2, ..., m\}$ and any $t_0 \in I^j_{\psi_1}$, there exists a neighborhood A_1 of t_0 such that for some number $M_1 > 0$, one has

$$|\upsilon^i_1(t, J)| \leq M_1 \tag{A.24}$$

for all $t \in A_1$, where υ^i_1 denotes the ith component of υ_1.

Similarly, if $I^0_{\psi_2}$ has nonempty interior, for all $i \in \{1, 2, ..., m\}$ and any $t_0 \in I^0_{\psi_2}$, there exists a neighborhood A_2 of t_0 such that for some number $M_2 > 0$, one has

$$|\upsilon^i_2(t, 0)| \leq M_2 \tag{A.25}$$

for all $t \in A_2$, where υ^i_2 denotes the ith component of υ_2.

We want to show that if I^J_{ψ} has nonempty interior, for all $i \in \{1, 2, ..., m\}$ and any $t_0 \in I^j_{\psi}$, there exists a neighborhood A of t_0 such that for some number $M > 0$ one has

$$|\upsilon^i(t, J)| \leq M \tag{A.26}$$

for all $t \in A$, where υ^i denotes the ith component of u.

When I^J_{ψ} has nonempty interior, then $I^J_{\psi} = I^J_{\psi_1} \cup (I^0_{\psi_2} + \{T\})$ has nonempty interior. If $I^J_{\psi_1}$ has nonempty interior, then, for $i \in \{1, 2, ..., m\}$ and any $t_0 \in I^J_{\psi_1} \backslash \{T\}$, there exists a neighborhood A_1 of t_0 such that for some number $M_1 > 0$ one has

$$|\upsilon^i_1(t, J)| = \upsilon^i(t, J) \leq M_1 \tag{A.27}$$

for all $t \in A_1$, where υ_1^i denotes the ith component of υ_1.

If $(I_{\psi_2}^0 + \{T\})$ has nonempty interior, then for $i \in \{1, 2, ..., m\}$ and any $t_0 \in I_{\psi_2}^0 + \{T\}$, there exists a neighborhood A_2 of t_0 such that for some number $M_2 > 0$ one has

$$|\upsilon_2^i(t - T, 0)| = \upsilon^i(t, J) \leq M_2 \tag{A.28}$$

for all $t \in A_2$, where υ_2^i denotes the ith component of υ_2.

In conclusion, when I_ψ^J has nonempty interior, for any $t_0 \in I_{\psi_1}^J \backslash \{T\}$, there exists $A = A_1$ such that for the number $M = M_1$, (A.26) holds for all $t \in A$; when $(I_{\psi_2}^0 + \{T\})$ has nonempty interior, for any $t_0 \in I_{\psi_2}^0 + \{T\}$, there exists $A = A_2$ such that for the number $M = M_2$, (A.26) holds for all $t \in A$.

Therefore, $t \mapsto \upsilon(t, j)$ is Lebesgue measurable and locally bounded for $j = J$ such that I_ψ^J has nonempty interior.

Therefore, $t \mapsto \upsilon(t, j)$ is Lebesgue measurable and locally bounded for all $j \in \mathbb{N}$ such that I_ψ^j has nonempty interior.

(d) This item proves that for all $j \in \mathbb{N}$ such that I_ψ^j has nonempty interior, for almost all $t \in I_\psi^j$,

$$\dot{\phi} = f(\phi(t, j), \upsilon(t, j)). \tag{A.29}$$

I Consider the case of all $j < J$ such that $I_{\psi_1}^j$ has nonempty interior. Since ψ_1 is a solution pair, then $\dot{\phi}_1(t, j) = f(\phi_1(t, j), \upsilon_1(t, j))$ for almost all

$t \in \operatorname{int} I^j_{\psi_1}$. Because

$$(\phi(t,j), \upsilon(t,j)) = (\phi_1(t,j), \upsilon_1(t,j))$$

for all $(t,j) \in \operatorname{dom} \psi_1 \backslash (T,J)$, therefore,

$$\dot{\phi}(t,j) = \dot{\phi}_1(t,j) = f(\phi_1(t,j), \upsilon_1(t,j)) = f(\phi(t,j), \upsilon(t,j)) \tag{A.30}$$

for almost all $t \in \operatorname{int} I^j_{\psi}$.

II Consider the case of all $j > J$ such that $I^{j-J}_{\psi_2}$ has nonempty interior. Since ψ_2 is a solution pair and $I^{j-J}_{\psi_2}$ has nonempty interior, $\dot{\phi}_2(t,j) = f(\phi_2(t-T, j-J), \upsilon_2(t-T, j-J))$ for almost all $t \in \operatorname{int} I^{j-J}_{\psi_2} + \{T\}$. Because

$$(\phi(t,j), \upsilon(t,j)) = (\phi_2(t-T, j-J), \upsilon_2(t-T, j-J))$$

for all $t \in I^{j-J}_{\psi_2} + \{T\}$, then

$$\begin{aligned} \dot{\phi}(t,j) &= \dot{\phi}_2(t-T, j-J) \\ &= f(\phi_2(t-T, j-J), \upsilon_2(t-T, j-J)) \\ &= f(\phi(t,j), \upsilon(t,j)) \end{aligned} \tag{A.31}$$

for almost all $t \in \operatorname{int} I^j_{\phi}$.

III If $j = J$ and $I^J_{\psi} = I^J_{\psi_1} \cup (I^0_{\psi_2} + \{T\})$ has nonempty interior, there are three cases to consider.

i. If $I^J_{\psi_1}$ has nonempty interior and $I^0_{\psi_2} + \{T\}$ has empty interior, then the interval $I^J_{\psi} = I^J_{\psi_1}$. Since $\dot{\phi}_1(t,j) = f(\phi_1(t,J), \upsilon_1(t,J))$ for almost

all $t \in \operatorname{int} I^J_{\psi_1}$ and $\psi(t,J) = \psi_1(t,J)$ for all $t \in I^J_{\psi_1} \backslash \{T\}$, then

$$\dot{\phi}(t,J) = \dot{\phi}_1(t,J) = f(\phi_1(t,J), \upsilon_1(t,J)) = f(\phi(t,J), \upsilon(t,j)) \quad \text{(A.32)}$$

for almost all $t \in \operatorname{int} I^j_\psi$.

ii. If $I^J_{\psi_1}$ has empty interior and $(I^0_{\psi_2} + \{T\})$ has nonempty interior, then the interval $I^J_\psi = (I^0_{\psi_2} + \{T\})$. Since $\dot{\phi}_2 = f(\phi_2(t-T,0), \upsilon_2(t-T,0))$ for almost all $t \in \operatorname{int} I^0_{\psi_2} + \{T\}$ and $\psi(t,J) = \psi_2(t-T,0)$ for all $t \in I^0_{\psi_2} + \{T\}$, then

$$\begin{aligned} \dot{\phi}(t,J) &= \dot{\phi}_2(t-T,0) \\ &= f(\phi_2(t-T,0), \upsilon_2(t-T,0)) \\ &= f(\phi(t,J), \upsilon(t,j)) \end{aligned} \quad \text{(A.33)}$$

for almost all $t \in \operatorname{int} I^J_\phi$.

iii. If both $I^J_{\psi_1}$ and $(I^0_{\psi_2} + \{T\})$ has nonempty interior, then the interval $I^J_\psi = I^J_{\psi_1} \cup (I^0_{\psi_2} + \{T\})$. Since we have had $\dot{\phi}(t,J) = f(\phi(t,J), \upsilon(t,j))$ for almost all $t \in I^J_{\psi_1}$ and $t \in I^0_{\psi_2} + \{T\}$, then $\dot{\phi}(t,j) = f(\phi(t,j), \upsilon(t,j))$ holds for $t \in I^J_\psi = I^J_{\psi_1} \cup (I^0_{\psi_2} + \{T\})$.

Therefore, $\dot{\phi}(t,j) = f(\phi(t,j), \upsilon(t,j))$ holds for almost all $t \in I^j_\psi$.

- The last item is to prove that for all $(t,j) \in \operatorname{dom} \psi$ such that $(t, j+1) \in \operatorname{dom} \psi$,

$$\begin{aligned} \psi(t,j) &\in D \\ \phi(t,j+1) &= g(\phi(t,j), \upsilon(t,j)). \end{aligned} \quad \text{(A.34)}$$

1. For all (t, j) such that $(t, j) \in \text{dom } \psi_1$ and $(t, j+1) \in \text{dom } \psi_1$, since ψ_1 is a solution pair, then

$$\begin{aligned} \psi_1(t, j) &\in D \\ \phi_1(t, j+1) &= g(\phi_1(t, j), \upsilon_1(t, j)). \end{aligned} \tag{A.35}$$

Due to $(t, j+1) \in \text{dom } \psi_1$, then $(t, j) \neq (T, J)$. Since $\psi(t, j) = \psi_1(t, j)$ for all $(t, j) \in \text{dom } \psi_1 \backslash (T, J)$, then

$$\begin{aligned} \psi(t, j) &= \psi_1(t, j) \in D \\ \phi(t, j+1) = \phi_1(t, j+1) &= g(\phi_1(t, j), \upsilon_1(t, j)) = g(\phi(t, j), \upsilon(t, j)). \end{aligned} \tag{A.36}$$

2. For all (t, j) such that $(t, j) \in \text{dom } \psi_2 + \{(T, J)\}$ and $(t, j+1) \in \text{dom } \psi_2 + \{(T, J)\}$, we can have $(t-T, j-J) \in \text{dom } \psi_2$ and $(t-T, j-J+1) \in \text{dom } \psi_2$. Since ψ_2 is a solution pair, then

$$\begin{aligned} \psi_2(t-T, j-J) &\in D \\ \phi_2(t-T, j-J+1) &= g(\phi_2(t-T, j-J), \upsilon_2(t-T, j-J)). \end{aligned} \tag{A.37}$$

Since $\psi(t, j) = \psi_2(t-T, j-J)$ for all $(t-T, j-J) \in \text{dom } \psi_2$, then

$$\begin{aligned} \psi(t, j) &= \psi_2(t-T, j-J) \in D \\ \phi(t, j+1) &= \phi_2(t-T, j-J+1) \\ &= g(\phi_2(t-T, j-J), \upsilon_2(t-T, j-J)) \\ &= g(\phi(t, j), \upsilon(t, j)). \end{aligned} \tag{A.38}$$

Because all of the four conditions above are satisfied, the result ψ of concatenation is a solution pair to $\mathcal{H}$.

A.3 Proof of Proposition 3.19

The following items are to prove that $\widetilde{\psi}$ satisfies all conditions in Definition 2.5.

- The first item is to prove that dom $\widetilde{\phi}$ equals dom $\widetilde{\upsilon}$ and dom $\widetilde{\psi}$ = dom $\widetilde{\phi}$ = dom $\widetilde{\upsilon}$ is a hybrid time domain.

  Note that

  $$\text{dom } \widetilde{\upsilon} = \text{dom } \upsilon \cap [T_1, T_2] \times \{J_1, J_1 + 1, ..., J_2\} - \{(T_1, J_1)\} \tag{A.39}$$

  and

  $$\text{dom } \widetilde{\phi} = \text{dom } \phi \cap [T_1, T_2] \times \{J_1, J_1 + 1, ..., J_2\} - \{(T_1, J_1)\}. \tag{A.40}$$

  Since $\psi = (\phi, \upsilon)$ is a solution pair of $\mathcal{H}$, then dom ϕ = dom υ. Thus,

  $$\begin{aligned} \text{dom } \widetilde{\phi} &= \text{dom } \phi \cap [T_1, T_2] \times \{J_1, J_1 + 1, ..., J_2\} - \{(T_1, J_1)\} \\ &= \text{dom } \upsilon \cap [T_1, T_2] \times \{J_1, J_1 + 1, ..., J_2\} - \{(T_1, J_1)\} \\ &= \text{dom } \widetilde{\upsilon}. \end{aligned} \tag{A.41}$$

  Therefore, dom $\widetilde{\phi}$ = dom $\widetilde{\upsilon}$.

  Since ψ is a solution pair of $\mathcal{H}$, dom ψ is a hybrid time domain. Hence, for each $(T, J) \in$ dom ψ, dom $\psi \cap [0, T] \times \{0, 1, ..., , J\}$ can be written as $\cup_{j=0}^{J}([t_j, t_{j+1}], j)$ for some finite sequence of times $0 = t_0 \leq t_1 \leq t_2 \leq ... \leq t_{J+1} = T$. In the truncation operation definition,

  $$\text{dom } \widetilde{\phi} = \text{dom } \phi \cap [T_1, T_2] \times \{J_1, J_1 + 1, ..., J_2\} - \{(T_1, J_1)\}. \tag{A.42}$$

Hence, for each $(T, J) \in \text{dom } \widetilde{\psi}$,

$$\begin{aligned}
&\text{dom } \widetilde{\phi} \cap [0,T] \times \{0,1,...,,J\} \\
&\quad = (\text{dom } \phi \cap [T_1, T_2] \times \{J_1, J_1+1, ..., J_2\} - \{(T_1, J_1)\}) \cap [0,T] \cap \{0,1,...,,J\} \\
&\quad = \cup_{j=0}^{J} ([t_j, t_{j+1}], j) \cap [T_1, T_2] \times \{J_1, J_1+1, ..., J_2\} - \{(T_1, J_1)\} \\
&\quad = \cup_{j=0}^{J_2 - J_1} ([t'_j, t'_{j+1}], j)
\end{aligned} \tag{A.43}$$

where $\{t'_j\}$ is constructed by

$$t'_j = t_{j+J_1} - T_1 \quad \forall j \in \{1, ..., J_2 - J_1\} \tag{A.44}$$

such that

$$0 = t'_0 \leq t'_1 \leq t'_2 \leq ... \leq t'_{J_2 - J_1 + 1} = T_2 - T_1.$$

Hence, dom $\widetilde{\phi}$ is a hybrid time domain. For the same reason, the domain dom $\widetilde{v}$ is a hybrid time domain.

- The second item is to prove that $\widetilde{\psi}(0,0) \in \overline{C} \cup D$.

  According to Definition B.11, the state input pair $\widetilde{\psi}(0,0) = \psi(T_1, J_1)$. In Assumption 3.18, there are three cases to consider.

  1. If $(T_1, J_1) = (0,0)$, since ψ is a solution pair to $\mathcal{H}$, then $\psi(0,0) \in \overline{C} \cup D$. Therefore, $\widetilde{\psi}(0,0) = \psi(T_1, J_1) = \psi(0,0) \in \overline{C} \cup D$.
  2. If $T_1 \in \text{int } I_\psi^{J_1}$, where $I_\psi^{J_1}$ has nonempty interior, Definition 2.5 implies that $\psi(T_1, J_1) \in C \subset \overline{C}$.

3. If $(T_1, J_1) \in \text{dom } \psi$ and $(T_1, J_1 + 1) \in \text{dom } \psi$, Definition 2.5 implies that $\psi(T_1, J_1) \in D$.

Therefore, $\widetilde{\psi}(0,0) = \psi(T_1, J_1) \in \overline{C} \cup D$.

- The third item is to prove that $\widetilde{\psi}$ satisfies the conditions in the first item in Definition 2.5 for all $j \in \mathbb{N}$ such that $I^j_{\widetilde{\psi}} = \{t : (t,j) \in \text{dom } \widetilde{\psi}\}$ has nonempty interior. The following items is to prove that each of these conditions is satisfied.

  (a) This item is to prove that for all $j \in \mathbb{N}$ such that $I^j_{\widetilde{\psi}} = \{t : (t,j) \in \text{dom } \widetilde{\psi}\}$ has nonempty interior, the function $t \mapsto \widetilde{\phi}(t,j)$ is locally absolutely continuous.

  Since the domain of $\widetilde{\psi}$ is constructed by

  $$\text{dom } \widetilde{\psi} = \text{dom } \psi \cap [T_1, T_2] \times \{J_1, J_1 + 1, ..., J_2\} - \{(T_1, J_1)\}, \tag{A.45}$$

  then $(t,j) \in \text{dom } \widetilde{\psi}$ implies $(t + T_1, j + J_1) \in \text{dom } \psi \cap [T_1, T_2] \times \{J_1, J_1 + 1, ..., J_2\}$. Therefore, $t \in I^j_{\widetilde{\psi}}$ implies $t + T_1 \in I^{j+J_1}_{\psi}$. Thus,

  $$I^j_{\widetilde{\psi}} = (I^{j+J_1}_{\psi} - \{T_1\}) \cap [0, T_2 - T_1] \tag{A.46}$$

  holds. When the interval $I^j_{\widetilde{\psi}}$ has nonempty interior, then interval $I^{j+J_1}_{\psi}$ has nonempty interior.

  Since ψ is a solution pair to $\mathcal{H}$, according to Definition 2.5, $t \mapsto \phi(t,j)$ is locally absolutely continuous for all $j \in \mathbb{N}$ such that I^j_{ψ} has nonempty interior. Thus, the function $t \mapsto \phi(t + T_1, j + J_1)$ is locally absolutely continuous for all j such that $I^j_{\widetilde{\psi}}$ has nonempty interior.

According to Definition B.11,

$$\widetilde{\phi}(t,j) = \phi(t+T_1, j+J_1)$$

holds for all $(t,j) \in \operatorname{dom} \widetilde{\phi}$. Therefore, the function $t \mapsto \widetilde{\phi}(t,j) = \phi(t+T_1, j+J_1)$ is locally absolutely continuous for all j such that $I^j_{\widetilde{\psi}} = I^{j+J_1}_{\psi} - \{T_1\}$ has nonempty interior.

(b) This item is to prove that for all $j \in \mathbb{N}$ such that $I^j_{\widetilde{\psi}} = \{t : (t,j) \in \operatorname{dom} \widetilde{\psi}\}$ has nonempty interior, $(\widetilde{\phi}(t,j), \widetilde{v}(t,j)) \in C$ for all $t \in \operatorname{int} I^j_{\widetilde{\psi}}$.

Considering (A.46), the interval $I^j_{\widetilde{\psi}}$ having nonempty interior implies that $I^{j+J_1}_{\psi} - \{T_1\}$ has nonempty interior. Because ψ is a solution pair to $\mathcal{H}$ and $I^{j+J_1}_{\psi} - \{T_1\}$ has nonempty interior, the state-input pair $\psi(t+T_1, j+J_1) \in C$ for all $t \in \operatorname{int} I^{j+J_1}_{\psi} - \{T_1\}$. Due to $\psi(t+T_1, j+J_1) = \widetilde{\psi}(t,j)$, then $\widetilde{\psi}(t,j) \in C$ for all $t \in \operatorname{int} I^j_{\widetilde{\psi}}$.

(c) This item is to prove that for all $j \in \mathbb{N}$ such that $I^j_{\widetilde{\psi}} = \{t : (t,j) \in \operatorname{dom} \widetilde{\psi}\}$ has nonempty interior, the function $t \mapsto \widetilde{v}(t,j)$ is Lebesgue measurable and locally bounded.

– This item is to show that if $I^j_{\widetilde{\psi}}$ has nonempty interior, the function $t \mapsto \widetilde{v}(t,j)$ is Lebesgue measurable.

Note that if $I^j_{\widetilde{\psi}}$ has nonempty interior, then $I^{j+J_1}_{\psi}$ has nonempty interior. Then $t \mapsto v(t, j+J_1)$ is Lebesgue measurable, which implies that for all $a \in \mathbb{R}$ and $i \in \{1, 2, ..., m\}$,

$$S_i := \{t \in I^{j+J_1}_{\psi} : v_i(t, j+J_1) > a\}$$

is Lebesgue measurable, where υ_i denotes the ith component of u. Hence,

$$S_i \cap [T_1, T_2] - \{T_1\} = \{t \in I^j_{\widetilde{\psi}} : \widetilde{\upsilon}_i(t, j) > a\} =: \widetilde{S}_i$$

is Lebesgue measurable, where $\widetilde{\upsilon}_i$ denotes the ith component of $\widetilde{\upsilon}$. Therefore, for all the $j \in \mathbb{N}$ such that $I^j_{\widetilde{\psi}}$ has nonempty interior, $a \in \mathbb{R}$ and $i \in \{1, 2, ..., m\}$, $\widetilde{S}_i$ is Lebesgue measurable, and hence, $t \mapsto \tilde{u}(t, j)$ is Lebesgue measurable.

– This item is to show that for all $j \in \mathbb{N}$ such that $I^j_{\widetilde{\psi}}$ has nonempty interior, the function $t \mapsto \widetilde{\upsilon}(t, j)$ is locally bounded. In other words, we want to show that for all $j \in \mathbb{N}$ such that $I^j_{\widetilde{\psi}}$ has nonempty interior, $i \in \{1, 2, ..., m\}$ and any $t_0 \in I^j_{\widetilde{\psi}}$, there exists a neighborhood $\widetilde{A}$ of t_0 such that for some number $\widetilde{M} > 0$, one has

$$|\widetilde{\upsilon}_i(t, j)| \leq \widetilde{M} \tag{A.47}$$

for all $t \in \widetilde{A}$, where $\widetilde{\upsilon}_i$ denotes the ith component of $\widetilde{\upsilon}$.

Note that $t \mapsto \upsilon(t, j)$ is locally bounded. Hence, for all $j \in \mathbb{N}$ such that I^j_{ψ} has nonempty interior, $i \in \{1, 2, ..., m\}$ and any $t_0 \in I^j_{\psi}$, there exists a neighborhood A of t_0 such that for some number $M > 0$, one has

$$|\upsilon_i(t, j)| \leq M \tag{A.48}$$

for all $t \in A$, where υ_i denotes the ith component of υ.

Note that $\text{dom } \widetilde{\psi} = \text{dom } \psi \cap [T_1, T_2] \times \{J_1, J_1 + 1, ..., J_2\} - (T_1, J_1)$. Therefore, for all the $j \in \mathbb{N}$ such that $I^j_{\widetilde{\psi}}$ has nonempty interior, $I^j_{\widetilde{\psi}} =$

$I_{\psi}^{j+J_1} \cap [T_1, T_2] - \{T_1\}$. Since $I_{\widetilde{\psi}}^{j}$ has nonempty interior, then $I_{\psi}^{j+J_1}$ has nonempty interior.

Therefore, for $i \in \{1, 2, ..., m\}$ and any $t_0 \in I_{\psi}^{j+J_1} \cap [T_1, T_2] - \{T_1\} = I_{\widetilde{\psi}}^{j}$, there exists a neighborhood $\widetilde{A} = A \cap [T_1, T_2] - T_1$ of t_0 such that for some number $\widetilde{M} = M > 0$, one has

$$|\widetilde{\upsilon}_i(t, j)| = |\upsilon_i(t + T_1, j + J_1)| \leq M \tag{A.49}$$

for all $t \in \widetilde{A}$.

Hence, the function $t \mapsto \widetilde{\upsilon}(t, j)$ is locally bounded for all $j \in \mathbb{N}$ such that $I_{\widetilde{\psi}}^{j}$ has nonempty interior.

Therefore, the function $t \mapsto \widetilde{\upsilon}(t, j)$ is Lebesgue measurable and locally bounded for all $j \in \mathbb{N}$ such that $I_{\widetilde{\psi}}^{j}$ has nonempty interior.

(d) This item is to prove that for all $j \in \mathbb{N}$ such that $I_{\widetilde{\psi}}^{j}$ has nonempty interior, for almost all $t \in I_{\widetilde{\psi}}^{j}$

$$\dot{\widetilde{\phi}}(t, j) = f(\widetilde{\phi}(t, j), \widetilde{\upsilon}(t, j)). \tag{A.50}$$

Since ψ is a solution pair of $\mathcal{H}$, Definition 2.5 implies that for all $j \in \mathbb{N}$ such that I_{ψ}^{j} has nonempty interior, for almost all $t \in I_{\psi}^{j}$,

$$\dot{\phi}(t, j) = f(\phi(t, j), \upsilon(t, j)). \tag{A.51}$$

Thus, for all $j \in \mathbb{N}$ such that $I_{\psi}^{j+J_1} - \{T_1\}$ has nonempty interior, for almost all $t \in I_{\psi}^{j+J_1} - \{T_1\}$,

$$\dot{\phi}(t + T_1, j + J_1) = f(\phi(t + T_1, j + J_1), \upsilon(t + T_1, j + J_1)) \tag{A.52}$$

holds.

According to Definition B.11,

$$\widetilde{\phi}(t,j) = \phi(t + T_1, j + J_1)$$

and

$$\widetilde{v}(t,j) = v(t + T_1, j + J_1)$$

hold. Therefore,

$$\begin{aligned} \dot{\widetilde{\phi}}(t,j) &= \dot{\phi}(t + T_1, j + J_1) \\ &= f(\phi(t + T_1, j + J_1), v(t + T_1, j + J_1)) \\ &= f(\widetilde{\phi}(t,j), \widetilde{v}(t,j)) \end{aligned} \tag{A.53}$$

holds for almost all $t \in I^j_{\widetilde{\psi}} = I^{j+J_1}_{\psi} - \{T_1\}$ where $I^j_{\widetilde{\psi}}$ has nonempty interior.

- The last item is to prove that for all $(t,j) \in \text{dom } \widetilde{\psi}$ such that $(t, j+1) \in \text{dom } \widetilde{\psi}$,

$$\begin{aligned} &(\widetilde{\phi}(t,j), \widetilde{v}(t,j)) \in D \\ &\widetilde{\phi}(t, j+1) = g(\widetilde{\phi}(t,j), \widetilde{v}(t,j)). \end{aligned} \tag{A.54}$$

According to Definition B.11,

$$(t,j) \in \text{dom } \widetilde{\psi} \quad (t, j+1) \in \text{dom } \widetilde{\psi}$$

implies that

$$(t + T_1, j + J_1) \in \text{dom } \psi \quad (t + T_1, j + J_1 + 1) \in \text{dom } \psi.$$

Therefore, according to Definition 2.5,

$$(\phi(t+T_1, j+J_1), \upsilon(t+T_1, j+J_1)) \in D$$
$$\phi(t+T_1, j+J_1+1) = g(\phi(t+T_1, j+J_1), \upsilon(t+T_1, j+J_1)). \tag{A.55}$$

Due to Definition B.11,

$$\widetilde{\phi}(t, j) = \phi(t+T_1, j+J_1)$$

and

$$\widetilde{\upsilon}(t, j) = \upsilon(t+T_1, j+J_1)$$

hold. Therefore,

$$\begin{aligned} (\widetilde{\phi}(t, j), \widetilde{\upsilon}(t, j)) &= (\phi(t+T_1, j+J_1), \upsilon(t+T_1, j+J_1)) \in D \\ \widetilde{\phi}(t, j+1) &= \phi(t+T_1, j+J_1+1) \\ &= g(\phi(t+T_1, j+J_1), \upsilon(t+T_1, j+J_1)) \\ &= g(\widetilde{\phi}(t, j), \widetilde{\upsilon}(t, j)) \end{aligned} \tag{A.56}$$

hold.

Since all the conditions in Definition 2.5 are satisfied, $\widetilde{\psi} = (\widetilde{\psi}, \widetilde{\upsilon})$ is a solution pair to $\mathcal{H}$.

Appendix B

Proof for Results in Chapter 4

B.1 A Computational Framework to Simulate Continuous Dynamics

B.1.1 Numerical integration scheme model

A general numerical integration scheme can be modeled as

$$F_s(x_k, x_{k+1}, \tilde{v}, f, t) = 0 \tag{B.1}$$

where s denotes the step size, x_k denotes the state at the kth step, x_{k+1} denotes the approximation of the state at the next step, $\tilde{v}$ denotes the applied input signal, f denotes the flow map, and t denotes the time at current step, namely, $t = ks$. Two examples on how (B.1) models explicit and implicit numerical integration scheme are given as follows.

Example B.1 (Forward Euler method)**.** *Given the differential equation* $\dot{x} = f(x, \tilde{v})$

with initial state x_0 and input signal $\tilde{v} \in \mathcal{U}_C$, the integration scheme using the forward Euler method is

$$x_{k+1} = x_k + sf(x, \tilde{v}(t)). \tag{B.2}$$

Following (B.1), (B.2) can be modeled as

$$F_s(x_k, x_{k+1}, \tilde{v}, f, t) := x_k + sf(x_k, \tilde{v}(t)) - x_{k+1} \tag{B.3}$$

and x_{k+1} can be obtained by solving (B.1).

Example B.2 (Backward Euler method)**.** *Given the differential equation $\dot{x} = f(x, \tilde{v})$ with initial state x_0 and input signal $\tilde{v} \in \mathcal{U}_C$, the integration scheme using backward Euler method is*

$$x_{k+1} = x_k + sf(x_{k+1}, \tilde{v}(t+s)). \tag{B.4}$$

Following (B.1), (B.4) can be modeled as

$$F_s(x_k, x_{k+1}, \tilde{v}, f, t) := x_k + sf(x_{k+1}, \tilde{v}(t+s)) - x_{k+1}$$

and x_{k+1} can be obtained by solving (B.1).

B.1.2 Zero-crossing detection model to approximate $\hat{t}$

When the priority option $rule = 1$, following Step 2 above, $\hat{t}$ in (4.5) is the first time when $(\hat{\phi}, \tilde{v})$ leaves C. When the priority option $rule = 2$, $\hat{t}$ is the first time when $(\hat{\phi}, \tilde{v})$ leaves $C \backslash D$.

Given a set $S \subset \mathbb{R}^n \times \mathbb{R}^m$, a zero-crossing function $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ to detect whether the pair (ϕ, v) is in set S is given by

1. $h(x,u) > 0$ for all $(x,u) \in \operatorname{int} S$,

2. $h(x,u) < 0$ for all $(x,u) \in \operatorname{int}((\mathbb{R}^n \times \mathbb{R}^m) \backslash S)$,

3. $h(x,u) = 0$ for all $(x,u) \in \partial S$,

4. h is continuous over $\mathbb{R}^n \times \mathbb{R}^m$.

When $rule = 1$, this function can be used with $S = C \backslash D$ to determine $\hat{t}$ as the first time when $(\hat{\phi}, \tilde{\upsilon})$ leaves $C \backslash D$ by checking zero crossings of $h_f(\hat{\phi}, \tilde{\upsilon})$. When $rule = 2$, this function can be used with $S = C$ to determine $\hat{t}$ as the first time when $(\hat{\phi}, \tilde{\upsilon})$ leaves C by checking zero crossings of $h(\hat{\phi}, \tilde{\upsilon})$. Next, we illustrate this approach by constructing a zero-crossing function for the bouncing ball system in Example 3.2.

Example B.3 (Example 3.2, revisited)**.** *In the bouncing ball system in Example 3.2, the flow set is* $C = \{(x,u) \in \mathbb{R}^2 \times \mathbb{R} : x_1 \geq 0\}$ *and the jump set is* $D = \{(x,u) \in \mathbb{R}^2 \times \mathbb{R} : x_1 = 0, x_2 \leq 0, u \geq 0\}$*. Then,*

$$h(x,u) := x_1 \quad \forall (x,u) \in \mathbb{R}^n \times \mathbb{R}^m.$$

is a zero-crossing function for the detection of solutions leaving C*. The same function can be used to detect solutions leaving* $C \backslash D$ *since* $D \subset \partial C$ *which implies that* $\operatorname{int}(C \backslash D) = \operatorname{int} C$ *and* $\operatorname{int}((\mathbb{R}^n \times \mathbb{R}^m) \backslash (C \backslash D)) = \operatorname{int}((\mathbb{R}^n \times \mathbb{R}^m) \backslash C)$.

The zero-crossing detection algorithm that approximates $\hat{t}$ can be modeled as

$$\hat{t} = t_{zcd}(\hat{\phi}, \tilde{\upsilon}, rule, C, D) \tag{B.5}$$

where the function t_{zcd} employs a zero-crossing detection function for $C \backslash D$ or C, depending on the value of *rule*. We set $\hat{t}$ as -1 when no zero-crossing is detected.

B.1.3 A computational framework to simulate continuous dynamics

A computational approach to simulate the continuous dynamics of $\mathcal{H}$ is given in Algorithm 9.

Algorithm 9 A computational framework to simulate the continuous dynamics

`continuous_simulator`

function CONTINUOUS_SIMULATOR$(C, f, D, rule, s, x_0, \tilde{\upsilon})$

  Set $\phi_0 \leftarrow x_0$, $\hat{t} \leftarrow 0$.

  while $t \leq \bar{t}(\tilde{\upsilon}) \& \hat{t} = -1$ **do**

    Compute $\hat{\phi}$ by solving $F_s(\phi_0, \hat{\phi}, \tilde{\upsilon}, f, t) = 0$.

    $\hat{t} \leftarrow t_{zcd}(\hat{\phi}, \tilde{\upsilon}, rule, C, D)$.

    if $\hat{t} = -1$ **then**

      $\phi \leftarrow \phi|\hat{\phi}$, $\upsilon \leftarrow \upsilon|\hat{\upsilon}$.

      $\phi_0 \leftarrow \hat{\phi}(\hat{T}, 0)$ where $(\hat{T}, 0) = \max \text{dom } \hat{\phi}$.

    else

      $\phi \leftarrow \phi|\hat{\phi}([0, \hat{t}], 0)$, $\upsilon \leftarrow \upsilon|\hat{\upsilon}([0, \hat{t}], 0)$.

      $\phi_0 \leftarrow \hat{\phi}(\hat{t}, 0)$.

    end if

  end while

  return (ϕ, υ).

end function

B.2 Proof of Proposition 4.14

B.2.1 Theoretical Tools to Prove Proposition 4.14

We show that $\psi = (\phi, \upsilon)$ in Proposition 4.14 satisfies the conditions in Definition 2.5 for $(C_\delta, f_\delta, D_\delta, g_\delta)$ where $(C_\delta, f_\delta, D_\delta, g_\delta)$ is the δ-inflation of hybrid dynamical system of (C, f, D, g) for each $\delta > 0$, as defined in Definition 4.12.

Lemma B.4. *Given a hybrid dynamical system $\mathcal{H} = (C, f, D, g)$ and its δ-inflation $\mathcal{H}_\delta = (C_\delta, f_\delta, D_\delta, g_\delta)$ for each $\delta > 0$, if $\psi = (\phi, \upsilon)$ is a solution pair to $\mathcal{H}$, then ψ is also a solution pair to $\mathcal{H}_\delta$.*

Proof. We show that ψ satisfies all the conditions in Definition 2.5 for $(C_\delta, f_\delta, D_\delta, g_\delta)$, namely,

1. ϕ is a hybrid arc and υ is a hybrid input;
2. $(\phi(0,0), \upsilon(0,0)) \in \overline{C_\delta} \cup D_\delta$;
3. For each $j \in \mathbb{N}$ such that I_ϕ^j has nonempty interior $\operatorname{int}(I_\phi^j)$, (ϕ, υ) satisfies

$$(\phi(t,j), \upsilon(t,j)) \in C_\delta$$

for all $t \in \operatorname{int} I_\phi^j$, and

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi(t,j) = f_\delta(\phi(t,j), \upsilon(t,j))$$

for almost all $t \in I_\phi^j$.

4. For all $(t, j) \in \text{dom } (\phi, \upsilon)$ such that $(t, j+1) \in \text{dom } (\phi, \upsilon)$,

$$\begin{aligned} (\phi(t,j), \upsilon(t,j)) &\in D_\delta \\ \phi(t, j+1) &= g_\delta(\phi(t,j), \upsilon(t,j)). \end{aligned} \tag{B.6}$$

Next, we show that each item above is satisfied.

1. Given that $\psi = (\phi, \upsilon)$ is a solution pair to $\mathcal{H}$, therefore, ϕ is a hybrid arc and υ is a hybrid input.

2. Since $\psi = (\phi, \upsilon)$ is a solution pair to $\mathcal{H}$, then $(\phi(0,0), \upsilon(0,0)) \in \overline{C} \cup D$. Note that $C \subset C_\delta$, $D \subset D_\delta$. Therefore,

$$(\phi(0,0), \upsilon(0,0)) \in \overline{C} \cup D \subset \overline{C_\delta} \cup D_\delta.$$

3. For all $j \in \mathbb{N}$ such that $I^j = \{t : (t, j) \in \text{dom } (\phi, \upsilon)\}$ has nonempty interior,

   (a) $(\phi(t,j), \upsilon(t,j)) \in C \subset C_\delta$ for all $t \in \text{int } I^j$.

   (b) for almost all $t \in I^j$,

$$\dot{\phi}(t,j) = f(\phi(t,j), \upsilon(t,j)) = f_\delta(\phi(t,j), \upsilon(t,j)).$$

4. For all $(t, j) \in \text{dom } (\phi, \upsilon)$ such that $(t, j+1) \in \text{dom } (\phi, \upsilon)$,

$$\begin{aligned} (\phi(t,j), \upsilon(t,j)) &\in D \subset D_\delta \\ \phi(t, j+1) &= g(\phi(t,j), \upsilon(t,j)) = g_\delta(\phi(t,j), \upsilon(t,j)). \end{aligned}$$

Since ψ satisfies all the items in Definition 2.5 for data $(C_\delta, f_\delta, D_\delta, g_\delta)$, it is established that ψ is a solution pair to $(C_\delta, f_\delta, D_\delta, g_\delta)$. □

B.2.2 Proof of Proposition 4.14

We show that the motion plan $\psi = (\phi, \upsilon)$ to $\mathcal{P}$ satisfies each condition in Problem 3.1 for the data $\mathcal{P}_\delta = (X_0, X_f, X_u, (C_\delta, f_\delta, D_\delta, g_\delta))$, namely,

1. $\phi(0,0) \in X_0$;
2. (ϕ, υ) is a solution pair to $\mathcal{H}_\delta$;
3. (T, J) is such that $\phi(T, J) \in X_f$, namely, the solution belongs to the final state set at hybrid time (T, J);
4. $(\phi(t,j), \upsilon(t,j)) \notin X_u$ for each $(t,j) \in \text{dom}\,(\phi, \upsilon)$ such that $t + j \leq T + J$.

Given that the initial state set X_0, the final state set X_f, and the unsafe set X_u are the same in both $\mathcal{P}$ and $\mathcal{P}_\delta$ and that ψ is a motion plan to $\mathcal{P}$, items 1, 3, and 4 are satisfied for free. Note that, by construction, every solution pair to $\mathcal{H}$ is a solution pair to $\mathcal{H}_\delta$. In fact, by Lemma B.4, (ϕ, υ) is a solution pair to $\mathcal{H}_\delta$, namely, item 2 is satisfied. Therefore, all the items are satisfied and (ϕ, υ) is a motion plan to $\mathcal{P}_\delta$.

B.3 Proof of Lemma 4.15

To prove Lemma 4.15, we first establish the following result.

Lemma B.5. *Given a hybrid dynamical system defined as $\mathcal{H} = (C, f, D, g)$ and its δ_f-inflation, denoted $\mathcal{H}_{\delta_f} = (C_{\delta_f}, f_{\delta_f}, D_{\delta_f}, g_{\delta_f})$ and defined in (4.12), if a state input pair $(y, v) \in \mathbb{R}^n \times \mathbb{R}^m$ is such that $(y, v) \in X$, where X can be either C or D, then for each $\delta' \in [0, \delta_f]$, we have $(y + \delta'\mathbb{B}, v + \delta'\mathbb{B}) \subset X_{\delta_f}$.*

Proof. Note that, from (4.12), C_{δ_f} and D_{δ_f} are captured by the following set by choosing $X = C$ or $X = D$, respectively:

$$X_{\delta_f} := \{(x, u) \in \mathbb{R}^n \times \mathbb{R}^m : \exists (y, v) \in X : x \in y + \delta_f \mathbb{B}, u \in v + \delta_f \mathbb{B}\}. \tag{B.7}$$

Then, to prove $(y + \delta' \mathbb{B}, v + \delta' \mathbb{B}) \subset X_{\delta_f}$ where $(y, v) \in X$, we show that for each $(y', v') \in (y + \delta' \mathbb{B}, v + \delta' \mathbb{B})$, we have $(y', v') \in X_{\delta_f}$.

Given that $(y', v') \in (y + \delta' \mathbb{B}, v + \delta' \mathbb{B})$ where $(y, v) \in X$, by (B.7), it follows that

$$(y', v') \in X_{\delta'}.$$

Since $\delta' \in [0, \delta_f]$, it follows that $y + \delta' \mathbb{B} \subset y + \delta_f \mathbb{B}$ and $v + \delta' \mathbb{B} \subset v + \delta_f \mathbb{B}$. Then, by (B.7), we have that

$$X_{\delta'} \subset X_{\delta_f}.$$

Therefore, for each $(y', v') \in (y + \delta' \mathbb{B}, v + \delta' \mathbb{B})$ and each $\delta' \in [0, \delta_f]$ where $(y, v) \in X$, we have

$$(y', v') \in X_{\delta'} \subset X_{\delta_f}.$$

Hence, we have

$$(y + \delta' \mathbb{B}, v + \delta' \mathbb{B}) \subset X_{\delta_f}.$$

□

To establish Lemma 4.15, we proceed as follows. Proposition 4.14 establishes that ψ, which is a motion to problem $\mathcal{P}$, is also a motion plan to problem $\mathcal{P}_{\delta_f}$. We need to show that ψ has clearance $\min\{\delta_s, \delta_f\}$.

Since $\psi = (\phi, \upsilon)$ is a solution pair to $\mathcal{H} = (C, f, D, g)$, then for all $j \in \mathbb{N}$ such that $I^j = \{t : (t, j) \in \text{dom } (\phi, \upsilon)\}$ has nonempty interior, we have

$$(\phi(t, j), \upsilon(t, j)) \in C$$

for all $t \in \text{int } I^j$. Therefore, by Lemma B.5, for each $\delta' \in [0, \delta_f]$ and each $j \in \mathbb{N}$ such that I^j has nonempty interior, it follows that

$$(\phi(t, j) + \delta'\mathbb{B}, \upsilon(t, j) + \delta'\mathbb{B}) \subset C_{\delta_f}.$$

Hence, item 1 in Definition 4.9 is satisfied.

Similarly, since ψ is a solution pair to $\mathcal{H} = (C, f, D, g)$, for each $(t, j) \in \text{dom } \psi$ such that $(t, j + 1) \in \text{dom } \psi$, we have

$$(\phi(t, j), \upsilon(t, j)) \in D.$$

Therefore, by Lemma B.5, for each $\delta' \in [0, \delta_f]$ and each $(t, j) \in \text{dom } \psi$ such that $(t, j + 1) \in \text{dom } \psi$, it follows that

$$(\phi(t, j) + \delta_f\mathbb{B}, \upsilon(t, j) + \delta_f\mathbb{B}) \subset D_{\delta_f}.$$

Hence, item 2 in Definition 4.9 is also satisfied.

Therefore, δ_f satisfies all the conditions in Definition 4.9 and, hence, is the dynamics clearance. Then, by Definition 4.11, the clearance of ψ is $\min\{\delta_s, \delta_f\}$.

B.4 Proof of Lemma 4.31

The proof closely parallels that of Lemma 4 as presented in [27]. Suppose there exists a vertex, denoted z, in the search graph $\mathcal{T} = (V, E)$ such that $\overline{x}_z \in S$ and

$\overline{x}_z \notin x_c + \delta\mathbb{B}$, as otherwise it is immediate that $\overline{x}_{v_{cur}} \in x_c + \delta\mathbb{B}$ because all the vertices in $\mathcal{T}$ belong to $x_c + \delta\mathbb{B}$. We show that, under the conditions in Lemma 4.31, if the sampling point x_{rand} returned by the function call `random_state` is such that $x_{rand} \in x_c + \delta/5\mathbb{B}$, it follows that $\overline{x}_{v_{cur}} \in x_c + \delta\mathbb{B}$.

Given that $x_{rand} \in x_c + \delta/5\mathbb{B}$, it follows that

$$|x_{rand} - x_c| \leq \frac{\delta}{5}. \tag{B.8}$$

Since $\overline{x}_v \in x_c + 2\delta/5\mathbb{B}$, then we have

$$|\overline{x}_v - x_c| \leq \frac{2\delta}{5}.$$

Therefore, by the triangle inequality, it follows that

$$|x_{rand} - \overline{x}_v| \leq |x_{rand} - x_c| + |x_c - \overline{x}_v| \leq \frac{3\delta}{5}.$$

Since $\overline{x}_z \notin x_c + \delta\mathbb{B}$, then $|\overline{x}_z - x_c| > \delta$. Using (B.8), by the triangle inequality, it follows that

$$\delta < |\overline{x}_z - x_c| \leq |\overline{x}_z - x_{rand}| + |x_{rand} - x_c| \leq |\overline{x}_z - x_{rand}| + \frac{\delta}{5},$$

namely, $|\overline{x}_z - x_{rand}| > 4\delta/5$.

Since $|\overline{x}_v - x_{rand}| \leq 3\delta/5$ and $|\overline{x}_z - x_{rand}| > 4\delta/5$, we have that x_{rand} is closer to v than to z. This implies that z, which can be any vertex that is not in $x_c + \delta\mathbb{B}$, will not be reported as v_{cur} by the function call `nearest_neighbor`.

If v is reported as v_{cur}, then, since $\overline{x}_v \in x_c + 2\delta/5$, we have that $\overline{x}_v \in x_c + 2\delta/5 \subset x_c + \delta\mathbb{B}$. If v is not reported as v_{cur}, then there must exists another vertex $y \in V$ such

that y is either closer to or equidistant from x_{rand} as compared to v, i.e.,

$$|\overline{x}_y - x_{rand}| \leq |\overline{x}_v - x_{rand}| \leq \frac{3\delta}{5}.$$

Then, $|\overline{x}_y - x_c| \leq |\overline{x}_y - x_{rand}| + |x_{rand} - x_c| \leq 4\delta/5$, which implies that $\overline{x}_y \in x_c + \delta\mathbb{B}$. This implies that if the sampling point x_{rand} is such that $x_{rand} \in x_c + \delta/5\mathbb{B}$, no matter v or y is reported as $x_{v_{cur}}$, we have $x_{v_{cur}} \in x_c + \delta\mathbb{B}$.

Note that $x_{rand} \in x_c + \delta/5\mathbb{B}$ implies x_{rand} is sampled from the ball centered at x_c with radius $\delta/5$. Therefore, by Definition 4.18 and (4.21), the probability of $x_{rand} \in x_c + \delta/5\mathbb{B}$ is $\zeta_n(\delta/5)^n/\mu(S)$, where ζ_n denotes the Lebesgue measure of the unit ball in $\mathbb{R}^n$.

B.5 Proof of Lemma 4.33

B.5.1 Supporting Lemmas to Prove Lemma 4.33

Lemma B.6 (Lemma 2 in [27])**.** *Given a hybrid dynamical system $\mathcal{H}$ that satisfies Assumption 4.24, let $\psi = (\phi, \upsilon)$ and $\psi' = (\phi', \upsilon')$ be two purely continuous solution pairs to $\mathcal{H}$ with $(T, 0) = \max dom\ \psi$, $(T', 0) = \max dom\ \psi'$ and $T' \leq T$. The input functions υ and υ' are assumed to be constant, denoted $\upsilon : [0, T] \times \{0\} \to u_C \in U_C$ and $\upsilon' : [0, T'] \times \{0\} \to u'_C \in U_C$, respectively. Suppose initial state $\phi(0, 0) \in \phi'(0, 0) + \delta\mathbb{B}$ for some $\delta > 0$. Then $|\phi(T') - \phi'(T')| \leq e^{K_x^f T'}\delta + K_u^f T' e^{K_x^f T'}\Delta u$, where $\Delta u = |u_C - u'_C|$, K_x^f and K_u^f are from Assumption 4.24.*

Lemma B.7. *Given a hybrid dynamical system $\mathcal{H}$ that satisfies Assumption 4.24, let $\psi = (\phi, \upsilon)$ and $\psi' = (\phi', \upsilon')$ be two purely continuous solution pairs to $\mathcal{H}$ with $(T, 0) =$*

$\max dom\ \psi,\ (T',0) = \max dom\ \psi'$ *and* $T' \leq T$. *The input functions* υ *and* υ' *are assumed to be constant, denoted* $\upsilon : [0,T] \times \{0\} \to u_C \in U_C$ *and* $\upsilon' : [0,T'] \times \{0\} \to u'_C \in U_C$, *respectively. Suppose initial state* $\phi(0,0) \in \phi'(0,0) + \kappa_1\delta\mathbb{B}$ *for some* $\delta > 0$ *and* $\kappa_1 \in (0,1/2)$. *Then, for each* $\kappa_2 \in (2\kappa_1, 1)$ *and each* $\epsilon \in (0, \frac{\kappa_2\delta}{2})$, ϕ *and* ϕ' *are* $(\overline{T}, \kappa_2\delta)$*-close where* $\overline{T} = \max\{T, T'\}$ *if the following hold:*

1. T *and* T' *are such that*

$$\begin{aligned} &T' \in T_k = \{t_l \in [\max\{T - \kappa_2\delta, 0\}, T] : \\ &\forall t' \in [t_l, T], \phi(t', 0) + (\frac{\kappa_2\delta}{2} - \epsilon)\mathbb{B} \subset \phi(T, 0) + \frac{\kappa_2\delta}{2}\mathbb{B}\}, \end{aligned} \tag{B.9}$$

2. u_C *and* u'_C *are such that*

$$|u_C - u'_C| < \frac{\frac{\kappa_2\delta}{2} - \epsilon - \exp(K_x^f T)\kappa_1\delta}{K_u^f T \exp(K_x^f T)}. \tag{B.10}$$

Proof. We show that ψ and ψ' satisfy each item in Definition 2.6.

1. This item shows that ψ and ψ' satisfy the first condition in Definition 2.6, namely, for all $(t,j) \in \text{dom } \phi$ with $t + j \leq \overline{T}$, there exists s such that $(s,j) \in \text{dom } \phi'$, $|t - s| < \kappa_2\delta$, and $|\phi(t,j) - \phi'(s,j)| < \kappa_2\delta$.

   Because of Assumption 4.24, according to Lemma B.6, it follows

$$|\phi(t,0) - \phi'(t,0)| \leq \exp(K_x^f t)\kappa_1\delta + K_u^f t \exp(K_x^f t)|u_C - u'_C|. \tag{B.11}$$

   Note that $T' \in T_k \subset [\max\{T - \kappa_2\delta, 0\}, T]$. Therefore, we have $T' \leq T$. Because $T' \leq T$, for all $(t,0) \in \text{dom } \phi'$ with $t + 0 \leq T' + 0 \leq T + 0 = T = \overline{T}$, there exists

$s = t$ such that $(s, 0) \in \text{dom } \phi$, $|t - s| = 0 < \kappa_2\delta$. Then, by applying (B.10) to (B.11), we have

$$\begin{aligned}
|\phi(t,0) - \phi'(s,0)| &= |\phi(t,0) - \phi'(t,0)| \\
&\leq \exp(K_x^f t)\kappa_1\delta + K_u^f t \exp(K_x^f t)\Delta u \\
&\leq \exp(K_x^f T)\kappa_1\delta + K_u^f T \exp(K_x^f T)\Delta u \\
&\leq \frac{\kappa_2\delta}{2} - \epsilon < \kappa_2\delta.
\end{aligned}$$

Hence, item 1 in Definition 2.6 is established.

2. This item shows that ψ and ψ' satisfy the second condition in Definition 2.6, namely, for all $(t, j) \in \text{dom } \phi'$ with $t + j \leq \overline{T} = T$, there exists s such that $(s, j) \in \text{dom } \phi$, $|t - s| < \kappa_2\delta$, and $|\phi'(t, j) - \phi(s, j)| < \kappa_2\delta$.

   We consider the following two cases.

   (a) For all $(t, 0) \in \text{dom } \phi$ with $0 \leq t + 0 \leq T' + 0 = T'$, there exists $s = t$ such that $(s, 0) \in \text{dom } \phi'$, $|t - s| = 0 < \kappa_2\delta$ and

   $$|\phi(t,0) - \phi'(s,0)| \leq \frac{\kappa_2\delta}{2} - \epsilon < \kappa_2\delta - \epsilon < \kappa_2\delta$$

   because of (B.11).

   (b) This item considers the case of $(t, 0) \in \text{dom } \phi$ with $T' \leq t + 0 \leq T + 0 = T$. For all $(t, 0) \in \text{dom } \phi$ with $T' \leq t + 0 \leq T + 0 = T$, let $s = T'$. Because of (B.9), then $s = T' \in [\max\{T - \kappa_2\delta, 0\}, T]$. Since $s \in [\max\{T - \kappa_2\delta, 0\}, T]$ and $t \in [T', T]$, therefore, we have

   $$|t - s| \leq \kappa_2\delta.$$

Also, because of the definition of T_k in (B.9), for all $(t,0) \in \operatorname{dom} \phi$ with $T' \leq t \leq T$, we have

$$\phi(t,0) \in \phi(t,0) + (\frac{\kappa_2 \delta}{2} - \epsilon)\mathbb{B} \subset \phi(T,0) + \frac{\kappa_2 \delta}{2}\mathbb{B}.$$

Because of (B.11) and (B.9), we also have

$$\begin{aligned}\phi'(s,0) &\in \phi(s,0) + (\frac{\kappa_2 \delta}{2} - \epsilon)\mathbb{B} \\ &\subset \phi(T,0) + \frac{\kappa_2 \delta}{2}\mathbb{B}.\end{aligned}$$

Therefore, $\phi(t,0)$ and $\phi'(s,0)$ are two points in a ball centered at $\phi(T,0)$ with radius $\frac{\kappa_2 \delta}{2}$. Note that the maximum distance between two points within a ball is its diameter. Hence, we have

$$|\phi(t,0) - \phi'(s,0)| < \kappa_2 \delta.$$

Therefore, when both (B.9) and (B.10) hold, ϕ and ϕ' are $(\overline{T}, \kappa_2\delta)$-close. □

Lemma B.8. *Given a hybrid dynamical system $\mathcal{H}$ that satisfies Assumption 4.24, let $\psi = (\phi, \upsilon)$ and $\psi' = (\phi', \upsilon')$ be two purely continuous solution pairs to $\mathcal{H}$ with $(T,0) = \max dom\ \psi$, $(T',0) = \max dom\ \psi'$ and $T' \leq T$. The input functions υ and υ' are assumed to be constant, denoted $\upsilon : [0,T] \times \{0\} \to u_C \in U_C$ and $\upsilon' : [0,T'] \times \{0\} \to u'_C \in U_C$, respectively. Suppose initial state $\phi(0,0) \in \phi'(0,0) + \kappa_1 \delta \mathbb{B}$ for some $\delta > 0$ and $\kappa_1 \in (0, 1/2)$. Then, for each $\kappa_2 \in (2\kappa_1, 1)$ and each $\epsilon \in (0, \frac{\kappa_2 \delta}{2})$, $\phi'(T',0) \in \phi(T,0) + \kappa_2 \delta \mathbb{B}$ if the following hold:*

1. T and T' are such that

$$\begin{aligned} &T' \in T_k = \{t_l \in [\max\{T - \kappa_2\delta, 0\}, T] : \\ &\forall t' \in [t_l, T], \phi(t', 0) + (\frac{\kappa_2\delta}{2} - \epsilon)\mathbb{B} \subset \phi(T, 0) + \frac{\kappa_2\delta}{2}\mathbb{B}\}, \end{aligned} \tag{B.12}$$

2. u_C and u'_C are such that

$$|u_C - u'_C| < \frac{\frac{\kappa_2\delta}{2} - \epsilon - \exp(K_x^f T)\kappa_1\delta}{K_u^f T \exp(K_x^f T)}. \tag{B.13}$$

Proof. Because of Assumption 4.24, according to Lemma B.6, we have

$$\begin{aligned} &|\phi(t, 0) - \phi'(t, 0)| \leq \exp(K_x^f t)\kappa_1\delta + \\ &K_u^f t \exp(K_x^f t)|u_C - u'_C|. \end{aligned} \tag{B.14}$$

Note that $T' \in T_k \subset [\max\{T - \kappa_2\delta, 0\}, T]$, therefore, we have $T' \leq T$. Then, by applying $T' \leq T$ and (B.13) to (B.14), we have

$$\begin{aligned} &|\phi(T', 0) - \phi'(T', 0)| \\ &\leq \exp(K_x^f T')\kappa_1\delta + K_u^f T' \exp(K_x^f T')|u_C - u'_C| \\ &\leq \exp(K_x^f T)\kappa_1\delta + K_u^f T \exp(K_x^f T)|u_C - u'_C| \\ &< (\frac{\kappa_2\delta}{2} - \epsilon). \end{aligned}$$

Therefore, we have $\phi'(T', 0) \in \phi(T', 0) + (\frac{\kappa_2\delta}{2} - \epsilon)\mathbb{B}$.

Since (B.12) holds, then we have

$$\begin{aligned} \phi(T', 0) + (\frac{\kappa_2\delta}{2} - \epsilon)\mathbb{B} &\subset \phi(T, 0) + \frac{\kappa_2\delta}{2}\mathbb{B} \\ &\subset \phi(T, 0) + \kappa_2\delta\mathbb{B}. \end{aligned}$$

Therefore, we have $\phi'(T', 0) \in \phi(T', 0) + (\frac{\kappa_2\delta}{2} - \epsilon)\mathbb{B} \subset \phi(T, 0) + \kappa_2\delta\mathbb{B}$. □

B.5.2 Proof of Lemma 4.33

This proof proceeds as follows. From item 1 in Definition 4.20, we denote the constant input signal that is randomly selected from $\mathcal{U}_C$ as $\tilde{\upsilon}' : [0, t'_m] \to u'_C \in U_C$, where t'_m denotes the time duration of $\tilde{\upsilon}'$. Since in the statement of Lemma 4.33, the input function υ is also constant, denote $\upsilon : [0, \tau] \times \{0\} \to u_C \in U_C$. Under Assumption 4.24, Lemma B.7 and Lemma B.8 guarantee that both E_1 and E_2 occur if t'_m and u'_C satisfy the following conditions:

1. t'_m is such that

$$\begin{aligned} t'_m \in T_k := \{t_l \in [\max\{\tau - \kappa_2\delta, 0\}, \tau] : \\ \phi(t', 0) + r'\mathbb{B} \subset \phi(\tau, 0) + \frac{\kappa_2\delta}{2}\mathbb{B} \quad \forall t' \in [t_l, \tau]\} \end{aligned} \tag{B.15}$$

   where $r' = \frac{\kappa_2\delta}{2} - \epsilon$ and the elements in (B.15) come from (B.9) and (B.12).

2. u'_C is such that

$$0 \leq \Delta u \leq \frac{\frac{\kappa_2\delta}{2} - \epsilon - \exp(K_x^f \tau)\kappa_1\delta}{K_u^f \tau \exp(K_x^f \tau)} \tag{B.16}$$

   where $\Delta u := |u_C - u'_C|$ and the elements in (B.16) come from (B.10) and (B.13).

Then, by the uniform execution of HyRRT as defined in Definition 4.18, we proceed to characterize the probability of selecting $\tilde{\upsilon}'$, namely, selecting t'_m and u'_C satisfying (B.15) and (B.16), respectively, and provide a lower bound as in (4.22).

We first show that set T_k has positive Lebesgue measure, which will be used to characterize p_t in (4.22). Since ϕ is purely continuous, for arbitrary small $\epsilon > 0$, there exists a lower bound $t'_l \in (0, \tau)$ such that

$$|\phi(t', 0) - \phi(\tau, 0)| < \epsilon \tag{B.17}$$

for each $t' \in [t'_l, \tau]$. For each $t' \in [t'_l, \tau]$ and each point $x_p \in \phi(t', 0) + r'\mathbb{B}$, by the triangle inequality, it follows that:

$$\begin{aligned} |x_p - \phi(\tau, 0)| &= |x_p - \phi(t', 0) + \phi(t', 0) - \phi(\tau, 0)| \\ &\leq |x_p - \phi(t', 0)| + |\phi(t', 0) - \phi(\tau, 0)|. \end{aligned}$$

From $|x_p - \phi(t', 0)| \leq r' = \frac{\kappa_2 \delta}{2} - \epsilon$ and (B.17), it follows that

$$\begin{aligned} |x_p - \phi(\tau, 0)| &\leq |x_p - \phi(t', 0)| + |\phi(t', 0) - \phi(\tau, 0)| \\ &\leq r' + \epsilon = \frac{\kappa_2 \delta}{2}. \end{aligned} \tag{B.18}$$

From (B.18), for each point $x_p \in \phi(t', 0) + r'\mathbb{B}$, it follows $x_p \in \phi(\tau, 0) + \frac{\kappa_2 \delta}{2}\mathbb{B}$. Therefore, we have

$$\phi(t', 0) + r'\mathbb{B} \subset \phi(\tau, 0) + \frac{\kappa_2 \delta}{2}\mathbb{B}$$

for each $t' \in [t'_l, \tau]$. This leads to the existence of $t'_l < \tau$ such that

$$t'_l \in \overline{T}_k := \left\{ t_l \in [0, \tau] : \forall t' \in [t_l, \tau], \phi(t', 0) + r'\mathbb{B} \subset \phi(\tau, 0) + \frac{\kappa_2 \delta}{2}\mathbb{B} \right\},$$

implying that $\mu(\overline{T}_k) \geq \tau - t'_l > 0$, where $\mu(\overline{T}_k)$ denotes the Lebesgue measure of $\overline{T}_k$. Since the interval $[\max\{\tau - \kappa_2 \delta, 0\}, \tau]$ has positive Lebesgue measure and the intervals $\overline{T}_k$ and $[\max\{\tau - \kappa_2 \delta, 0\}, \tau]$ are both upper bounded by τ, then $T_k = \overline{T}_k \cap [\max\{\tau - \kappa_2 \delta, 0\}, \tau]$ has positive Lebesgue measure. By (4.21), the probability of selecting the t'_m such that (B.15) is satisfied, denoted p_t, is computed as follows:

$$p_t = \frac{\mu(T_k)}{T_m} \in (0, 1]$$

where T_m is from Definition 4.20.

Next, we discuss the conditions on u'_C such that both E_1 and E_2 occur. In addition to (B.16), to ensure that no intersection between ψ_{new} and the unsafe set X_u prevents the return of ψ_{new} in the function call `new_state`, the following condition is also required:

$$\Delta u \leq \delta. \tag{B.19}$$

Therefore, to ensure that both E_1 and E_2 occur, u'_C need to satisfy both (B.16) and (B.19), namely,

$$\Delta u \leq \min \left\{ \frac{\frac{\kappa_2 \delta}{2} - \epsilon - \exp(K_x^f \tau)\kappa_1 \delta}{K_u^f \tau \exp(K_x^f \tau)}, \delta \right\}. \tag{B.20}$$

Note that the choice of u'_C that satisfies (B.20) is a ball in $\mathbb{R}^m$ centered at u_C with radius $\max\left\{\min\left\{\frac{\frac{\kappa_2\delta}{2}-\epsilon-\exp(K_x^f\tau)\kappa_1\delta}{K_u^f\tau\exp(K_x^f\tau)}, \delta\right\}, 0\right\}$, where the operator $\max\{\cdot, 0\}$ prevents the negative values for the radius. Therefore, according to (4.21), the probability of selecting u'_C satisfying (B.16), denoted p_u, is

$$p_u = \frac{\zeta_n \left(\max\left\{ \min\left\{ \frac{\frac{\kappa_2\delta}{2}-\epsilon-\exp(K_x^f\tau)\kappa_1\delta}{K_u^f\tau\exp(K_x^f\tau)}, \delta \right\}, 0 \right\} \right)^m}{\mu(U_C)}.$$

Hence, we have

$$\begin{aligned} &\text{Prob}[E_1 \& E_2] \geq p_t p_u \\ &= p_t \frac{\zeta_n \left(\max\left\{ \min\left\{ \frac{\frac{\kappa_2\delta}{2}-\epsilon-\exp(K_x^f\tau)\kappa_1\delta}{K_u^f\tau\exp(K_x^f\tau)}, \delta \right\}, 0 \right\} \right)^m}{\mu(U_C)}. \end{aligned}$$

B.6 Proof of Lemma 4.35

Under Assumption 4.27, we have

$$|\phi(0,1) - \phi_{new}(0,1)| \leq K_x^g |\phi(0,0) - \phi_{new}(0,0)| + K_u^g \Delta u$$

where $\Delta u = |\upsilon(0,0) - \upsilon_{new}(0,0)|$. From $\phi_{new}(0,0) \in \phi(0,0) + \kappa_1 \delta \mathbb{B}$, we have

$$|\phi(0,1) - \phi_{new}(0,1)| \leq K_x^g \kappa_1 \delta + K_u^g \Delta u \tag{B.21}$$

with $K_x^g, K_u^g > 0$. We denote the input value that is randomly selected from $\mathcal{U}_D$ in the function call `new_state` as $u_D \in \mathbb{R}^m$. Given that u_D is input to the discrete dynamics simulator (4.8) to simulate $(\phi_{new}, \upsilon_{new})$, it follows that $\upsilon_{new}(0,0) = u_D$ and, hence, $\Delta u = |\upsilon(0,0) - u_D|$.

Then, we show that if

$$\Delta u \leq \frac{(\kappa_2 - K_x^g \kappa_1)\delta}{K_u^g}, \tag{B.22}$$

then, the probabilistic event E occurs. From (B.22), it follows from (B.21) that

$$|\phi(0,1) - \phi_{new}(0,1)| \leq K_x^g \kappa_1 \delta + K_u^g \Delta u \leq \kappa_2 \delta,$$

namely, $\phi_{new}(0,1) \in \phi(0,1) + \kappa_2 \delta \mathbb{B}$, implying that E occurs.

To ensure that no intersection between ψ_{new} and the unsafe set X_u prevents the return of ψ_{new} in the function call `new_state`, the following condition on the sampling result of u_D is also required:

$$0 \leq \Delta u = |\upsilon(0,0) - u_D| \leq \delta. \tag{B.23}$$

Therefore, to satisfy both (B.22) and (B.23) such that E occurs, we have

$$0 \leq \Delta u \leq \min\left\{\frac{(\kappa_2 - K_x^g \kappa_1)\delta}{K_u^g}, \delta\right\}. \tag{B.24}$$

By (4.21), the probability to randomly select an input value from $\mathcal{U}_D$ that satisfies (B.24) is

$$p_u = \frac{\zeta_n \left(\max\left\{\min\left\{\frac{(\kappa_2 - K_x^g \kappa_1)\delta}{K_u^g}, \delta\right\}, 0\right\}\right)^m}{\mu(U_D)} \tag{B.25}$$

where ζ_n is the Lebesgue measure of the unit ball in $\mathbb{R}^m$ and $\mu(U_D)$ denotes the Lebesgue measure of U_D. Therefore, we have

$$\text{Prob}[E] \geq p_u = \frac{\zeta_n \left(\max\left\{\min\left\{\frac{(\kappa_2 - K_x^g \kappa_1)\delta}{K_u^g}, \delta\right\}, 0\right\}\right)^m}{\mu(U_D)}. \tag{B.26}$$

B.7 Closeness Guarantee between the Concatenation Results of Hybrid Arcs

B.7.1 Supporting Lemma

Lemma B.9. *Given two compact hybrid arcs ϕ and ϕ' that are (τ, ϵ)-close, then we have $|T - T'| < \epsilon$ and $J = J'$, where $(T, J) = \max dom\ \phi$, $(T', J') = \max dom\ \phi'$, and $\tau = \max\{T + J, T' + J'\}$.*

Proof. We prove

$$|T - T'| < \epsilon$$

by contradiction. Suppose

$$T - T' \geq \epsilon. \tag{B.27}$$

Since ϕ and ϕ' are (τ, ϵ)-close, from Definition 2.6, for $(T, J) \in \text{dom } \phi$ satisfying $T + J \leq \tau = \max\{T + J, T' + J'\}$, there exists s such that $(s, J) \in \text{dom } \phi'$ and

$$|T - s| < \epsilon. \tag{B.28}$$

By $(s, J) \in \text{dom } \phi'$ and (B.27), it follows that $s \leq T' < T$. Then, by (B.27), we have

$$|T - s| \geq T - T' \geq \epsilon,$$

which contradicts (B.28). Therefore, (B.27) cannot hold. A similar contradiction can be derived if we suppose $T' - T \geq \epsilon$. Therefore, we have $|T - T'| < \epsilon$.

Similarly, we prove $J = J'$ by contradiction. Suppose

$$J > J'. \tag{B.29}$$

Then, from Definition 2.6, for $(T, J) \in \text{dom } \phi$ satisfying $T + J \leq \tau = \max\{T + J, T' + J'\}$, there exists s such that $(s, J) \in \text{dom } \phi'$. However, since $(T', J') = \max \text{dom } \phi'$ and (B.29), such s does not exist. Therefore, (B.29) cannot hold. A similary contradiction can also be derived if we suppose $J < J'$. Therefore, we have $J = J'$. □

B.7.2 Closeness Guarantee

Next, we demonstrate that the operation of concatenation preserves the closeness between the hybrid arcs.

Proposition B.10. *Given compact hybrid arcs ϕ_1, ϕ_2, ϕ_1', and ϕ_2' such that ϕ_1 and ϕ_1' are (τ_1, ϵ_1)-close and ϕ_2 and ϕ_2' are (τ_2, ϵ_2)-close, where $(T_1, J_1) = \max \text{dom } \phi_1$,*

$(T_1', J_1') = \max dom\ \phi_1'$ *and* $\tau_1 = \max\{T_1 + J_1, T_1' + J_1'\}$, *then* $\phi_1|\phi_2$ *and* $\phi_1'|\phi_2'$ *are* $(\tau_1 + \tau_2, \epsilon_1 + \epsilon_2)$-*close.*

Proof. We show that $\phi = \phi_1|\phi_2$ and $\phi' = \phi_1'|\phi_2'$ satisfy the following, as introduced in Definition 2.6:

C1. for all $(t, j) \in \text{dom } \phi$ with $t + j \leq \tau_1 + \tau_2$, there exists s such that $(s, j) \in \text{dom } \phi'$, $|t - s| < \epsilon_1 + \epsilon_2$, and $|\phi(t, j) - \phi'(s, j)| < \epsilon_1 + \epsilon_2$;

C2. for all $(t, j) \in \text{dom } \phi'$ with $t + j \leq \tau_1 + \tau_2$, there exists s such that $(s, j) \in \text{dom } \phi$, $|t - s| < \epsilon_1 + \epsilon_2$, and $|\phi'(t, j) - \phi(s, j)| < \epsilon_1 + \epsilon_2$.

First, we show that C1 holds. From Definition 3.5, it follows that dom ϕ = dom $\phi_1 \cup$ (dom $\phi_2 + \{(T_1, J_1)\}$), where $(T_1, J_1) = \max \text{dom } \phi_1$. Then, for all $(t, j) \in \text{dom } \phi =$ dom $\phi_1 \cup (\text{dom } \phi_2 + \{(T_1, J_1)\})$ with $t + j \leq \tau_1 + \tau_2$, we show that C1 is satisfied when 1) $(t, j) \in \text{dom } \phi_1$ and when 2) $(t, j) \in \text{dom } \phi_2 + \{(T_1, J_1)\}$, respectively.

1. For all $(t, j) \in \text{dom } \phi$ such that $(t, j) \in \text{dom } \phi_1$, namely, $t + j \leq T_1 + J_1$, given that $\tau_1 = \max\{T_1 + J_1, T_1' + J_1'\}$, it follows that

$$t + j \leq \tau_1 = \max\{T_1 + J_1, T_1' + J_1'\}.$$

   Since ϕ_1 and ϕ_1' are (τ_1, ϵ_1)-close, there exists s' such that $(s', j) \in \text{dom } \phi_1'$, $|t - s'| < \epsilon_1$, and $|\phi_1(t, j) - \phi_1'(s', j)| < \epsilon_1$. Therefore, there exists $s = s'$ such that $(s, j) \in \text{dom } \phi_1' \subset \text{dom } \phi'$, $|t - s| < \epsilon_1 < \epsilon_1 + \epsilon_2$, and $|\phi(t, j) - \phi'(s, j)| = |\phi_1(t, j) - \phi_1'(s, j)| < \epsilon_1 < \epsilon_1 + \epsilon_2$. Hence, C1 is established.

2. For all $(t, j) \in \text{dom } \phi$ such that $(t, j) \in (\text{dom } \phi_2 + \{(T_1, J_1)\})$, namely, $t + j \geq T_1 + J_1$, since ϕ_2 and ϕ_2' are (τ_2, ϵ_2)-close, there exists s' such that

$$(s', j - J_1) \in \text{dom } \phi_2' \tag{B.30}$$

$$|t - T_1 - s'| < \epsilon_2, \tag{B.31}$$

$$|\phi_2(t - T_1, j - J_1) - \phi_2'(s', j - J_1)| < \epsilon_2. \tag{B.32}$$

Since ϕ_1 and ϕ_1' are (τ_1, ϵ_1)-close where $\tau_1 = \max\{T_1 + J_1, T_1' + J_1'\}$, by Lemma B.9, we have

$$|T_1 - T_1'| \leq \epsilon_1 \tag{B.33}$$

and

$$J_1 = J_1'. \tag{B.34}$$

By (B.30) and (B.34), it follows that s' is such that

$$(s' + T_1', j) \in \text{dom } \phi_2' + \{(T_1', J_1)\} = \text{dom } \phi_2' + \{(T_1', J_1')\} \subset \text{dom } \phi'.$$

Furthermore, because of (B.31) and (B.33), by the triangle inequality, it follows that

$$|t - (s' + T_1')| \leq |t - T_1 - s'| + |T_1 - T_1'| < \epsilon_2 + \epsilon_1.$$

By (B.32), it follows that

$$|\phi(t, j) - \phi'(s' + T_1', j)| = |\phi_2(t - T_1, j - J_1) - \phi_2'(s', j - J_1)|$$

$$< \epsilon_2 < \epsilon_1 + \epsilon_2.$$

Therefore, we can find $s = s' + T_1'$ such that $(s, j) \in \text{dom } \phi'$, $|t - s'| < \epsilon_1 + \epsilon_2$, and $|\phi(t, j) - \phi'(s', j)| < \epsilon_1 + \epsilon_2$.

The proof for C2 follows a similar logic to the aforementioned arguments, achieved by swapping ϕ and ϕ'. Therefore, $\phi_1|\phi_2$ and $\phi_1'|\phi_2'$ are $(\tau_1+\tau_2, \epsilon_1+\epsilon_2)$-close. □

B.8 Definition of Truncation and Translation Operation

Definition B.11 (Truncation and translation operation)**.** *Given a function $\phi : dom\ \phi \to \mathbb{R}^n$, where dom ϕ is hybrid time domain, and pairs of hybrid time $(T_1, J_1) \in dom\ \phi$ and $(T_2, J_2) \in dom\ \phi$ such that $T_1 \leq T_2$ and $J_1 \leq J_2$, the function $\widetilde{\phi} : dom\ \widetilde{\phi} \to \mathbb{R}^n$ is the truncation of ϕ between (T_1, J_1) and (T_2, J_2) and translation by (T_1, J_1) if*

1. *$dom\ \widetilde{\phi} = dom\ \phi \cap ([T_1, T_2] \times \{J_1, J_1+1, ..., J_2\}) - \{(T_1, J_1)\}$, where the minus sign denotes Minkowski difference;*

2. *$\widetilde{\phi}(t, j) = \phi(t + T_1, j + J_1)$ for all $(t, j) \in dom\ \widetilde{\phi}$.*

Appendix C

Proof for Results in Chapter 5

C.1 Proof of Lemma 5.3

Given that ψ^{bw} is purely continuous, $\psi^{\mathrm{bw}'}$ is purely continuous and, by (5.5), P1 is established immediately. By the Lipschitz continuity of the flow map f in Assumption 4.24, since ϕ^{r} is generated by (5.4), [27, Lemma 2] (also presented in the appendix as Lemma B.6) establishes that $|\phi^{\mathrm{r}}(T,0) - \phi(T,0)| \leq \exp\left(K_x^f T\right)\delta$.

C.2 Proof of Lemma 5.4

Given that ψ^{bw} is purely discrete, $\psi^{\mathrm{bw}'}$ is purely discrete and P1 is immediately established by (5.5). Furthermore, since ϕ^{r} is generated by (5.4), in accordance with

Assumption 4.27, we derive the following inequality:

$$\begin{aligned}
|\phi^{\mathrm{r}}(0,J)-\phi(0,J)| &\leq \exp(\ln K_x^g)|\phi^{\mathrm{r}}(0,J-1)-\phi(0,J-1)| \\
&\leq \exp(2\ln K_x^g)|\phi^{\mathrm{r}}(0,J-2)-\phi(0,J-2)| \\
&\leq ... \\
&\leq \exp(J\ln K_x^g)|\phi^{\mathrm{r}}(0,0)-\phi(0,0)| = \exp(J\ln K_x^g)\delta,
\end{aligned}$$

thus establishing P2.

Appendix D

Proof for Results in Chapter 6

D.1 Supporting Result for Theorem 6.9

The forth coming Lemma D.1 shows that the pruning process guarantees the existence of a vertex close to the optimal motion planning with a decreasing cost if a vertex that is close enough to the optimal motion plan has been generated.

Lemma D.1 (Lemma 27 in [35]). *Suppose Assumption 6.7 and Assumption 6.8 hold. Let* $\delta_c = \delta - \delta_{BN} - 2\delta_s$. *If vertex* $v \in V_{active}$ *is generated at iteration* k *so that* $\overline{x}_v \in \mathcal{B}_{\delta_c}(x^*)$ *where* x^* *is a state on the motion plan with positive clearance* δ, *then for every iteration* $k' > k$, *then there exists a vertex* $v' \in V_{active}$ *so that* $\overline{x}_{v'} \in x^* + (\delta - \delta_{BN})\mathbb{B}$ *and* $\overline{c}_{v'} \leq \overline{c}_v$.

Remark D.2. *When a trajectory is generated that ends in* $\mathcal{B}_{\delta_c}(x^*)$, *Lemma D.1 guarantees that there will always be an active vertex associated with a state in* $\mathcal{B}_{\delta}(x^*)$.

The forthcoming Lemma D.3 characterize the probability that a vertex that is close to the optimal motion plan is selected by the `best_near_selection` function call.

Lemma D.3. *(Lemma 28 in [35]) Suppose Assumption 6.7, Assumption 4.20, Assumption 4.18 hold, if there exists $v \in V_{active}$ such that $\overline{x}_v \in x^* + \delta_c \mathbb{B}$ at iteration k, then the probability that* `best_near_selection` *for propagation a vertex v' such that $\overline{x}_{v'} \in x^* + \delta\mathbb{B}$ can be lower bounded by a positive constant γ_{select} for every $k' > k$.*

D.2 Proof for Theorem 6.9

First, because the cost function c is Lipchitz continuous, non-decreasing, and non-degenerate (Items 1, 2, 3.c, and 3.d in Assumption 6.5), and Lemma 4.15 guarantees that the motion plan has positive clearance, a sequence of balls $\{\mathcal{B}_i\}_{i=0}^N$ centered at the motion plan is constructed such that the costs of the truncation between the centers of the consecutive balls are constant. Then, Lemma D.3 shows that the probability that `best_near_selection` selects a vertex in $\mathcal{B}_i$ is positive and Lemma 4.33 and Lemma 4.35 show that the probability that `new_state` generates a new vertex in $\mathcal{B}_{i+1}$ is positive. Therefore, by using induction, we can show that the probability that HySST generates a solution that is close to the optimal motion plan converges to one as the number of iterations goes to infinity. Lemma D.1 guarantees that the pruning process will only improve the quality of the generated motion plan. Items 1) - 2) in Assumption 6.5 guarantee that the closeness between the generated motion plan and the optimal motion plan leads to a bounded differences between their costs.

Since ψ^* is assumed to have positive safety clearance $\delta_s > 0$ and HySST is used to solve $\mathcal{P}^*_{\delta_f}$, then according to Lemma 4.15, ψ^* is a motion plan to $\mathcal{P}^*_{\delta_f}$ with a positive

clearance at least $\delta = \min\{\delta_s, \delta_f\}$.

Then consider a sequence of hybrid time instance $S_{hti} := \{(T_i, J_i) \in \text{dom } \psi^*\}_{i=0}^N$ such that $0 = T_0 + J_0 < T_1 + J_1 < ... < T_N + J_N = T + J$ where $(T, J) = \max \text{dom } \psi^*$, and, for $i = 0, 1, ..., N-1$ and some constant $\Delta c_f \in \mathbb{R}_{\geq 0}$, either of the following holds:

1. $J_i = J_{i+1}$ and $c(\widetilde{\phi^*}_{(T_i,J_i)\to(T_{i+1},J_{i+1})}) = \Delta c_f$;

2. $J_{i+1} = J_i + 1$ and $T_{i+1} = T_i$.

Namely, using such a sequence of hybrid time instance, ψ^* can be truncated into either, say, N_f number of purely continuous segments whose cost equals Δc_f, or, say, N_g number of purely discrete segments with a single jump such that

$$N_f + N_g = N. \tag{D.1}$$

Denote the minimal cost of the purely discrete segments as Δc_j.

Constructing the sequence following the second item above is trivial because it can be implemented by selecting all the hybrid time instances before and after all the jumps in ψ^*. Note that the cost function is continuous, non-decreasing and non-degeneracy along the state trajectory during flows because 3.a - 3.c in Assumption 6.5 are assumed. Suppose that $[T_i, T_i + \Delta T] \times \{J_i\} \subset \text{dom } \psi^*$ for some $\Delta T > 0$. Then there exists $\Delta c_f > 0$ such that $0 = c(\widetilde{\phi}_{(T_i,J_i)\to(T_i+0,J_i)}) < \Delta c_f \leq c(\widetilde{\phi}_{(T_i,J_i)\to(T_i+\Delta T,J_i)})$. In the meantime, given this Δc_f, there exists $\Delta t \in (0, \Delta T]$ such that $c(\widetilde{\phi}_{(T_i,J_i)\to(T_i+\Delta t,J_i)}) = \Delta c_f$. Therefore, the sequence of the hybrid time instances following the first item above exists for a sufficiently small Δc_f.

From S_{hti}, we can build a sequence of balls $S_b := \{c_i + r_i\mathbb{B}\}_{i=0}^N$ in $\mathbb{R}^n$ such that

1. $c_i := \phi^*(T_i, J_i)$ for all $i = 0, 1, ..., N$;

2. $r_i := \delta$.

Then, denote $\delta_c = \delta - \delta_{BN} - 2\delta_c$. Let $A_i^{(k)}$ denote event that at the k-th iteration of HySST, a solution pair $\psi_{new} = (\phi_{new}, u_{new})$ is generated such that

1. $\phi_{new}(0, 0) \in \phi^*(T_i, J_i) + \delta\mathbb{B}$;

2. $\phi_{new}(T_{new}, J_{new}) \in \phi^*(T_{i+1}, J_{i+1}) + \delta_c\mathbb{B}$ where $(T_{new}, J_{new}) = \max \text{dom } \phi_{new}$;

3. ϕ_{new} is $(\overline{\tau}, \delta)$-close to $\widetilde{\phi^*}_{(T_i,J_i)\to(T_{i+1},J_{i+1})}$, where $(T_{new}, J_{new}) = \max \text{dom } \phi_{new}$, $\overline{\tau} = \max\{T_{new} + J_{new}, T_{i+1} - T_i + J_{i+1} - J_i\}$.

Then, let $E_i^{(k)}$ denote the event that from iteration j from 1 to k, at least one such ψ_{new} is generated, namely, at least one j such that $A_i^{(j)}$ occurs.

Based on the definitions of the events $E_i^{(k)}$ and $A_i^{(k)}$, the probability that event $E_i^{(k)}$ fails, denoted $\neg E_i^{(k)}$, depends on a sequence of A_i events failing:

$$Pr(\neg E_i^{(k)}) = Pr(\neg A_i^{(1)}) \times Pr(\neg A_i^{(2)} | \neg A_i^{(1)}) \cdots Pr(\neg A_i^{(k)} | \bigcap_{j=1}^{k-1} \neg A_i^{(j)}). \tag{D.2}$$

The probability that $\neg A_i^k$ occurs given $\bigcap_{j=1}^{k-1} \neg A_i^{(j)}$ is equivalent with

1. the probability that HySST fails to generate a vertex in the search tree such that its associated state is in $\phi^*(T_i, J_i) + \delta_c\mathbb{B}$, plus

2. the probability that HySST generates a vertex in the search tree such that its associated state is in $\phi^*(T_i, J_i) + \delta_c\mathbb{B}$ but fails to generate a new solution pairs $\psi_{new} = (\phi_{new}, u_{new})$ such that $\phi_{new}(T_{new}, J_{new}) \in \phi^*(T_{i+1}, J_{i+1}) + \delta_c\mathbb{B}$ where

$(T_{new}, J_{new}) = \max \text{dom } \phi_{new}$, which can be characterized by the γ_{select} in Lemma D.3, and $\gamma_{fg} := \min\{\gamma_f, \gamma_g\}$ where γ_f comes from Lemma 4.33 and γ_g comes from Lemma 4.35.

Therefore,

$$\begin{aligned} Pr(\neg A_i^{(k)} | \bigcap_{j=1}^{k-1} \neg A_i^{(j)}) &= Pr(\neg E_{i-1}^k) + Pr(E_{i-1}^k) \times Pr(\phi_{new}(T_{new}, J_{new}) \notin \phi^*(T_i, J_i) + \delta_c \mathbb{B}) \\ &\leq Pr(\neg E_{i-1}^k) + Pr(E_{i-1}^k) \times (1 - \gamma_{select}\gamma_{fg}) \\ &= 1 - Pr(E_{i-1}^k)\gamma_{select}\gamma_{fg}). \end{aligned} \tag{D.3}$$

Using (D.2) and (D.3), we have

$$Pr(E_i^{(k)}) \geq 1 - \prod_{j=1}^{k}(1 - Pr(E_{i-1}^j)\gamma_{select}\gamma_{fg})). \tag{D.4}$$

Then we can use the following induction to prove that if $\lim_{k\to\infty} Pr(E_i^{(k)}) = 1$, then $\lim_{k\to\infty} Pr(E_{i+1}^{(k)}) = 1$. If this holds, then eventually, HySST will eventually generate a solution that is close to the optimal motion plan.

For the base case, $\lim_{k\to\infty} Pr(E_0^k) = 1$ is true, because when the *random_state* is executed for infinite number of times, eventually a sample will fall into $\phi^*(0,0) + \delta_c\mathbb{B}$. Then we have

$$Pr(E_1^k) \geq 1 - (1 - \gamma_{select}\gamma_{fg})^k. \tag{D.5}$$

which leads to $\lim_{k\to\infty} Pr(E_1^k) = 1$.

For the induction step, assuming that $\lim_{k\to\infty} Pr(E_i^k) = 1$, we need to show that $\lim_{k\to\infty} Pr(E_{i+1}^k) = 1$. Define $y_i^{(k)} = \prod_{j=1}^{k}(1 - Pr(E_{i-1}^j)\gamma_{select}\gamma_{fg})$. The logarithm

of $y_i^{(k)}$ behaves like the follows:

$$\log y_i^{(k)} = \log \prod_{j=1}^{k} (1 - Pr(E_{i-1}^j)\gamma_{select}\gamma_{fg}) = \sum_{j=1}^{k} \log(1 - Pr(E_{i-1}^j)\gamma_{select}\gamma_{fg}). \quad \text{(D.6)}$$

The above leads to the following:

$$\log y_i^{(k)} < \sum_{j=1}^{k} -Pr(E_{i-1}^j)\gamma_{select}\gamma_{fg} = -\gamma_{select}\gamma_{fg} \sum_{j=1}^{k} Pr(E_{i-1}^j). \quad \text{(D.7)}$$

Note that according to the induction assumption, $\lim_{k\to\infty} Pr(E_i^k) = 1$, then $\lim_{k\to\infty} \sum_{j=1}^{k} Pr(E_i^j) = \infty$. Therefore,

$$\lim_{k\to\infty} \log y_{i+1}^{(k)} = -\gamma_{select}\gamma_{fg} \lim_{k\to\infty} \sum_{j=1}^{k} Pr(E_i^j) = -\infty$$

which implies that

$$\lim_{k\to\infty} y_{i+1}^{(k)} = 0.$$

With (D.4) and $\lim_{k\to\infty} y_{i+1}^{(k)} = 0$, it can be shown that

$$\lim_{k\to\infty} Pr(E_{i+1}^k) = 1 - \lim_{k\to\infty} y_{i+1}^{(k)} = 1 - 0 = 1.$$

Therefore, the probability that HySST finds a motion plan that is close to ψ^* is converging to 1 as the number of iteration goes to infinite.

Note that when the event A_i^k occurs, ϕ_{new} is $(\overline{\tau}, \delta)$-close to $\widetilde{\phi^*}_{(T_i,J_i)\to(T_{i+1},J_{i+1})}$, and ϕ_{new} and $\widetilde{\phi^*}_{(T_i,J_i)\to(T_{i+1},J_{i+1})}$ are either both purely continuous or purely discrete. If ϕ_{new} and $\widetilde{\phi^*}_{(T_i,J_i)\to(T_{i+1},J_{i+1})}$ are both purely continuous, since item 1 in Assumption

6.5 is assumed, then

$$
\begin{aligned}
&|c(\phi_{new}) - c(\widetilde{\phi^*}_{(T_i,J_i)\to(T_{i+1},J_{i+1})})| \\
&\leq K_c \sup_{\forall(t,0)\in \mathrm{dom}\ \phi_{new}\cap \mathrm{dom}\ \widetilde{\phi^*}_{(T_i,J_i)\to(T_{i+1},J_{i+1})}} \{|\phi_{new}(t,0) - \widetilde{\phi^*}_{(T_i,J_i)\to(T_{i+1},J_{i+1})}(t,0)|\} \\
&= K_c\delta.
\end{aligned}
\tag{D.8}
$$

If ϕ_{new} and $\widetilde{\phi^*}_{(T_i,J_i)\to(T_{i+1},J_{i+1})}$ are both purely discrete, since item 2 in Assumption 6.5 is assumed, then

$$
\begin{aligned}
|c(\phi_{new}) - c(\widetilde{\phi^*}_{(T_i,J_i)\to(T_{i+1},J_{i+1})})| &\leq K_d \sup_{\forall j\in\{0,1\}} \{|\phi_{new}(0,j) - \widetilde{\phi^*}_{(T_i,J_i)\to(T_{i+1},J_{i+1})}(0,j)|\} \\
&= K_d\delta.
\end{aligned}
\tag{D.9}
$$

Denote the new vertex generated at this iteration as v^i_{new} which is possible to be pruned in the forthcoming iteration. Note that $\phi_{new}(T_{new}, J_{new}) \in \phi^*(T_i, J_i) + \delta_c\mathbb{B}$. Therefore, Lemma D.1 guarantees that the vertex returned by the *best_near_selection* function, denoted $v^{i'}_{new}$ is such that

$$
\overline{c}_{v^{i'}_{new}} \leq \overline{c}_{v^i_{new}}
$$

which implies that the pruning process will not affect (D.8) and (D.9).

Define $K_{\max} := \max\{K_c, K_d\}$. Define $\Delta c_{\min} := \min\{\Delta c_f, \Delta c_g\}$. Because of (D.8), (D.9) and (D.1), then difference between the cost of $\psi^* = (\phi^*, u^*)$ and the cost of $\psi = (\phi, u)$ constructed by concatenating N number of ψ_{new} can be characterized as

follows

$$
\begin{aligned}
c(\phi) &\leq c(\phi^*) + N_f K_c \delta + N_g K_d \delta \leq c(\phi^*) + N K_{\max} \delta \\
&\leq c(\phi^*) + \frac{c(\phi^*)}{\Delta c_{\min}} K_{\max} \delta = (1 + \frac{K_{\max} \times \delta}{\Delta c_{\min}}) c(\phi^*).
\end{aligned}
\tag{D.10}
$$

Bibliography

[1] Zlatan Ajanović, Enrico Regolin, Barys Shyrokau, Hana Ćatić, Martin Horn, and Antonella Ferrara. Search-based task and motion planning for hybrid systems: Agile autonomous vehicles. *Engineering Applications of Artificial Intelligence*, 121:105893, 2023.

[2] Berk Altin and Ricardo G. Sanfelice. On Model Predictive Control for Hybrid Dynamical Systems. Technical report, University of California, Santa Cruz, Department of Electrical and Computer Engineering, 2018. Password: HyMPC-03-18.

[3] Adam Ames, Nan Wang, and Ricardo G Sanfelice. A set-based motion planning algorithm for aerial vehicles in the presence of obstacles exhibiting hybrid dynamics. In *2022 IEEE Conference on Control Technology and Applications (CCTA)*, pages 583–588. IEEE, 2022.

[4] Patrick Billingsley. *Probability and measure*. John Wiley & Sons, 2017.

[5] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[6] Michael S Branicky, Michael M Curtiss, Joshua Levine, and Stuart Morgan. Sampling-based planning and control. In *Proceedings of the 12th Yale Workshop on Adaptive and Learning Systems, New Haven, CT*. Citeseer, 2003.

[7] Michael S Branicky, Michael M Curtiss, Joshua A Levine, and Stuart B Morgan. Rrts for nonlinear, discrete, and hybrid planning and control. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 1, pages 657–663. IEEE, 2003.

[8] Richard L Burden. *Numerical analysis*. Brooks/Cole Cengage Learning, 2011.

[9] Jun Chai and Ricardo G Sanfelice. Forward invariance of sets for hybrid dynamical systems (part i). *IEEE Transactions on Automatic Control*, 64(6):2426–2441, 2018.

[10] Peng Cheng. Sampling-based motion planning with differential constraints. Technical report, 2005.

[11] Paolo De Petris, Stephen J Carlson, Christos Papachristos, and Kostas Alexis. Collision-tolerant aerial robots: A survey. *arXiv:2212.03196*, 2022.

[12] Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification: 22nd International Conference*, 2010.

[13] Eugene Fink and Manuela Veloso. Prodigy planning algorithm. Technical report, Carnegie-Mellon University Pittsburgh PA Dept of Computer Science, 1994.

[14] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:265–293, 2021.

[15] Thomas Gillespie. *Fundamentals of vehicle dynamics*. SAE International, 2021.

[16] Jake Gipple. The volume of n-balls. *Rose-Hulman Undergraduate Mathematics Journal*, 15(1):14, 2014.

[17] Rafal Goebel, Ricardo G Sanfelice, and Andrew R Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28–93, 2009.

[18] Ken Goldberg. Completeness in robot motion planning. In *Workshop on the Algorithmic Foundations of Robotics*, pages 419–429. Citeseer, 1994.

[19] Jessy W Grizzle, Gabriel Abba, and Franck Plestan. Asymptotically stable walking for biped robots: Analysis via systems with impulse effects. *IEEE Transactions on Automatic Control*, 46(1):51–64, 2001.

[20] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[21] Sylvia L Herbert, Mo Chen, SooJean Han, Somil Bansal, Jaime F Fisac, and Claire J Tomlin. Fastrack: A modular framework for fast and guaranteed safe

motion planning. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1517–1522. IEEE, 2017.

[22] Mengxue Hou, Yingke Li, Fumin Zhang, Shreyas Sundaram, and Shaoshuai Mou. An interleaved algorithm for integration of robotic task and motion planning. In *2023 American Control Conference (ACC)*, pages 539–544. IEEE, 2023.

[23] Qiang Huang, Kazuhito Yokoi, Shuuji Kajita, Kenji Kaneko, Hirohiko Arai, Noriho Koyachi, and Kazuo Tanie. Planning walking patterns for a biped robot. *IEEE Transactions on Robotics and Automation*, 17(3):280–289, 2001.

[24] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[25] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the RRT. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483. IEEE, 2011.

[26] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*, pages 396–404. Springer, 1986.

[27] Michal Kleinbort, Kiril Solovey, Zakary Littlefield, Kostas E Bekris, and Dan Halperin. Probabilistic completeness of RRT for geometric and kinodynamic planning with forward propagation. *IEEE Robotics and Automation Letters*, 4(2):x–xvi, 2018.

[28] James J Kuffner, Satoshi Kagami, Koichi Nishiwaki, Masayuki Inaba, and Hirochika

Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, 2002.

[29] James J Kuffner and Steven M LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[30] James J Kuffner, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Footstep planning among obstacles for biped robots. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 1, pages 500–505. IEEE, 2001.

[31] Tobias Kunz and Mike Stilman. Kinodynamic RRTs with fixed time step and best-input extension are not probabilistically complete. In *Algorithmic Foundations of Robotics XI*, pages 233–244. Springer, 2015.

[32] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[33] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[34] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[35] Yanbo Li, Zakary Littlefield, and Kostas E Bekris. Asymptotically optimal sampling-based kinodynamic planning. *The International Journal of Robotics Research*, 35(5):528–564, 2016.

[36] Maxim Likhachev, David I Ferguson, Geoffrey J Gordon, Anthony Stentz, and Sebastian Thrun. Anytime dynamic A*: An anytime, replanning algorithm,. In *ICAPS*, volume 5, pages 262–271, 2005.

[37] Sikang Liu, Nikolay Atanasov, Kartik Mohta, and Vijay Kumar. Search-based motion planning for quadrotors using linear quadratic minimum time control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2872–2879. IEEE, 2017.

[38] Jauwairia Nasir, Fahad Islam, Usman Malik, Yasar Ayaz, Osman Hasan, Mushtaq Khan, and Mannan Saeed Muhammad. RRT*-SMART: A rapid convergence implementation of RRT. *International Journal of Advanced Robotic Systems*, 10(7):299, 2013.

[39] Oren Nechushtan, Barak Raveh, and Dan Halperin. Sampling-diagram automata: A tool for analyzing path quality in tree planners. In *the 9th International Workshop on the Algorithmic Foundations of Robotics*, pages 285–301. Springer, 2010.

[40] Truong Nghiem, Sriram Sankaranarayanan, Georgios Fainekos, Franjo Ivancić, Aarti Gupta, and George J Pappas. Monte-carlo techniques for falsification of temporal

properties of non-linear hybrid systems. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, 2010.

[41] Pedro M Pinheiro, Armando A Neto, Ricardo B Grando, César B da Silva, Vivian M Aoki, Dayana S Cardoso, Alexandre C Horn, and Paulo LJ Drews Jr. Trajectory planning for hybrid unmanned aerial underwater vehicles with smooth media transition. *Journal of Intelligent & Robotic Systems*, 104(3):46, 2022.

[42] Ahmed Hussain Qureshi, Yinglong Miao, Anthony Simeonov, and Michael C Yip. Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics*, 37(1):48–66, 2020.

[43] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[44] John H Reif. Complexity of the mover's problem and generalizations. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 421–427. IEEE Computer Society, 1979.

[45] Tomasz Rybus. Point-to-point motion planning of a free-floating space manipulator using the rapidly-exploring random trees (RRT) method. *Robotica*, 38(6):957–982, 2020.

[46] Ricardo Sanfelice, David Copp, and Pablo Nanez. A toolbox for simulation of hybrid systems in matlab/simulink: Hybrid equations (HyEQ) toolbox. In *Proceedings of*

the 16th International Conference on Hybrid Systems: Computation and Control, pages 101–106, 2013.

[47] Ricardo G Sanfelice. *Hybrid feedback control*. Princeton University Press, 2021.

[48] Brendan E Short and Ricardo G Sanfelice. A hybrid predictive control approach to trajectory tracking for a fully actuated biped. In *2018 Annual American Control Conference (ACC)*, pages 3526–3531. IEEE, 2018.

[49] Mengxuan Song, Nan Wang, Timothy Gordon, and Jun Wang. Flow-field guided steering control for rigid autonomous ground vehicles in low-speed manoeuvring. *Vehicle System Dynamics*, 57(8):1090–1107, 2019.

[50] Mengxuan Song, Nan Wang, Jun Wang, and Timothy Gordon. A fluid dynamics approach to motion control for rigid autonomous ground vehicles. In *Dynamics of Vehicles on Roads and Tracks Vol 1*, pages 347–352. CRC Press, 2017.

[51] Zhitao Song, Linzhu Yue, Guangli Sun, Yihu Ling, Hongshuo Wei, Linhai Gui, and Yun-Hui Liu. An optimal motion planning framework for quadruped jumping. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11366–11373. IEEE, 2022.

[52] Siyu Teng, Xuemin Hu, Peng Deng, Bai Li, Yuchen Li, Yunfeng Ai, Dongsheng Yang, Lingxi Li, Zhe Xuanyuan, Fenghua Zhu, et al. Motion planning for autonomous driving: The state of the art and future perspectives. *IEEE Transactions on Intelligent Vehicles*, 2023.

[53] Prahlad Vadakkepat, Tong Heng Lee, and Liu Xin. Application of evolutionary artificial potential field in robot soccer system. In *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569)*, pages 2781–2785. IEEE, 2001.

[54] Nan Wang, Stefano Di Cairano, and Ricardo Sanfelice. A switched reference governor for high performance trajectory tracking under state and input constraints.

[55] Nan Wang, Stefano Di Cairano, and Ricardo G Sanfelice. A switched reference governor for high performance trajectory tracking control under state and input constraints. In *2024 American Control Conference (ACC)*, pages 4663–4668. IEEE, 2024.

[56] Nan Wang and Ricardo G Sanfelice. A rapidly-exploring random trees motion planning algorithm for hybrid dynamical systems. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 2626–2631. IEEE, 2022.

[57] Nan Wang and Ricardo G Sanfelice. HySST: A stable sparse rapidly-exploring random trees optimal motion planning algorithm for hybrid dynamical systems. *arXiv preprint arXiv:2305.18649*, 2023.

[58] Nan Wang and Ricardo G Sanfelice. HySST: An asymptotically near-optimal motion planning algorithm for hybrid systems. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 2865–2870. IEEE, 2023.

[59] Nan Wang and Ricardo G Sanfelice. HyRRT-Connect: A bidirectional rapidly-

exploring random trees motion planning algorithm for hybrid systems. *IFAC-PapersOnLine*, 2024.

[60] Nan Wang and Ricardo G Sanfelice. HyRRT-Connect: A bidirectional rapidly-exploring random trees motion planning algorithm for hybrid systems. *arXiv preprint arXiv:2403.18413*, 2024.

[61] Nan Wang and Ricardo G Sanfelice. Motion planning for hybrid dynamical systems: Framework, algorithm template, and a sampling-based approach. *arXiv preprint arXiv:2406.01802*, 2024.

[62] Nan Wang and Ricardo G Sanfelice. Motion planning for hybrid dynamical systems: Framework, algorithm template, and a sampling-based approach. *To appear in the International Journal of Robotics Research*, 2025.

[63] Nan Wang, Mengxuan Song, Jun Wang, and Timothy Gordon. A flow-field guided method of path planning for unmanned ground vehicles. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2762–2767. IEEE, 2017.

[64] Albert Wu, Sadra Sadraddini, and Russ Tedrake. R3t: Rapidly-exploring random reachable set tree for optimal kinodynamic planning of nonlinear hybrid systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4245–4251. IEEE, 2020.

[65] Beverly Xu, Nan Wang, and Ricardo Sanfelice. cHyRRT and cHySST: Two motion

planning tools for hybrid dynamical systems. *arXiv preprint arXiv:2411.11812*, 2024.

[66] Jiaming Zha. *Expanding the Operational Environments of UAVs: Design, Control, and Motion Planning for a Tensegrity Aerial Vehicle and an Uncrewed Underwater Aerial Vehicle*. University of California, Berkeley, 2023.

[67] Jiaming Zha and Mark W Mueller. Exploiting collisions for sampling-based multicopter motion planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7943–7949. IEEE, 2021.