Robustly Complete Temporal Logic Control Synthesis for Nonlinear Systems

by

Yinan Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Applied Mathematics

Waterloo, Ontario, Canada, 2019

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner	Mireille E. Broucke Professor, Department of Electrical and Computer Engineering University of Toronto
Supervisor(s)	Jun Liu Associate Professor, Department of Applied Mathematics
	Xinzhi Liu Professor, Department of Applied Mathematics
Internal Member(s)	Sue Ann Campbell Professor, Department of Applied Mathematics
	Sander Rhebergen Assistant Professor, Department of Applied Mathematics
Internal-external Member	Sherman Shen Professor, Department of Electrical and Computer Engineering

Author’s Declaration

Abstract

Modern systems such as spacecrafts and autonomous vehicles are complex yet safety-critical, and therefore the control methods that can deal with different dynamics and constraints while being provably correct are sought after. Formal methods are rigorous techniques originally used for developing and verifying finite-state systems with respect to specifications in formal languages. This thesis is concerned with using formal methods in control synthesis for nonlinear systems, which can guarantee the correctness of the resulting control strategies.

For nonlinear continuous-state dynamical systems, formal control synthesis relies on finite abstractions of the original system by discretizing the system state space and over approximating system transitions. Without further assumptions, control synthesis is usually not complete in the way that no control strategies can be found even if there exists one. To deal with this problem, this thesis proposes a formal control synthesis approach that is *sound and robustly complete* in the sense that correct control strategies can be found whenever the specifications can be realized for the system with additional disturbance.

Fundamental to the soundness and robust completeness is a fixed-point characterization of the *winning set* of the system with respect to a given specification, which is the set of initial conditions that can be controlled to satisfy the specification. Regarding discrete-time systems, such characterizations are first presented by using iterative computation of *predecessors* for basic linear temporal logic (LTL) specifications, including invariance, reachability and reach-and-stay. A more general class of LTL formulas, which can be translated into deterministic Büchi automata (DBA), is also considered, and an algorithm guided by the graph structure of the LTL-equivalent DBA is proposed for characterizing the winning set in this situation. It is then shown that the computational complexity of the algorithm can be reduced by using a pre-processing procedure to the graphs of the DBA.

Because of the general nonlinearity, exact computation of winning sets is currently almost impossible. In this work, the conditions for set approximations are derived so that control synthesis is robustly complete. To meet such conditions, the proposed approach adopts interval arithmetic and a subdivision scheme in the approximation of predecessors. Under such a scheme, the system state space is adaptively partitioned with respect to both the given dynamics and specification and set approximation can be made arbitrarily precise to satisfy the robust completeness conditions. The proposed method is also shown applicable to sampled-data systems by computing validated solutions over one sampling period based on high-order Taylor expansion.

Applications such as converter voltage regulation, parallel parking, and reactive locomotion planning problems are studied to show the effectiveness and efficiency of the proposed approach.

Acknowledgements

I would not have made this thesis possible without the unreserved support from my supervisor Prof. Jun Liu. I cannot forget how you commented and revised my first technical paper word by word, how you showed me what a real mathematical proof is, how you helped me formulate my ideas mathematically, and how you made our every individual meeting a friend chat. You always encourage me to present my research at various conferences. You provide me with as many opportunities as you can to build up my future career path and have written for me in your every reference letter with strong support. Many thanks also to my co-supervisor Prof. Xinzhi Liu. I learned from your classes the systematic method for analyzing dynamical systems. Even short conversations with you sparkled in wisdom of life. I want to say more to express my gratitude to both of you but just feel wordless.

I would also like to thank my other examining committee members Prof. Sue Ann Campbell and Prof. Sander Rhebergen, my internal/external examiner Prof. Sherman Shen, and my external examiner Prof. Mireille Broucke for taking your time reading my thesis and tolerating my wordy statements and proofs. I am very grateful for your comments on improving this work.

Many thanks to my friends and group mates in Waterloo – Chuanzheng, Milad, Luc, Riley, Kevin, Yiming, Mengyao, Jiamu and Zhibing. I spent so much time with you and enjoyed every moment of it. Special thanks to Minxin, my close pal from the same hometown, for accompanying me throughout my first year in Waterloo and offering me help when I was struggling with math. To Ruowei and Tingli, you made my life in Sheffield memorable. Also to my landlady, I cannot count how many times you saved my life by giving me food when I came home starving. Thank you.

The process to earn this title of "doctor of philosophy" turns out to make me think like a philosopher. I started to think about the meaning of my life, which, I finally realized, was to deserve the love of my parents and how they made me the person I am now.

To Mom, Dad, and my childlike curiosity

Table of Contents

List of Figures

List of Tables

Acronyms

BA Büchi Automaton , 89–92, 94

CoM center of mass , 136–141, 145

CSP Constraint-Satisfaction Problem , 55, 56

DAG Directed Acyclic Graph , 92, 111, 119

DBA Deterministic Büchi Automaton , 91–95, 98–104, 107, 108, 111, 112, 114, 117, 119, 120, 125, 128, 155, 156

DE Difference Equation , 5, 24

DT Discrete-Time

FA Finite Automaton , 90, 91

HM hopping mode , 139

iff if and only if

LT Linear Time , 9

LTI Linear Time Invariant , 34, 35, 41

LTL Linear Temporal Logic , 9–15, 17, 19, 20, 28, 30, 31, 39, 45, 46, 48, 50, 52, 64, 65, 69, 73, 89, 90, 92–94, 102–104, 107, 108, 111–113, 117, 119, 120, 125, 129, 132, 134, 135, 141, 143, 144, 154–157, 172

MCIS Maximal controlled invariant set

MCM multi-contact mode , 139, 153, 180

MPC Model Predictive Control , 1, 30

NBA Non-deterministic Büchi Automaton , 91, 92, 155

o.w. Otherwise

ODE Ordinary Differential Equation , 8, 41, 74, 121, 132, 174

PID Proportional-Integral-Derivative , 1

PIPM prismatic inverted pendulum mode , 136, 138, 149–152, 179, 180

PNF Positive Normal Form , 10

PPM prismatic pendulum mode , 138, 150–152, 180

ROA Region of Attraction , 129–131, 157, 178

s.t. such that

SCC Strongly Connected Component , 92, 111, 112, 117

SIVIA Set Inversion Via Interval Analysis , 56, 59

SLM stop-launch mode , 138, 139

SM sliding mode , 140

WBDL whole-body dynamic locomotion , 135, 136, 140, 144, 145

ZOH Zero Order Hold , 121

List of Symbols

$\mathbf{0}^n$ The n-dimensional zero vector.

$\mathbf{1}^n$ The n-dimensional vector with all its elements being 1.

A^* The set of all finite sequences taking values in set A

A^∞ The set of all infinite and finite sequences taking values in set A, i.e., $A^* \cup A^\omega$

A^ω The set of all infinite sequences taking values in set A

$\mathbb{D}$ A set of disturbances, a subset of $\mathbb{R}^n$

$\mathbb{IR}^n$ A set of n-dimensional interval vectors

$\mathbb{N}$ A set of non-negative integers or natural numbers

$\mathbb{R}_{>0}$ A set of positive real numbers

$\mathbb{R}^n$ A set of n-dimensional real vectors

$\mathbb{R}$ A set of real numbers

$\mathbb{U}$ System input space, a subset of $\mathbb{R}^m$

$\mathbb{X}$ System state space, a subset of $\mathbb{R}^n$

$\mathbb{Z}^+$ A set of positive integers

$\mathbb{Z}^m$ A set of integers

$\mathbb{Z}$ A set of m-dimensional integers

$\Box\Diamond$ The temporal operator denoting "always eventually" (or Büchi).

$\Box$ The derived temporal operator denoting “always”, i.e., $\Box\varphi \triangleq \neg\Diamond\neg\varphi$.

$\mathcal{B}_\delta$ An n-dimensional ball with radius δ, i.e., $\mathcal{B}_\delta = \{x \in \mathbb{R}^n : |x| \leq \delta\}$

∂A The boundary of the set A

$|A|$ The cardinal number of set A

$\mathbf{cl}(A)$ The closure of the set A

A^c The complement of the set $A \subseteq \mathbb{R}^n$, i.e., $A^c = \mathbb{R}^n \setminus A$

$g \circ f$ The composition function of functions g and f, which are consistent.

$\Diamond\Box$ The temporal operator denoting “eventually always” (reach-and-stay or co-Büchi).

$\Diamond$ The derived temporal operator denoting “eventually”, i.e., $\Diamond\varphi \triangleq \top\mathbf{U}\varphi$.

$\bot$ A false statement.

$\overset{\circ}{A}$ The interior of the set A

$A \oplus B$ Minkowski sum, i.e., $A \oplus B \triangleq \{a + b \,|\, a \in A, b \in B\}$

$\bigcirc$ The temporal operator denoting “next”.

$\|\cdot\|_2$ The Euclidean norm in $\mathbb{R}^n$ space

$\|\cdot\|_\infty$ The infinity norm in $\mathbb{R}^n$ space

$A \ominus B$ Pontryagin difference, i.e., $A \ominus B \triangleq \{c \in \mathbb{R}^n \mid c + b \in A, \forall b \in B\}$

2^A The power set of set A

$A \setminus B$ The subtraction between set A and B, i.e., $A \setminus B \triangleq \{x \in A : x \notin B\}$

$\top$ A true statement.

$\mathbf{U}$ The temporal operator denoting “until”.

$\models$ Satisfaction relation, $s \models \varphi$ means the formula φ is true at s

Chapter 1

Introduction

1.1 Motivation

Nonlinearity, constraints, and uncertainties are among the critical factors that increase the difficulty of solving practical control problems, which are ubiquitous in this modern world. Most of dynamical systems are nonlinear, and a control system can be as simple as a water heater controller or as sophisticated as a spacecraft control system or the control system for a network of autonomous vehicles.

Because of the various behaviors in different operating domains, nonlinear systems are more difficult to deal with than linear systems, for which control theory is well developed even with the consideration of exogenous disturbances (see [129]). Linear control methods are still applicable to nonlinear systems via linearization, only limited to an unknown neighborhood around a desired state of the nonlinear system. In industry, Proportional-Integral-Derivative (PID) control is the most frequently used nonlinear control method but the problem is that it relies on repetitive and empirical tuning of parameters. One of the systematic nonlinear control methods is Lyapunov-based method such as using control Lyapunov functions, sliding mode control, and passivity-based control [69]. The design of a proper Lyapunov function is rather technical, and state or control constraints are usually not considered in these settings. A renowned control framework of handling constraints is Model Predictive Control (MPC) [88], which has been widely used in process industries [107]. Using MPC, a nonlinear constrained control problem is typically tackled by attempting to solve a series of finite-horizon optimal control problems, to which solutions may not exist. Since these aforementioned control methods in the control literature do not provide correctness guarantee of a controller designed for nonlinear systems under constraints and uncertainties, an *a posteriori* verification is often required to ensure that the control specifications are satisfied.

More recently, there is a rising demand of understanding and control of cyber-physical systems (CPS), which is a new generation of systems with integrated computational and physical capabilities that can interact with humans through many new modalities, such as interoperable medical systems, intelligent transportation systems equipped with autonomous vehicles, and smart grid that are energy efficient. Such systems exhibit both discrete and continuous behaviors. Even for continuous-time dynamical systems, under digital control scheme, continuous time-varying states are sampled and quantized to strings of discrete-time data. For example, programmable logic controllers are used for controlling industrial production such as chemical reaction processes. The complexity of control problems for such systems further increases as richer classes of control objectives (or specifications) are required, not restricted to stabilization or tracking as in the traditional control design. A typical example is the robot motion planning problem [44, 41]. While being subject to mechanical constraints and dynamics, robots are designed to fulfill tasks such as pickup-delivery, parts assembly, surveillance and persistent monitoring. Usually, these tasks have to be completed in specific orders, and robots are required to be reactive to the change of environment.

To reduce the cost of *a posteriori* verification for complex systems as such, we wonder if it is possible to design a *correct-by-construction* approach for control purposes. The idea of *model checking* [30, 8] inspires the use of *formal methods* in control. Formal methods are rigorous techniques and tools for specifying properties, designing and verifying software and hardware systems, and model checking techniques are used to automatically and systematically check whether a given formal property holds for (a given state in) a finite-state model of a system. Such techniques have been used successfully in practice to verify complex sequential circuit designs and communication protocols. In model checking, temporal logic such as linear temporal logic and computational tree logic [8] is often used as a formal description of specifications, and it is shown to be expressive enough to capture control specifications that are used for various control settings such as robot motion planning [76, 41, 136] and automatic cruise control [96].

Under this background, this thesis is concerned with control synthesis for nonlinear systems using formal methods. Particularly, the specifications are given in the form of linear temporal logic formulas. As formal methods originally apply to finite-state systems, dynamical systems need to be discretized so that the existing computer algorithms can be used directly. The challenge lies in the connection of discrete methods to continuous state control and the conditions that algorithmic control synthesis methods can realize control specifications correctly whenever it is possible.

1.2 Thesis Overview

In order to guarantee the correctness of control synthesis for nonlinear systems with respect to temporal logic specifications, most of the methods in the literature work on over approximations of system dynamics, which are the types of system discretizations that cover all the possible behaviors of the original continuous-state systems and probably includes spurious transitions as well, if no discretization that accurately represents the original dynamics can be found. As a result, these methods are conservative in the way that it may not be able to find a control strategy even if there exists one, or in other words, *not complete*. Without any assumptions on the system dynamics or stability properties, making control synthesis for general nonlinear systems with respect to temporal logic formulas sound and complete is nontrivial.

As a main contribution, this thesis proposes *sound and robustly complete* control synthesis algorithms with respect to general classes of linear temporal logic specifications, which are guaranteed to find correct control strategies provided the specifications can be realized for the system with additional disturbances. Furthermore, the proposed algorithms, which are implemented via an interval subdivision scheme, are shown to be more efficient in practice than the abstraction-based methods, which often require a uniform discretization of the system state space.

To illustrate the proposed algorithms with respect to different linear temporal logic formulas and how the robust completeness takes effects in these algorithms, the thesis is organized as follows.

Chapter 2 presents a formal definition of the control synthesis problem under consideration. *Transition systems* are used to connect the linear temporal logic specifications with the behaviors of dynamical systems. As opposed to conventional control problems where the initial conditions are usually given, this research is concerned with finding the set of initial conditions from which a given specification can be satisfied by using some control strategy, which is termed as the *winning set*. We define *robust completeness* for control synthesis algorithms in this chapter since it is a key concept for dealing with nonlinear dynamics and goes through the remaining chapters in the thesis.

In Chapter 3, we provide preliminaries for set-theoretic analysis: the Pontryagin difference and set convergence. For the purpose of winning set computation, we focus on discussing the properties of the *predecessor* map, which is defined as the set of states that can be controlled into a given set in the system state space in one step. These properties are crucial for developing the completeness results for invariance and reachability specifications.

Chapter 4 is devoted to solving control synthesis problems with respect to the most fundamental linear temporal logic specifications, including invariance, reachability and reach-and-stay formulas. We show that these control problems are essentially regulation problems in the control literature and can be solved by sound and complete fixed-point algorithms based on the computation of predecessors. Considering numerical difficulties in computing predecessors under nonlinear dynamics, we propose to use approximations of predecessors under certain conditions so that the algorithms can be made robustly complete.

To make the proposed control synthesis methods in Chapter 4 solid, Chapter 5 focuses on the interval implementation of the proposed control synthesis algorithms, in which predecessors are approximated by unions of intervals. The approximation procedure is carried out by integrating interval arithmetic in a bisection scheme. We will discuss sufficient conditions for the proposed interval implementation to be sound and robustly complete as well as finitely terminating.

Chapter 6 is concerned with a general class of linear temporal logic formulas: the formulas that can be translated into deterministic Büchi automata. The idea of solving such control problems is inline with the proposed method in Chapter 4 and 5. To deal with the generality in the form of the specifications, we perform control synthesis under the guidance of the graph structure of the deterministic Büchi automaton representation of the control specification.

The above results also hold for sampled-data systems, which is the topic of Chapter 7. The behaviors of sampled-data systems are determined by ordinary differential equations but the system state is measured and controlled only at discrete time instances. For sampled-data systems, the *reachable set* from an initial set of states after a sampling time step needs to be evaluated to determine whether a transition between two states is valid. In this chapter, we approximate the reachable set by computing Taylor expansion of the system solution over one sampling period by using interval arithmetic.

In Chapter 8, we demonstrate how the proposed control synthesis method can be applied to solving the reactive locomotion planning problem, where the bipedal robot is required to perform different types of locomotion in response to the changing environment. Because of the complexity in both specifications and dynamics, a hierarchical control design is usually used. The proposed control method will be used to verify the correctness of high-level plan as well as to generate a middle-level strategy that synergizes the high-level plan and low-level controllers.

Furthermore, we introduce a self-developed tool ROCS in the appendix, which is used to conduct all the numerical experiments in this thesis.

Chapter 2

Problem Formulation

Applying formal methods in control requires proper system models and control specifications that can be understood by computer algorithms. Discrete-time discrete-state systems, which can be modeled by finite state-transition graphs, such as *Markov decision process* [12] that are used in dynamic programming for optimal control problems and *Kripke structure* for model checking [30] are favorable, because graph searching algorithms on such systems are more likely to terminate in a finite number of iterations. Limited by the nature of digital computers, real-world continuous-time dynamical systems are only observed at certain discrete time instances, which motivates the study of sampled-data systems. However, the state space of most of the physical systems is continuous and thus contains an infinite number of states.

This chapter is devoted to a formal statement of the control synthesis problem, in which the control objective is given as *linear temporal logic* formulas and dynamical systems are formulated as *transition systems* that incorporate basic elements of temporal logic.

As opposed to conventional control problems where the possibility of the fulfillment of the given control objectives is not considered or discussed, the control synthesis problem formulated in this chapter also explores all the initial conditions from which the control specification can be achieved by proper control strategies. Based on such a formulation, two provably-correct control synthesis approaches are briefly introduced.

2.1 Control System

Consider the following discrete-time nonlinear system given by Difference Equations (DEs):

$$x_{t+1} = f(x_t, u_t), \tag{2.1}$$

where $t \in \mathbb{N}$ is the time instance, $x_t \in \mathbb{X} \subseteq \mathbb{R}^n$ is the state, $u_t \in \mathbb{U} \subseteq \mathbb{R}^m$ is the control input, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a function that determines the system state evolution. The sets $\mathbb{X}$ and $\mathbb{U}$ are the state and control spaces of (2.1), respectively.

Practical feedback control systems are often subject to imperfections in multiple aspects of the control structure. Measurements are corrupted by noise. Delay happens in transferring measured data from sensors to controllers and also from controllers to plants. In sampled-data systems, numerical errors are inevitable during quantization. From a robust control perspective, we hope that the controller designed for the nominal system (2.1) still functions in the presence of uncertainties. In the following, we assume additive bounded disturbances:

$$x_{t+1} = f(x_t, u_t) + d_t, \tag{2.2}$$

where $d_t \in \mathbb{D} \subseteq \mathbb{R}^n$ is a unknown but bounded disturbance, and

$$\mathbb{D} \triangleq \{d \in \mathbb{R}^n : \|d\|_\infty \leq \delta, \, \delta \geq 0\}. \tag{2.3}$$

This is without loss of generality since most of the physical systems evolve continuously over a bounded domain so that the uncertainty of the state change is still within some bound around the nominal value. Clearly, the disturbed form (2.2) reduces to the nominal form (2.1) when $\delta = 0$. For the sake of simplicity, we refer to (2.2) in the rest of the thesis.

2.1.1 Transition System

Transition systems, whose behaviors are determined by state transition relations, are usually used for modeling software and hardware systems. Typically in computer science, transition systems contain finite numbers of states and inputs so that desired properties can be verified or synthesized by running computer programs.

Definition 2.1 ([8]). A *transition system* is defined as a tuple $\mathcal{T} : \langle S, \, Act, \, R, \, AP, \, L \rangle$, where

- S is a set of states;
- Act is a set of actions;
- $R : S \times Act \to 2^S$ is a transition function;
- AP is a set of atomic propositions;
- $L : S \to 2^{AP}$ is a labeling function.

Atomic propositions are true or false statements about system state properties. For example, statements such as "s is between 7 and 20"(s is a variable) and "all birds can fly" can be considered as atomic propositions. If $7 < s < 20$, then the first atomic proposition is true; otherwise it is false. The *labeling function* L assigns each state $s \in S$ a set of atomic propositions, i.e., $L(s) \in 2^{AP}$.

A transition system is said to be *finite* if the set S, Act, and AP are all finite. Given any state $s \in S$ and any control action $a \in Act$, if there is only one state in $R(s, a)$, then the system is *deterministic*, otherwise it is *non-deterministic*.

2.1.2 Control System as Transition System

System (2.2) can be translated to an equivalent transition system

$$\mathcal{S} : \langle \mathbb{X},\ \mathbb{U},\ R,\ AP,\ L \rangle, \tag{2.4}$$

where $\mathbb{X}$ and $\mathbb{U}$ are the set of states and inputs, respectively, and the transition relation R is defined by $R(x, u) \triangleq \{f(x, u) + d :\ d \in \mathbb{D}\}$ for all $x \in \mathbb{X}$ and $u \in \mathbb{U}$.

For discrete-time dynamical systems over continuous state and input spaces, the sets $\mathbb{X}$ and $\mathbb{U}$ are infinite. There is a transition from x to x' whenever there exists $u \in \mathbb{U}$ such that $x' \in R(x, u)$. Therefore, system $\mathcal{S}$ is infinite and non-deterministic if $\delta \neq 0$. Specifically, we refer to $\mathcal{S}^0$ as the nominal control system (2.1) since $\mathcal{S}$ reduces to (2.1) when $\delta = 0$, which is deterministic.

An infinite sequence of control inputs $\mathbf{u} = \{u_t\}_{t=0}^{\infty}$ ($u_t \in \mathbb{U}$ for all $t \in \mathbb{N}$) is called a *control signal*, and a sequence of disturbance, is denoted by $\mathbf{d} = \{d_t\}_{t=0}^{\infty}$.

Definition 2.2. Given an initial condition $x \in \mathbb{X}$ and a control signal $\mathbf{u}$, a *solution* of $\mathcal{S}$ is an infinite sequence of states $\mathbf{x} = \{x_t\}_{t=0}^{\infty}$ generated by the transition relation R, i.e., $x_0 = x$ and $x_{t+1} \in R(x_t, u_t)$ for all $t \in \mathbb{N}$.

Let Σ be an *alphabet*, and each element of Σ is called a *letter*. A sequence of letters from an alphabet Σ is called a *word*. The word $\mathbf{w}$ is *infinite* if it is an infinite sequence. Given an infinite word $\mathbf{w} = \sigma_0\sigma_1\cdots$, a finite sequence composed of the first i elements of $\mathbf{w}$, i.e., $\sigma_0\cdots\sigma_i$ ($i \in \mathbb{N}$) is called a *prefix* of $\mathbf{w}$, and an infinite sequence $\sigma_i\sigma_{i+1}\cdots$ ($i \in \mathbb{N}$) is called a *suffix* of $\mathbf{w}$.

Definition 2.3. The *trace* of a solution $\mathbf{x} = \{x_t\}_{t=0}^{\infty}$ of system $\mathcal{S}$ is an infinite word $\text{Trace}(\mathbf{x}) = \{L(x_t)\}_{t=0}^{\infty}$ over the power set 2^{AP} of the set AP of atomic propositions.

Interpreting system solutions by their traces, a property expressed by atomic propositions can be verified for system $\mathcal{S}$ or used to guide the synthesis of a controller.

There are usually a finite number of atomic propositions while the number of states in the state space $\mathbb{X}$ is infinite. Now the question comes to the design of the labeling function so that the set AP and $\mathbb{X}$ are properly mapped to each other.

Definition 2.4. Given a set $\Omega \subseteq \mathbb{R}^n$ and a positive integer N, a finite collection of sets $\mathcal{P} = \{P_1, P_2, \cdots, P_N\}$ is said to be a *partition* of Ω if

(i) $P_i \subseteq \Omega$, for all $i \in \{1, \cdots, N\}$;

(ii) $\overset{\circ}{P}_i \cap \overset{\circ}{P}_j = \varnothing$ for all $i, j \in \{1, \cdots, N\}$ where $\overset{\circ}{P}_i$ denotes the interior of set P_i;

(iii) $\Omega \subseteq \bigcup_{i=1}^{N} P_i$.

Each element P_i of the partition $\mathcal{P}$ is called a *cell*.

Let $\{\gamma_1, \cdots, \gamma_N\}$, where $\gamma_i \in 2^{AP}$ is a subset of the set AP of atomic propositions and

$$\bigvee_{i=1}^{N} \gamma_i = \top, \quad \gamma_i \wedge \gamma_j = \bot, \, i \neq j, \tag{2.5}$$

where $\top$ and $\bot$ means *true* and *false*, respectively. Then we can obtain a partition $\mathcal{P}_0 = \{P_1, P_2, \cdots, P_N\}$ of the state space $\mathbb{X}$, where

$$P_i \triangleq L^{-1}(\gamma_i) = \{x \in \mathbb{X} : \, L(x) = \gamma_i\}\,. \tag{2.6}$$

This is to say that all the states inside a cell are assigned the same atomic proposition. Additionally, $L^{-1}(\top) = \mathbb{X}$.

Let us illustrate by the following example how the set of atomic propositions to be designed and equipped to system $\mathcal{S}$ for specifying desired properties.

Example 2.1. The adaptive cruise control system for a single vehicle is modeled by the following Ordinary Differential Equations (ODEs) [96]:

$$\begin{aligned} \dot{v} &= F_w/m - f_0 - f_1 v - f_2 v^2, \\ \dot{h} &= v_L - v, \end{aligned}$$

where F_w is the control input, h is the headway, v_L and v are the velocity of the leading and following vehicle, respectively, f_0, f_1, and f_2 are real constants.

There are two modes determined by the *time headway* $w \triangleq h/v$: the *set speed* mode $M_1 = \{(v,h) : v \leq h/w_d\}$ and the *time gap* mode $M_2 = \{(v,h) : v > h/w_d\}$, where w_d denotes the desired time headway. Then the (h,v) space is partitioned into two cells by using $AP = \{\mathrm{set}, \mathrm{gap}\}$ and the labeling function

$$L(h,v) = \begin{cases} \mathrm{set} & (v,h) \in M_1, \\ \mathrm{gap} & (v,h) \in M_2. \end{cases}$$

2.2 Linear Temporal Logic

Properties of a transition system are usually evaluated over its traces, which, as defined in Definition 2.3, evolve over time. In this sense, such properties are called Linear Time (LT) properties. To verify or synthesize an LT property for a transition system through algorithmic computation, it is crucial to describe LT properties in a way that can be operated by computer programs.

Linear Temporal Logic (LTL) is a logical formalism defined over an alphabet 2^{AP}, which can specify LT properties. An LTL formula consists of *propositional logic* operators (e.g., true ($\top$), negation ($\neg$), and conjunction ($\wedge$)), and *temporal operators* (e.g., next ($\bigcirc$) and until ($\mathbf{U}$)). The syntax of LTL over AP is defined in the Backus Naur Form ($p \in AP$):

$$\varphi ::= \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \mathbf{U}\varphi,$$

which reads inductively as

- $\varphi = \top$ is an LTL formula;
- $\varphi = p \in AP$ is an LTL formula;
- if φ, φ_1, and φ_2 are LTL formulas, then $\neg\varphi$, $\varphi_1 \wedge \varphi_2$, $\bigcirc\varphi$, and $\varphi_1 \mathbf{U} \varphi_2$ are LTL formulas.

Based on these basic operators, several other important temporal operators can also be defined. For example,

$$\varphi_1 \vee \varphi_2 \triangleq \neg(\neg\varphi_1 \wedge \neg\varphi_2), \quad \varphi_1 \rightarrow \varphi_2 \triangleq \neg\varphi_1 \vee \varphi_2,$$
$$\Diamond\varphi \triangleq \top\mathbf{U}\varphi, \quad \Box\varphi \triangleq \neg\Diamond\neg\varphi.$$

If negations are only allowed to appear adjacent to atomic propositions, which is the so-called Positive Normal Form (PNF), false ($\bot$), an additional temporal operator *release* ($\mathbf{R}$), and a propositional operator *disjunction* ($\vee$) need to be used to transform any LTL formula into PNF:

$$\varphi ::= \top \mid \bot \mid p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \mathbf{U}\varphi \mid \varphi_1\mathbf{R}\varphi_2$$

The semantics of LTL is defined with respect to a transition system. Given a word $\sigma = \sigma_0\sigma_1\sigma_2\cdots$ over 2^{AP}, let $\sigma[i] \triangleq \sigma_i$, $\sigma[i,j] \triangleq \sigma_i\cdots\sigma_j$, and $\sigma[i,\cdots] \triangleq \sigma_i\cdots$. We define $\sigma, i \models \varphi$, meaning that σ satisfies an LTL formula φ at position i, inductively as follows:

- $\sigma, i \models \top$ iff $\sigma_i = \top$;
- $\sigma, i \models p$ iff $\sigma_i \models a$;
- $\sigma, i \models \neg\varphi$ iff $\sigma, i \not\models \varphi$;
- $\sigma, i \models \varphi_1 \wedge \varphi_2$ iff $\sigma, i \models \varphi_1$ and $\sigma, i \models \varphi_2$;
- $\sigma, i \models \varphi_1 \vee \varphi_2$ iff $\sigma, i \models \varphi_1$ or $\sigma, i \models \varphi_2$;
- $\sigma, i \models \bigcirc\varphi$ iff $\sigma, i+1 \models \varphi$;
- $\sigma, i \models \varphi_1\mathbf{U}\varphi_2$ iff there exists $j \geq i \geq 0$ such that $\sigma, j \models \varphi_2$ and $\sigma, k \models \varphi_1$ for all $i \leq k < j$;
- $\rho, i \models \varphi_1\mathbf{R}\varphi_2$ iff, for all $j \geq i$, at least one of the following holds: $\sigma, j \models \varphi_2$ or there exists $i \leq k < j$ such that $\sigma, k \models \varphi_1$.

We write $\sigma \models \varphi$ if $\sigma, 0 \models \varphi$ and say σ *satisfies* φ. The set of words satisfying an LTL formula φ is called the *language* of φ, denoted by $\mathcal{L}(\varphi)$.

The semantics for the derived operators $\Diamond$ and $\Box$, the followings can be derived:

- $\sigma \models \Diamond\varphi$ iff $\exists i \geq 0$ s.t. $\sigma[i\cdots] \models \varphi$.
- $\sigma \models \Box\varphi$ iff $\forall i \geq 0$ s.t. $\sigma[i\cdots] \models \varphi$.
- $\sigma \models \Diamond\Box\varphi$ iff $\exists i \geq 0$ s.t. $\sigma[j\cdots] \models \varphi,\ \forall j \geq i$.
- $\sigma \models \Box\Diamond\varphi$ iff $\forall i \geq 0$ s.t. $\sigma[j\cdots] \models \varphi,\ \exists j \geq i$.

time	0	1	2	3	$\cdots$
$\bigcirc\varphi$	*	φ	*	*	$\cdots$
$\varphi_1\mathbf{U}\varphi_2$	φ_1	φ_1	φ_2	*	$\cdots$
$\varphi_1\mathbf{R}\varphi_2$	φ_2	φ_2	φ_1	*	$\cdots$
$\varphi_1\mathbf{R}\varphi_2$	φ_2	φ_2	φ_2	φ_2	$\cdots$
$\Box\varphi$	φ	φ	φ	φ	$\cdots$
$\Diamond\varphi$	*	*	φ	*	$\cdots$
$\Diamond\Box\varphi$	*	*	φ	φ	$\cdots$
$\Box\Diamond\varphi$	*	φ	*	φ	$\cdots$

Figure 2.1: Illustration of the semantics of temporal operators. The formula φ, φ_1 or φ_2 showing at a certain time instance $i \in \mathbb{N}$ means that φ, φ_1 or φ_2 is true at i. The star marker denotes any formula or proposition. For $\Box\Diamond\varphi$, the formula φ is true for infinitely many times.

As interpreted above, the temporal operators $\Box\Diamond$ and $\Diamond\Box$ intuitively express the properties "infinitely often" and "eventually forever". The LTL formula $\Box\Diamond\varphi$ means that φ hold true for infinitely many times, and $\Diamond\Box\varphi$ means that φ will be always true from some time instance. An intuitive illustration of the above semantics is given in Figure 2.1.

Example 2.2. Suppose that $AP = \{a, b, c\}$. Then

$$\begin{aligned} \varphi_1 &= (\bigcirc a)\mathbf{U}(a \wedge \neg b), \\ \varphi_2 &= \Box(\neg a \vee \neg c), \\ \varphi_3 &= \Box\Diamond b \to \Box\Diamond c \end{aligned}$$

are all LTL formulas. Even for a specific LTL formula φ, the words that match φ can be different. Figure 2.2 shows the words described by φ_1, φ_2, and φ_3. The formula φ_3 expresses a *fairness* property, which pictures that "infinitely often requests must be responded infinitely many times".

LTL formulas are expressive enough to capture *safety*, *liveness*, and *fairness* properties. Safety properties rule out forbidden behaviors which would cause damage to the system. liveness properties focus on infinite behaviors and impose no confinement to any finite behaviors. Fairness properties restrict the system behaviors in response to environment changes. In a strong sense, we hope that any request to a system constantly should be answered infinitely often, which often help distribute the resources.

time:	0	1	2	3	$\cdots$
φ_1:	*	a	*	$a \wedge \neg b$	$\cdots$
φ_1:	*	a	$a \wedge \neg b$	*	$\cdots$
φ_1:	$a \wedge \neg b$	*	*	*	$\cdots$
φ_2:	$\neg a$	$\neg a$	$\neg c$	$\neg a$	$\cdots$
φ_2:	$\neg c$	$\neg c$	$\neg c$	$\neg c$	$\cdots$
φ_3:	b	b	$\neg c$	c	$\cdots$

Figure 2.2: Example words specified by formulas φ_1, φ_2, and φ_3 in Example 2.2. Whether formulas φ_2 and φ_3 hold or not can not be told by finite parts of a word.

Here are some of the real-world specifications that can be expressed by LTL formulas [8, 38].

Example 2.3. The control logic for an elevator must satisfy the safety property: the door must only open when the elevator is at some floor and the liveness property: any floor can be reached eventually if there is a request.

Let $\{f_i\}_{i=1}^{l}$ and $\{b_i\}_{i=1}^{l}$ be the set of atomic propositions indicating the position of the elevator and the activation status of floor buttons, respectively. The elevator is at ith floor iff $f_i = \top$ and $b_i = \top$ iff the button of ith floor is pressed down ($i = 1, \cdots, l$). Then the safety and liveness properties can be expressed by

$$\varphi_{\text{safety}} = \neg\text{open} \wedge \bigwedge_{i=1}^{l} \neg f_i,$$

$$\varphi_{\text{liveness}} = \bigwedge_{i=1}^{l} \Box(b_i \to \Diamond f_i).$$

Example 2.4. The interpretation of the desired behavior of traffic lights can be from different perspectives. The order of activating lights of different colors can be expressed by

$$\Box(\text{green} \to \bigcirc\text{yellow}) \wedge \Box((\text{yellow} \to \bigcirc\text{red}) \wedge \Box(\text{red} \to \bigcirc\text{green}).$$

The long-term behavior of a single light, e.g. the red light should be on infinitely often, can be expressed by an LTL formula:

$$\Box\Diamond\text{red} \wedge \Box\Diamond\text{yellow} \wedge \Box\Diamond\text{green}.$$

The requirement that the yellow light must be lit after the red light and before the green light is expressed by

$$\text{red}\,\mathbf{U}(\text{yellow} \wedge \bigcirc(\text{yellow}\,\mathbf{U}\,\text{green})).$$

2.3 Control Synthesis Problem

The goal is to find a control strategy such that the traces of resulting sequences of system states satisfies a given LTL formula. Prior to presenting the formal definition of the LTL control synthesis problem, we rely on the following definitions.

Definition 2.5. A *control strategy* of system $\mathcal{S}$ is a partial function that maps a history of system state to a set of control inputs:

$$\kappa : \mathbb{X}^* \to 2^{\mathbb{U}}, \tag{2.7}$$

where $\mathbb{X}^*$ denotes the set of all finite sequences taking values from the set $\mathbb{X}$. A control signal $\mathbf{u}$ is said to *conform to* a control strategy κ, if

$$u_t \in \kappa(\{x_0, \cdots, x_t\}), \quad \forall t \in \mathbb{N}.$$

For many of the control problems for system $\mathcal{S}$, however, remembering past system state is not necessary. The control strategy can be simplified to a function with the state space $\mathbb{X}$ as its domain.

Definition 2.6. A control strategy κ is called *memoryless* if it only takes in the current state as the input, i.e.,

$$\kappa : \mathbb{X} \to 2^{\mathbb{U}}. \tag{2.8}$$

Definition 2.7. An LTL formula φ is said to be *realizable* for system $\mathcal{S}$ if there exists an initial condition $x \in \mathbb{X}$ and a control strategy κ such that the trace of any solution for system $\mathcal{S}$ under any control signal $\mathbf{u}$ that conforms to κ is guaranteed to satisfy φ, i.e., $\text{Trace}(\mathbf{x}) \models \varphi$ for all $\mathbf{x} = \{x_t\}_{t=0}^{\infty}$ with $x_0 = x$. We say κ *realizes* φ for system $\mathcal{S}$ at x.

Now we are in the position to present the control synthesis problem with respect to LTL specifications:

Problem 2.1 (LTL Control Synthesis Problem). Consider system $\mathcal{S}$ and an LTL formula φ.

(i) Determine whether φ is realizable for $\mathcal{S}$;

(ii) Synthesize a control strategy κ such that the closed-loop system satisfies φ if possible.

To check the realizability of the LTL formula φ, we need the following definition.

Definition 2.8. The set of initial conditions of all the solutions of system $\mathcal{S}$ whose trace satisfies φ is called the *winning set* of system $\mathcal{S}$ with respect to φ, written as $\mathrm{Win}^{\delta}_{\mathcal{S}}(\varphi)$. Specifically, the winning set for $\mathcal{S}^0$ is denoted by $\mathrm{Win}_{\mathcal{S}}(\varphi)$.

If $\mathrm{Win}^{\delta}_{\mathcal{S}}(\varphi) \neq \varnothing$, then φ can be realized for system $\mathcal{S}$. Ideally, we hope to construct a control strategy that correctly realizes the given specification as long as there exists one. In other words, the control strategy we aim at should be well defined on the winning set $\mathrm{Win}^{\delta}_{\mathcal{S}}(\varphi)$ in the first place. This naturally motivates our intention of seeking *sound and complete* control synthesis methods, which is defined below.

Definition 2.9. Control synthesis for system $\mathcal{S}$ with respect to a given LTL formula φ is said to be *sound* if the resulting control strategies realize φ. It is *complete* if a control strategy can be found for all initial state $x_0 \in \mathrm{Win}_{\mathcal{S}}(\varphi)$.

Determination of the winning set $\mathrm{Win}^{\delta}_{\mathcal{S}}(\varphi)$ for nonlinear systems by analytical analysis is challenging as the controllability analysis of nonlinear systems is nontrivial and often relies on additional assumptions such as the function f in (2.2) being invertible [63]. A more feasible solution is to algorithmically compute $\mathrm{Win}^{\delta}_{\mathcal{S}}(\varphi)$. The accurate computation, however, is usually impossible because of the inevitable numerical error and quantization of measurements. Hence, it is more practical to relax the control synthesis problem. Our relaxation is based on the following definition of robust realizability of a specification.

Definition 2.10. An LTL specification φ is said to be *δ-robustly realizable* for system $\mathcal{S}^0$ if it is realizable for $\mathcal{S}^{\delta}$. If $\delta > 0$, then φ is called *robustly realizable* for $\mathcal{S}^0$.

Problem 2.2 (Relaxed LTL Control Synthesis Problem). Consider system $\mathcal{S}^0$ and an LTL formula φ. Solve one of the two following problems:

(i) Construct a control strategy if φ is robustly realizable for $\mathcal{S}^0$.

(ii) Verify that φ is not realizable for $\mathcal{S}^{\delta}$ with some $\delta > 0$.

For a numerical method that solves Problem 2.2, it is foremost that computation (or approximation) of the winning set is guaranteed to stop in finite time.

Definition 2.11. An algorithm is said to be *finitely terminating* if it terminates in a finite number of steps.

The possibility of finding sound and complete control synthesis methods is also questioned on account of the situations where approximations of winning sets and numerical errors are inevitable. Therefore, for the relaxed Problem 2.2, we propose the following concept to relax the completeness requirement to a control synthesis method.

Definition 2.12 (Robust Completeness)**.** Control synthesis for system $\mathcal{S}$ with respect to a given LTL formula φ is said to be *robustly complete* if a control strategy that realizes φ can be constructed whenever φ is robustly realizable for system $\mathcal{S}$.

The conditions that guarantee the robust completeness of control synthesis algorithms for different LTL specifications will be derived in the following chapters. Before we dive into details, let us review the common approaches to such a control synthesis problem.

2.4 Overview of Control Synthesis Approach

The LTL control synthesis problem is hybrid by definition: the control specification is expressed in a logical language while the system evolves on a continuous state space. As we have mentioned before, a systematic and analytical approach is difficult, and the approach for handling both continuous dynamics and discrete specifications for LTL control synthesis in the literature is built up on discretizing system dynamics. In this way, the control problem can be solved in a discrete domain by automated algorithmic computation.

2.4.1 Abstraction-Based Control

Abstraction-based control, which is also termed as *symbolic* control [121], relies on discrete abstractions of the original systems. The continuous state space is often partitioned into a finite number of regions. All states that belong to such a region are represented by one abstract state. Based on this finite partition, the continuous system evolution is also replaced by transitions between the finite states of a discrete abstraction. Owing to such discrete representation, abstraction-based control is named symbolic control in some of the works in the literature.

Abstraction-based control procedure primarily consists of three steps (see Figure 2.3) [10]:

S1 Construct a finite abstraction for a dynamical system by abstracting state and control space as well as transitions.

S2 Synthesize a discrete controller that satisfies the given specifications over the finite abstraction if there exists one, otherwise returns empty. Such a discrete control synthesis is usually carried out by graph searching algorithms (e.g., Dijkstra algorithm) [33] or the algorithms for solving two-player infinite games [124] over a product system of the abstraction and the specification. Depending on different specifications, the algorithms can be simplified. Details about control synthesis with respect to general LTL formulas will be discussed in Chapter 6.

S3 Translate the discrete controller into a continuous one that solves the original control synthesis problem.

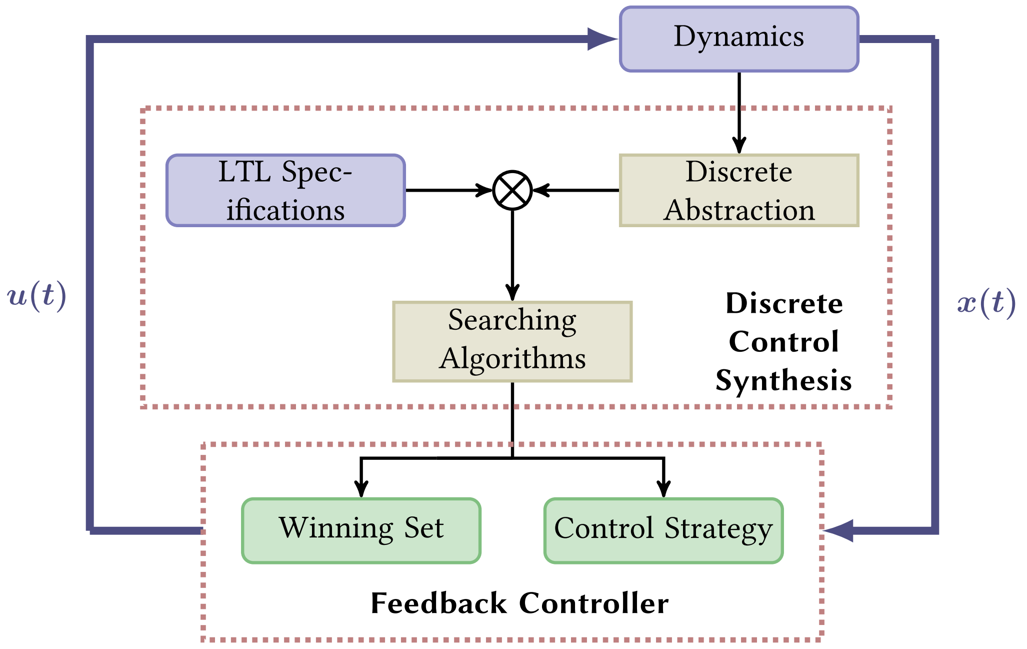


Figure 2.3: Abstraction-based control framework.

The notion of abstractions of continuous-state systems first appears in [101] to reduce the complexity of analyzing properties of complex dynamical systems. Such abstractions are *bisimulations* [2, 1, 60] of the original systems, which have coarser state partitions while maintaining equivalence in terms of the properties being concerned [60]. The *approximate bisimulation* is later proposed in [20] for reachability verification of a class of hybrid systems, which extends the class of the systems that have (approximate) bisimulations. To reduce the complexity in control synthesis, the design of a discrete controller based on a *dynamically consistent (DC) partition machine* of a nonlinear continuous-state system is proposed in [24, 25]. Similar to an abstraction, a DC partition machine of a dynamical system is a finite input-state machine defined on a finite partition of the continuous state space, and the transitions between partition cells are consistent with the original dynamics. The idea of using abstractions introduces algorithmic procedures for the verification and synthesis of pure discrete-state systems to continuous-state systems.

As shown in [100, 57], a *bisimilar transition system* can be constructed by using a linear quotient map that preserves the observation of the linear control system if the kernel of the quotient map is a controlled invariant subspace inside the kernel of the observation map. Such a quotient map can be found for any discrete-time controllable linear systems [122]. Based on bisimilar transition systems of linear systems, the symbolic control approach is then proposed to solve LTL control synthesis problems for linear systems [123, 120, 121]. Specifically, a type of abstractions based on simplices is studied to solve reach control [56, 21, 22] and LTL control [72] problems for linear affine systems. A bisimilar abstraction for nonlinear systems might require the properties of *hybrid between-block controllable (HHBC)* and *hybrid in-block controllable (HIBC)*. If HHBC or HIBC are not satisfied, one way to extend the symbolic control approach to nonlinear systems is via the *approximate bisimulation relation*, which is introduced in [52, 51, 50] and applied to the nonlinear [105, 106] and switched systems [53] that are incrementally stable [4]. To further relax the stability constraints posed to the system dynamics, *approximate simulation relation* is then used in [119, 139] to construct finite abstractions.

In order to be sound and complete in control synthesis, abstractions that are (approximately) equivalent to the original systems is usually needed, which is shown feasible for incrementally stable systems [105, 53]. Without such a stability assumption, we can still construct over-approximations [139, 86, 87, 113], but it does not always guarantee a feasible control strategy, even if one exists, because spurious transitions are introduced and control synthesis is separated from abstraction. Using sufficiently small granularities, approximately complete control synthesis can be achieved without stability assumptions [85] but it is at the cost of intractable computation.

As a summary, the current stage of abstraction-based control synthesis suffers from expensive computation in order to be (robustly) complete.

2.4.2 Specification-guided control

In contrast with the abstraction-based approach, a *specification-guided* approach performs control synthesis directly on the original system with respect to a given linear temporal logic specification. Construction of discrete abstractions is avoided as it is often unnecessary to explore the entire state space for a specific control objective. As a result of direct control synthesis, a specification-guided approach is more efficient in comparison with the abstraction-based approach when it comes down to a specific control problem. The framework of this approach is given in Figure 2.4.

Central to a specification-guided approach are the translations of LTL specifications to fixed-point forms and computational mechanisms performing fixed-point iterations. Treating disturbances or the nondeterminism in a control system as an adversary player, an LTL control synthesis problem can be formulated in the two-player game setting, where the winning strategies

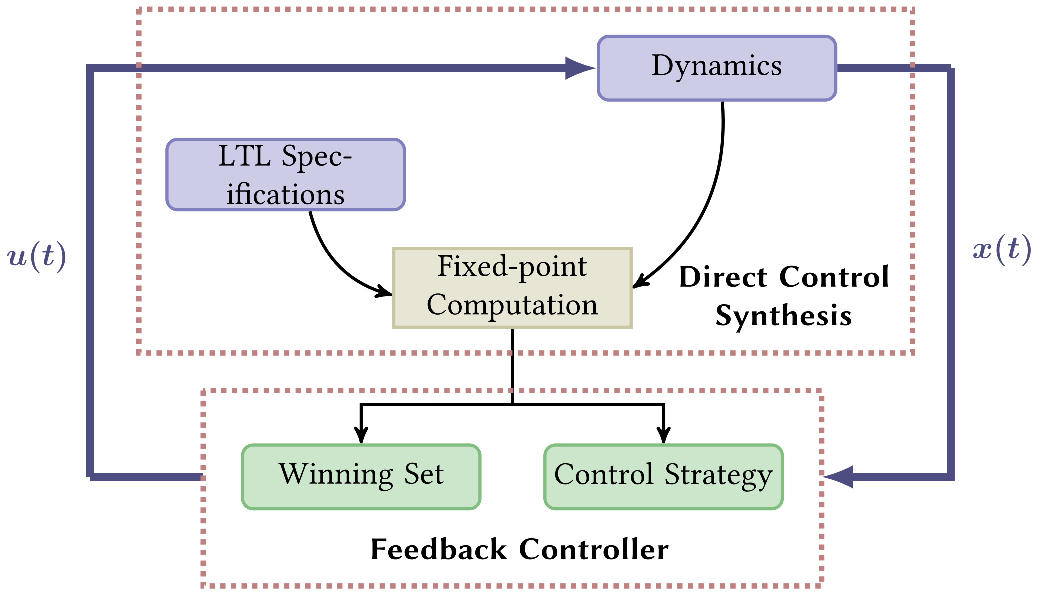


Figure 2.4: Specification-guided control synthesis framework.

for the players can be solved by fixed-point algorithms [23, 40, 89]. For transition systems, some of the properties can be expressed by modal μ-calculus formulas, which are based on least and greatest fixed-point operators [75, 39]. Many of the fixed-point schemes for two-player games, such as reachability and Büchi games, have their equivalent μ-calculus versions, and every ω-regular objective can be formulated by a μ-calculus formula [35].

In this thesis, the fixed-point algorithms in the specification-guided framework is an extension of the ones for transition systems to the control systems defined on continuous state spaces. In this context, a *fixed point* refers to a set of states of the given system $\mathcal{S}$, which is mapped to itself by a set operator. The control synthesis procedure for most of the important control objectives can be viewed as iterative fixed-point computation. For example, to realize controlled invariance, i.e., to control the system states inside a given target set for all future time, the first and foremost step is to determine the maximal controlled invariant set within the target set. A set operation, which takes in a set Ω and computes the set of states inside Ω that can also be controlled inside Ω for one step of time, is performed repeatedly until a fixed point is reached. This fixed point is the maximal controlled invariant set, and also the winning set of the invariance control specification for system $\mathcal{S}$. The way to compute fixed points depends heavily on set representation and system dynamics. Polyhedral and ellipsoidal representations are most studied because they are either natural descriptions of physical constraints or efficient for set computation.

In the scope of a specification-guided approach, most of the research progress is found in safety specification synthesis for discrete-time linear systems [7, 116, 97]. Very few attention has been paid to more general LTL formulas. Besides, the systems under investigation are often linear since computing exact fixed points is nontrivial even for linear systems. This is primarily because of the lack of finite termination guarantee in the fixed-point algorithm. For nonlinear systems, another challenge rises in fixed-point set computation according to nonlinear dynamics.

The specification-guided framework is the one we take to tackle the LTL control synthesis problem throughout the thesis for the purpose of lower computational expense. We will discuss in detail in the following chapters how this framework can be carried out for different levels of LTL specifications and how the control synthesis can be made sound and robustly complete for general nonlinear systems.

Chapter 3

Preliminaries for Set-Theoretic Control Synthesis

Checking the emptiness of a winning set of system $\mathcal{S}$ with respect to an LTL formula is the key to solving the LTL control synthesis problem. This chapter provides the preliminaries for set-theoretic analysis used in the rest of the thesis, especially the *predecessor* map and its related properties.

3.1 The Pontryagin Difference

The Minkowski sum and Pontryagin difference are often used in set relationships. Given two sets $A, B \subseteq \mathbb{R}^n$, the Minkowski sum $A \oplus B$ and Pontryagin difference $A \ominus B$ are defined by

$$A \oplus B \triangleq \{a + b \,|\, a \in A, b \in B\}. \tag{3.1}$$

$$A \ominus B \triangleq \{c \in \mathbb{R}^n \,|\, c + b \in A, \forall b \in B\}. \tag{3.2}$$

The following properties are important for solving LTL control synthesis, and we provide the complete proof of these properties below in order to be self-contained, although part of the proof can be found in [73].

Proposition 3.1. Let $A, B \subseteq \mathbb{R}^n$ and assume that $A \ominus B \neq \emptyset$. Then the following properties hold.

(i) $A \ominus B \subseteq A$ if $\mathbf{0}^n \in B$.

(ii) $A \ominus B \oplus B \subseteq A \subseteq A \oplus B \ominus B$.

(iii) $A \ominus B = (A_1 \ominus B) \cap (A_2 \ominus B), \quad A = A_1 \cap A_2,\ A_1, A_2 \subseteq \mathbb{R}^n$.

(iv) $A \ominus B = \bigcap_{i=1}^{\infty}(A_i \ominus B)$, where $A = \bigcap_{i=1}^{\infty} A_i$.

(v) $A \ominus B$ is closed (compact) if A is closed (compact).

Proof. Property (i) is straightforward since for all $a \in A \ominus B$, $\mathbf{0}^n \in B$ implies that $a + \mathbf{0}^n \in A$.

To show property (ii), let $z \in A \ominus B \oplus B$. Then we can find $y \in A \ominus B$ and $b \in B$ such that $z = y + b$ by (3.1). By (3.2), $y + b \in A$, which gives that $z \in A$. Hence $A \ominus B \oplus B \subseteq A$. Let $a \in A$ be arbitrary. Then $a + b \in A \oplus B$ for all $b \in B$. It follows that $a \in A \oplus B \ominus B$ by (3.2).

We now show (iii). By (3.2), we have

$$\begin{aligned} A \ominus B = (A_1 \cap A_2) \ominus B &= \{x \in A_1 \cap A_2 :\ x + b \in A_1 \cap A_2, \forall b \in B\} \\ &= \{x \in A_1 \cap A_2 :\ (x + b \in A_1) \wedge (x + b \in A_2), \forall b \in B\}, \\ (A_1 \ominus B) \cap (A_2 \ominus B) &= \{y \in A_1 \cap A_2 :\ (y + b_1 \in A_1) \wedge (y + b_2 \in A_2), \forall b_1, b_2 \in B\}. \end{aligned}$$

Then $(A_1 \ominus B) \cap (A_2 \ominus B) \subseteq A \ominus B$ clearly. For any $y \notin (A_1 \ominus B) \cap (A_2 \ominus B)$, there exists $b' \in B$ so that $(y + b' \notin A_1) \vee (y + b' \notin A_2)$, which indicates that $y \notin A \ominus B$. Hence $A \ominus B \subseteq (A_1 \ominus B) \cap (A_2 \ominus B)$, and (iii) holds.

To show (iv), we prove both $(\bigcap_{i=1}^{\infty} A_i) \ominus B \subseteq \bigcap_{i=1}^{\infty}(A_i \ominus B)$ and $\bigcap_{i=1}^{\infty}(A_i \cap B) \subseteq (\bigcap_{i=1}^{\infty} A_i) \ominus B$. Let $x \in (\bigcap_{i=1}^{\infty} A_i) \ominus B$. Then $x + b \in \bigcap_{i=1}^{\infty} A_i$ for all $b \in B$. That is to say $x + b \in A_i$ for all $b \in B$ and $i \in \mathbb{Z}^+$. It then implies that $x \in A_i \ominus B$ for all $i \in \mathbb{Z}^+$, i.e., $x \in \bigcap_{i=1}^{\infty}(A_i \ominus B)$. For the other direction, let $x \notin \bigcap_{i=1}^{\infty} A_i \ominus B$. Then there exists $b \in B$ such that $x + b \notin \bigcap_{i=1}^{\infty} A_i$, i.e., there exists $j \in \mathbb{Z}^+$ such that $x + b \notin A_j$. It follows that $x \notin A_j \ominus B$ and hence $x \notin \bigcap_{i=1}^{\infty}(A_i \ominus B)$.

For (v), we first show the closedness property. By (3.2), $A \ominus B = \bigcap_{b \in B}(A \setminus \{b\})$. If A is closed, then $A \setminus \{b\}$ is closed for all $b \in B$. It follows that $\bigcap_{b \in B}(A \setminus \{b\})$ is also closed. By (ii), we have $A \ominus B \oplus B \subseteq A$. If additionally A is bounded, then $A \ominus B$ is also bounded. □

3.2 Set Convergence

In approximating winning sets, we rely on the following definitions and results on set limits and convergence.

Definition 3.1. The *limit inferior* of a sequence $\{x_i\}$ is defined by

$$\liminf_{i\to\infty} x_i = \lim_{i\to\infty}\left(\inf_{j\geq i} x_j\right).$$

Similarly, the *limit superior* of $\{x_n\}$ is defined by

$$\limsup_{i\to\infty} x_i = \lim_{i\to\infty}\left(\sup_{j\geq i} x_j\right).$$

Let $d_A(x) \triangleq \inf_{y\in A}|x-y|$ denote the distance from a point x to a set A. Based on the definition of the distance between a point and a set, we provide the following set limits.

Definition 3.2 (Painlevé-Kuratowski Convergence)**.** For a sequence $\{A_i\}_{i=1}^{\infty}$ of subsets of $\mathbb{R}^n$. The *outer limit* of $\{A_i\}_{i=1}^{\infty}$ is defined by

$$\limsup_{i\to\infty} A_i = \left\{x\in\mathbb{R}^n : \liminf_{i\to\infty} d_{A_i}(x) = 0\right\}.$$

The *inner limit* of $\{A_i\}_{i=1}^{\infty}$ is defined by

$$\liminf_{i\to\infty} A_i = \left\{x\in\mathbb{R}^n : \limsup_{i\to\infty} d_{A_i}(x) = 0\right\}.$$

The *(set) limit* of $\{A_i\}_{i=1}^{\infty}$ exists iff the outer and inner limit sets are equal:

$$\lim_{i\to\infty} A_i = \limsup_{i\to\infty} A_i = \liminf_{i\to\infty} A_i.$$

Both inner and outer limits of any sequence of subsets on $\mathbb{R}^n$ by definition are closed [115]. Specifically for any *monotone* sequence $\{A_i\}_{i=1}^{\infty}$, i.e., either $A_i \subseteq A_{i+1}$ or $A_i \supseteq A_{i+1}$ for all $i\in\mathbb{Z}^+$, the set limit always exists.

Proposition 3.2 ([115])**.** Consider a sequence of sets $\{A_i\}_{i=1}^{\infty}$. Then

(i) $\lim_{i\to\infty} A_i = \mathrm{cl}\left(\bigcup_{i=1}^{\infty} A_i\right)$ whenever $A_i \subseteq A_{i+1}$ for all $i\in\mathbb{Z}^+$.

(ii) $\lim_{i\to\infty} A_i = \bigcap_{i=1}^{\infty}\mathrm{cl}\left(A_i\right)$ whenever $A_i \supseteq A_{i+1}$ for all $i\in\mathbb{Z}^+$.

3.3 Predecessor

A fundamental concept for the analysis of nonlinear control systems is called predecessor, which is the preimage of a given set under system dynamics.

Definition 3.3. Given a set $B \subseteq \mathbb{X}$, the *predecessor* of B with respect to system $\mathcal{S}$ is a set of states defined by

$$\text{Pre}^{\delta}(B) = \{x \in \mathbb{X} : \exists u \in \mathbb{U}, \text{ s.t. } f(x,u) + d \in B, \forall d \in \mathbb{D}\}. \tag{3.3}$$

The set of *valid control values* that lead to one-step transition to B for an $x \in \text{Pre}^{\delta}(B)$ is

$$\Pi_B^{\delta}(x) = \{u \in \mathbb{U} : f(x,u) + d \in B, \forall d \in \mathbb{D}\}. \tag{3.4}$$

We denote by $\text{Pre}(B)$ the predecessor of set B for system (2.1), and it is straightforward that $\text{Pre}^0(B) = \text{Pre}(B)$ for any $B \subseteq \mathbb{R}^n$. Likewise, we let $\Pi_B \triangleq \Pi_B^0$.

For $A, B \subseteq \mathbb{X}$, the predecessor of B that resides in a set A is the set $A \cap \text{Pre}^{\delta}(B)$. To simplify the notation, we let

$$\text{Pre}^{\delta}(B|A) \triangleq A \cap \text{Pre}^{\delta}(B). \tag{3.5}$$

The map Pre^{δ} satisfies the following properties since they are true for any function between subsets of states in $\mathbb{R}^n$.

Proposition 3.3. Let $A, B \subseteq Y \subseteq \mathbb{R}^n$. Given a function $h : Y \to Y$, then

(i) $h(A \cap B) \subseteq h(A) \cap h(B)$;

(ii) $h(A) \cup h(B) \subseteq h(A \cup B)$;

Without further assumptions on (2.2), we can additionally derive the following properties of the map Pre^{δ}.

Proposition 3.4. Let $A, B \subseteq \mathbb{X}$ and $\delta \geq 0$. Then

(i) $\text{Pre}^{\delta}(A) \subseteq \text{Pre}^{\delta}(B)$ if $A \subseteq B$,

(ii) $\text{Pre}^{\delta_2}(A) \subseteq \text{Pre}^{\delta_1}(A)$ if $0 \leq \delta_1 \leq \delta_2$,

(iii) $\text{Pre}^{\delta}(A) = \text{Pre}(A \ominus \mathcal{B}_{\delta})$.

Proof. The first two properties are straightforward by (3.3). For (iii),

$$\begin{aligned}\mathrm{Pre}(A \ominus \mathcal{B}_\delta) &= \{x \in \mathbb{X} : \ f(x,u) \in A \ominus \mathcal{B}_\delta\} \\ &= \{x \in \mathbb{X} : \ f(x,u) + y \in A, \forall y \in \mathcal{B}_\delta\} && \text{Expand } A \ominus \mathcal{B}_\delta \\ &= \mathrm{Pre}^\delta(A).\end{aligned}$$

Hence, (iii) is proved. □

Proposition 3.4 (i) indicates that the map Pre^δ is increasing, and (iii) implies that predecessors for non-deterministic system $\mathcal{S}^\delta$ can be constructed by using Pontryagin difference in computing the ones for nominal system $\mathcal{S}^0$.

If continuity is imposed to (2.2), then $\mathrm{Pre}^\delta(\cdot)$ will have more favorable properties for the control synthesis problems we considered in this thesis.

Assumption 3.1. The function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ in (2.2) is continuous with respect to both arguments, and the state space $\mathbb{X}$ and the input space $\mathbb{U}$ are compact.

In many real-world applications such as electrical power converters [45] and DISC engines [114], system state is controlled by switching between different operating modes, and system evolution under each mode may be determined by different functions. Control synthesis for systems with complex dynamics or specifications, e.g., robot motion planning [76] and flight management [44], is usually simplified to switching control between different operating modes and motion primitives. Such systems can be described by the following DEs:

$$x_{t+1} = f_{u_t}(x_t), \tag{3.6}$$

where $u_t \in \mathbb{U}$ indicates the active mode at time $t \in \mathbb{Z}_{\geq 0}$, and the input space $\mathbb{U}$ is finite.

A form as (3.6) can be represented by (2.2), but the function f is mostly not continuous with respect to the second argument. Hence, we make the following assumption.

Assumption 3.2. The function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ in (2.2) is continuous with respect to the first argument. The state space $\mathbb{X}$ is compact and the input space $\mathbb{U}$ is finite.

We now show in the following propositions that, under Assumption 3.1 or 3.2, the map Pre^δ preserves open and closedness property of a set.

Proposition 3.5. Suppose that Assumption 3.1 or 3.2 holds.

(i) If $\Omega \subseteq \mathbb{X}$ is closed (compact), then $\mathrm{Pre}^\delta(\Omega)$ is closed (compact).

(ii) If $\Omega \subseteq \mathbb{X}$ is open, then $\text{Pre}^\delta(\Omega)$ is open.

Proof. Given that Ω is closed (compact), $\Omega \ominus \mathbb{D} = \Omega \ominus \mathcal{B}_\delta$ is also closed (compact) by Proposition 3.1 (v). The conclusion trivially hold if $\Omega \ominus \mathbb{D} = \emptyset$, because $\text{Pre}^\delta(\Omega) = \emptyset$, which is compact. Hence we assume that $\Omega \ominus \mathbb{D} \neq \emptyset$. By Proposition 3.4 (iii), we can simplify the proof by considering Pre only.

Let $\{x_k\}_{k=0}^\infty$ be a convergent sequence in the set $\text{Pre}(\Omega)$ with the limit x^*, i.e., $\lim_{k\to\infty} x_k = x^*$. By (3.3), for all k, there exists $u_k \in \mathbb{U}$ such that $f(x_k, u_k) = \widetilde{x}_k \in \Omega$. We aim to show that $x^* \in \Omega$.

Under Assumption 3.1, the input space $\mathbb{U}$ is closed (compact). Then there exists a subsequence $\{u_{k_i}\}_{i=0}^\infty$ of $\{u_k\}_{k=0}^\infty$ $(0 \leq k_i \leq k)$ that converges to a point $u^* \in \mathbb{U}$. Let $\{x_{k_i}\}_{i=0}^\infty$ be the corresponding subsequence of $\{x_k\}_{k=0}^\infty$. By the continuity of f with respect to both arguments, we have

$$\lim_{i\to\infty} \widetilde{x}_{k_i} = \lim_{i\to\infty} f(x_{k_i}, u_{k_i}) = f(\lim_{i\to\infty} x_{k_i}, \lim_{i\to\infty} u_{k_i}) = f(x^*, u^*),$$

which means that $\{\widetilde{x}_{k_i}\}_{i=0}^\infty$ converges to some point $\widetilde{x}^* = f(x^*, u^*)$. Since Ω is closed (compact), $f(x^*, u^*) = \widetilde{x}^* \in \Omega$, which implies $x^* \in \text{Pre}(\Omega)$.

Under Assumption 3.2, $\mathbb{U}$ is finite. Let $\{x_{k_i}\}_{i=0}^\infty$ be the subsequence of $\{x_k\}_{k=0}^\infty$ that belong to Pre by using a common $u \in \mathbb{U}$, and $\lim_{i\to\infty} x_{k_i} = x^*$. By the continuity of $f(x, u)$ with respect to x for a fixed u, we have

$$\widetilde{x}^* = \lim_{i\to\infty} \widetilde{x}_{k_i} = \lim_{i\to\infty} f(x_{k_i}, u) = f(\lim_{i\to\infty} x_{k_i}, u) = f(x^*, u). \tag{3.7}$$

Similarly, (3.7) implies $x^* \in \Omega$.

To show (ii), we consider the complement $\left(\text{Pre}^\delta(\Omega)\right)^c$ of $\text{Pre}^\delta(\Omega)$:

$$\left(\text{Pre}^\delta(\Omega)\right)^c = \{x \in \mathbb{X} : \forall u \in \mathbb{U}, \exists d \in \mathbb{D}, \text{ s.t. } f(x, u) + d \in \Omega^c\},$$

where $\Omega^c = \mathbb{X} \setminus \Omega$ is closed with respect to $\mathbb{X}$ since $\Omega \subseteq \mathbb{X}$ is open.

Let $\{x_i\}_{i=0}^\infty$ be a convergent sequence in $\left(\text{Pre}^\delta(\Omega)\right)^c$ with $x = \lim_{i\to\infty} x_i$. Then for any given $u \in \mathbb{U}$, there exists $d_i \in \mathbb{D}$ such that $f(x_i, u) + d_i = y_i \in \Omega^c$ for all $i \in \mathbb{N}$. Since $\mathbb{D}$ is compact, there exists a convergent subsequence $\{d_{i_j}\}_{j=0}^\infty$ of $\{d_i\}_{i=0}^\infty$ with $d = \lim_{j\to\infty} d_{i_j} \in \mathbb{D}$. Then with the continuity of f with respect to x under Assumption 3.1 or 3.2 we have

$$\lim_{j\to\infty} (f(x_{i_j}, u) + d_{i_j}) = f(\lim_{j\to\infty} x_{i_j}, u) + \lim_{j\to\infty} d_{i_j} = f(x, u) + d = \lim_{j\to\infty} y_{i_j} = y \in \Omega^c$$

if Ω is open. It follows that $x \in \left(\text{Pre}^\delta(\Omega)\right)^c$ (i.e., for any $u \in \mathbb{U}$ there exists $d \in \mathbb{D}$ such that $f(x, u) + d \in \Omega^c$). Hence, $\text{Pre}^\delta(\Omega)$ is open with respect to $\mathbb{X}$ since $x \in \left(\text{Pre}^\delta(\Omega)\right)^c$ is closed. $\square$

The proofs of Proposition 3.5 (i) for nonlinear disturbed systems under similar assumptions to Assumption 3.1 can also be found in [108, Theorem 2], and [17, Theorem 5.2].

Similarly, the set of valid control values for each state in the predecessor of a given subset in the state space is compact.

Proposition 3.6. Suppose that Assumption 3.1 or 3.2 holds. The set $\Pi_\Omega^\delta(x)$ is compact for all $x \in \text{Pre}^\delta(\Omega)$, where $\Omega \subseteq \mathbb{X}$ is compact.

Proof. The set $\Pi_\Omega^\delta(x)$ is trivially compact if $\mathbb{U}$ is finite under Assumption 3.2. Suppose that Assumption 3.1 holds. Let $x \in \text{Pre}^\delta(\Omega)$ and $\{u_i\}_{i=0}^\infty \subseteq \Pi_\Omega^\delta(x)$ be a convergent sequence with $u = \lim_{i\to\infty} u_i$. Then $f(x, u_i) \in \Omega \ominus \mathbb{D}$ for all $i \in \mathbb{N}$. Since $\Omega \ominus \mathbb{D}$ is compact, we can find a convergent subsequence $\{f(x, u_{i_k})\}_{k=0}^\infty$ with $\lim_{k\to\infty} f(x, u_{i_k}) \in \Omega \ominus \mathbb{D}$. By the continuity of f with respect to u, we have

$$\lim_{k\to\infty} f(x, u_{i_k}) = f(x, \lim_{k\to\infty} u_{i_k}) = f(x, u) \in \Omega \ominus \mathbb{D},$$

which means $u \in \Pi_\Omega^\delta(x)$. Therefore, the set $\Pi_\Omega^\delta(x)$ is closed and hence compact for all $x \in \text{Pre}^\delta(x)$. □

If we consider a decreasing sequence of compact subsets of the state space $\mathbb{X}$ of system $\mathcal{S}$, the following distributive property of map Pre^δ under countable intersections can be shown.

Proposition 3.7. Suppose that Assumption 3.1 or 3.2 holds. Let $\{A_i\}_{i=0}^\infty$ be a decreasing sequence of compact subsets of $\mathbb{X}$. Then

$$\bigcap_{i=0}^{\infty} \text{Pre}^\delta(A_i) = \text{Pre}^\delta\left(\bigcap_{i=0}^{\infty} A_i\right). \tag{3.8}$$

Proof. We prove (3.8) by showing

$$\bigcap_{i=0}^{\infty} \text{Pre}^\delta(A_i) \subseteq \text{Pre}^\delta\left(\bigcap_{i=0}^{\infty} A_i\right), \tag{3.9}$$

$$\text{Pre}^\delta\left(\bigcap_{i=0}^{\infty} A_i\right) \subseteq \bigcap_{i=0}^{\infty} \text{Pre}^\delta(A_i). \tag{3.10}$$

We show (3.10) first. For any $x \in \text{Pre}^\delta(\bigcap_{i=0}^\infty A_i)$ there exists $u \in \mathbb{U}$ such that $f(x, u)+d \in A_i$ for all $d \in \mathbb{D}$ and $i \in \mathbb{Z}^+$. By definition we also have $x \in \bigcap_{i=0}^\infty \text{Pre}^\delta(A_i)$, which means that $\text{Pre}^\delta(\bigcap_{i=0}^\infty A_i) \subseteq \bigcap_{i=0}^\infty \text{Pre}^\delta(A_i)$.

To see (3.9), we aim to show that $x \in \mathrm{Pre}^{\delta}(\bigcap_{i=0}^{\infty} A_i)$ for all $x \in \bigcap_{i=0}^{\infty} \mathrm{Pre}^{\delta}(A_i)$. Let $x \in \bigcap_{i=0}^{\infty} \mathrm{Pre}^{\delta}(A_i)$ be arbitrary. Then there exists $u_i \in \mathbb{U}$ such that $a_i = f(x, u_i) \in (A_i \ominus \mathbb{D})$ for any fixed $i \in \mathbb{Z}^+$. Now consider the sequences $\{u_i\}_{i=1}^{\infty}$ and $\{a_i\}_{i=1}^{\infty}$.

Under Assumption 3.1, there exists a convergent subsequence $\{u_{i_j}\}_{j=0}^{\infty}$ with $\lim_{j\to\infty} u_{i_j} = u \in \mathbb{U}$, and $\{a_{i_j}\}_{j=1}^{\infty}$ is the corresponding subsequence of $\{a_i\}_{i=1}^{\infty}$. We can also find a convergent subsequence $\{a_{i_{j_k}}\}_{k=0}^{\infty}$ of $\{a_{i_j}\}_{j=1}^{\infty}$ with the limit point a, i.e.,

$$\lim_{k\to\infty} a_{i_{j_k}} = a, \quad a_{i_{j_k}} \in \left(A_{i_{j_k}} \ominus \mathbb{D}\right).$$

Since $A_i \ominus \mathbb{D}$ is closed for all $i \in \mathbb{Z}^+$, we have $a \in \bigcap_{k=0}^{\infty} \left(A_{i_{j_k}} \ominus \mathbb{D}\right) = \bigcap_{i=0}^{\infty} (A_i \ominus \mathbb{D})$. Then $a \in (\bigcap_{i=0}^{\infty} A_i) \ominus \mathbb{D}$ according to Proposition 3.1 (iv). By the continuity of $f(x, \cdot)$, we have

$$a = \lim_{k\to\infty} f(x, u_{i_{j_k}}) = f(x, \lim_{k\to\infty} u_{i_{j_k}}) = f(x, u) \in \left(\bigcap_{i=0}^{\infty} A_i\right) \ominus \mathbb{D}.$$

Under Assumption 3.2, $\mathbb{U}$ is finite, and thus there exists a constant subsequence $\{u_{i_j}\}_{j=0}^{\infty}$ with $u_{i_j} = u$ for all $j \in \mathbb{Z}^+$ such that $f(x, u) \in A_i \ominus \mathbb{D}$ for infinitely many i. Then $f(x, u) \in (\bigcap_{i=0}^{\infty} A_i) \ominus \mathbb{D}$.

Both assumptions all imply that there exists $u \in \mathbb{U}$ such that $f(x, u) \in (\bigcap_{i=0}^{\infty} A_i) \ominus \mathbb{D}$ for the arbitrary $x \in \bigcap_{i=0}^{\infty} \mathrm{Pre}^{\delta}(A_i)$. Hence $x \in \mathrm{Pre}^{\delta}(\bigcap_{i=0}^{\infty} A_i)$. This completes the proof. □

Similarly, a distributive property of map Pre^{δ} under countable unions of subsets of $\mathbb{X}$ can also be concluded.

Proposition 3.8. Let $\{A_i\}_{i=0}^{\infty}$ be an increasing sequence of open subsets of $\mathbb{X}$. Then

$$\bigcup_{i=0}^{\infty} \mathrm{Pre}^{\delta}(A_i) = \mathrm{Pre}^{\delta}\left(\bigcup_{i=0}^{\infty} A_i\right). \tag{3.11}$$

Proof. It is easy to see that $\bigcup_{i=0}^{\infty} \mathrm{Pre}^{\delta}(A_i) \subseteq \mathrm{Pre}^{\delta}(\bigcup_{i=0}^{\infty} A_i)$ by Proposition 3.3 (ii). Hence we only need to show that $\mathrm{Pre}^{\delta}(\bigcup_{i=0}^{\infty} A_i) \subseteq \bigcup_{i=0}^{\infty} \mathrm{Pre}^{\delta}(A_i)$.

Let $x \in \mathrm{Pre}^{\delta}(\bigcup_{i=0}^{\infty} A_i)$ be arbitrary. Then by definition there exists $u \in \mathbb{U}$ such that $f(x, u) + d \in \bigcup_{i=0}^{\infty} A_i$ for all $d \in \mathbb{D}$. By the Borel-Lebesgue finite covering theorem, there exists $i \in \mathbb{N}$ such that $f(x, u) + d \in A_i$ for all $d \in \mathbb{D}$ since the set $\{x \in \mathbb{X} : f(x, u) + d, \forall d \in \mathbb{D}\}$ is compact and $A_0 \subseteq A_1 \subseteq \cdots$. It follows that $x \in \mathrm{Pre}^{\delta}(A_i) \subseteq \bigcup_{i=0}^{\infty} \mathrm{Pre}^{\delta}(A_i)$. Therefore, $\mathrm{Pre}^{\delta}(\bigcup_{i=0}^{\infty} A_i) \subseteq \bigcup_{i=0}^{\infty} \mathrm{Pre}^{\delta}(A_i)$, which completes the proof. □

Note that Assumption 3.1 or 3.2 is not necessary in Proposition 3.8, but set A_i (for all $i \in \mathbb{N}$) has to be open in order that (3.11) holds.

It is nontrivial to exactly compute the predecessor $\mathrm{Pre}(Y)$ because of the nonlinear dynamics. Only for some special cases, e.g. predecessors of polyhedral sets with respect to linear dynamics, which can be characterized by linear inequalities, the exact computation is possible. Even for linear systems with polyhedral or ellipsoidal constraints, set operations such as Pontryagin difference are likely to introduce irregular shapes, which makes computation of accurate reachable sets impossible. Therefore, one has to seek approximations of $\mathrm{Pre}^{\delta}(Y)$.

For the purpose of control synthesis, inner approximations of predecessors are often used. Otherwise the set of valid control values for each predecessor is not well defined.

Assumption 3.3. Let $\widehat{\mathrm{Pre}} : 2^{\mathbb{X}} \to 2^{\mathbb{X}}$ be an approximation of the map Pre for system $\mathcal{S}$. Assume that the map $\widehat{\mathrm{Pre}}$ satisfies:

(i) $\widehat{\mathrm{Pre}}$ is monotone, i.e., $\widehat{\mathrm{Pre}}(A) \subseteq \widehat{\mathrm{Pre}}(B)$ if $A \subseteq B \subseteq \mathbb{X}$,

(ii) $\widehat{\mathrm{Pre}}(Y)$ is closed (compact) for any closed (compact) set $Y \subseteq \mathbb{X}$, and

(iii) $\widehat{\mathrm{Pre}}$ is lower bounded by Pre^{δ} for some $\delta > 0$, i.e.,

$$\mathrm{Pre}^{\delta}(Y) \subseteq \widehat{\mathrm{Pre}}(Y) \subseteq \mathrm{Pre}(Y) \tag{3.12}$$

for any set $Y \subseteq \mathbb{X}$.

To derive the robust completeness results, which will be presented in the following chapters, by using an approximation $\widehat{\mathrm{Pre}}$ of the predecessor map Pre, we rely on the properties for $\widehat{\mathrm{Pre}}$ given in Assumption 3.3. The monotonicity in (i) can be easily satisfied for most of the approximations of the predecessor map and is used to guarantee that the sequences of sets generated from the fixed-point iterations based on $\widehat{\mathrm{Pre}}$ are monotone. If (ii) is additionally satisfied, those sequences of sets retain the closedness (compactness) property, which is particularly useful for Theorem 4.1. And (iii) is crucial in developing the upper and lower bounds for the approximated the winning sets with respect to different LTL specifications in Chapter 4, 5, 6, and 7.

Chapter 4

Robustly Complete Invariance and Reachability Control

How to regulate the state or output of a control system is one of the most commonly studied problems in the control community, in which the goal is to design a feedback controller such that the system state or output converges to some given value, which is also termed as a *setpoint*. For a dynamical system in the form of (2.2), the regulation condition can be written mathematically as

$$\lim_{t\to\infty} \|x_t - r\|_2 = 0, \tag{4.1}$$

where $r \in \mathbb{X}$ is the setpoint and $\|\cdot\|_2$ denotes the Euclidean norm. The importance of such control problems lies in their wide applications in industry, e.g., voltage regulation of electrical power converters [45], room temperature stabilization inside a building [98], attitude control in flight control systems [42], and the adaptive cruise control [77, 96] and lane-keeping problems for autonomous vehicles [3].

According to the definition of the regulation problem as in (4.1), invariance and reachability control is another way to phrase this problem. The objective of *invariance control* is to maintain system state inside a given target area of the state space. In the presence of disturbances, the convergence in (4.1) cannot always be achieved, especially with additive disturbances (see (2.2)). Hence, it is more practical to consider a small region around the given setpoint. Set invariance [17] is also a paramount concept in constrained control where the controlled system trajectories are ideally inside an invariant set that is consistent with the given constraints so that the constraints would not be violated for all time.

Reachability control deals with the situations where the initial condition x_0 is outside a prescribed target set $\Omega \subseteq \mathbb{X}$, and the goal is to steer the system state to Ω at some finite future time instance. Such a problem has been investigated since [13] and studied for different types of systems such as piecewise affine systems [56, 21, 22, 59] and quantized control systems [15]. It is usually solved by formulating an optimal control problem [16, 102].

When the target set Ω is controlled invariant, solving the corresponding reachability problem leads to a control strategy that can drive the state of the system to Ω and maintain it inside Ω afterwards, which is what is concerned with in *reach-and-stay control* synthesis. A sound and complete control synthesis method for solving the reach-and-stay problem under uncertainties, however, is not addressed clearly in the literature.

Additionally, it is not difficult to see the importance of considering invariance, reachability and reach-and-stay control problems because they are intimately related to the MPC framework: the invariant set contained in the safe region under constraints needs to be determined in order to guarantee the satisfaction of constraints all the time; the control sequence is computed by solving optimal control problems for a fixed time horizon until the system state is stabilized to origin [88].

In this chapter, we will see how those traditional control problems can be translated to LTL control synthesis problems, and how the specialized Problem 2.1 can be solved by answering the following questions specifically:

- *Can the control synthesis for system $\mathcal{S}$ with respect to invariance, reachability or reach-and-stay objectives be sound and complete?*
- *Is memoryless control strategy sufficient for solving such problems?*
- *Is solving the reach-and-stay control problem equivalent to solving a reachability control problem with respect to a controlled invariant set?*

To express the above traditional invariance, reachability, and reach-and-stay control specifications in LTL formulas, a simple set of atomic proposition $AP = \{G, Fr\}$ is sufficient, where G and F stand for "goal area" and "free workspace", respectively. Suppose $\Omega \subseteq \mathbb{X}$ is the target set. Then the state space is initially partitioned into two cells Ω and $\mathbb{X} \setminus \Omega$, and the labeling function for system $\mathcal{S}$ can be defined as

$$L(x) = \begin{cases} G & x \in \Omega, \\ Fr & x \in \mathbb{X} \setminus \Omega. \end{cases} \tag{4.2}$$

As opposed to the indirect optimal control approach for solving regulation problems, we tackle such problems in a more direct way, which aims to provide fixed-point characterizations of the winning sets of system $\mathcal{S}$ with respect to the LTL formulas for invariance, reachability or reach-stay objectives. These winning sets are obtained by iterative computation of predecessors, which is mostly nontrivial for general nonlinear systems. Considering the cases in which the soundness and completeness is unlikely to achieve, we also discuss the conditions for sound and robustly complete control algorithms for such control specifications.

4.1 Invariance Control

First of all, we provide a formal definition of the invariance property for system $\mathcal{S}$.

Definition 4.1. Let Ω be a subset of the state space $\mathbb{X}$ of system $\mathcal{S}$ with the labeling function (4.2). A solution $\mathbf{x} = \{x_t\}_{t=0}^{\infty}$ of the system $\mathcal{S}$ satisfies an *invariance* property with respect to Ω if $x_t \in \Omega$ for all $t \in \mathbb{N}$. Such a property is written in an LTL formula $\varphi_s = \Box G$.

4.1.1 Maximal Controlled Invariant Set

Design of control strategies that realize the invariance property in Definition 4.1 is closely related to the following property of the give target set Ω.

Definition 4.2. A set $\Omega \subseteq \mathbb{R}^n$ is said to be δ*-robustly controlled invariant* for system $\mathcal{S}$ if, for any initial state $x_0 \in \Omega$, for all δ-bounded sequences of disturbances $\mathbf{d} = \{d_t\}_{t=0}^{\infty}$, i.e., $d_t \in \mathbb{D}$ for all $t \in \mathbb{N}$, there exists a control signal $\mathbf{u} = \{u_t\}_{t=0}^{\infty}$ such that $\text{Trace}(\mathbf{x}) \models \varphi_s$ where $\mathbf{x} = \{x_t\}_{t=0}^{\infty}$ is the resulting solution of $\mathcal{S}$. If $\delta = 0$, then Ω is called controlled invariant for system $\mathcal{S}^0$.

To check whether a set is (robustly) controlled invariant or not, we can rely on the following criterion based on predecessors.

Proposition 4.1 ([17, 67])**.** A set $\Omega \subseteq \mathbb{X}$ is δ-robustly controlled invariant for system $\mathcal{S}$ iff $\Omega \subseteq \text{Pre}^{\delta}(\Omega)$, where Pre^{δ} is defined in (3.3).

The given target set Ω in an invariance control problem is not necessarily (robustly) controlled invariant. If Ω is (robustly) controlled invariant itself, then $\Pi_{\Omega}^{\delta}(x) \neq \emptyset$ for all $x \in \Omega$ and the function Π_{Ω}^{δ} is a memoryless invariance control strategy. When Ω is not (robustly) controlled invariant, it is still possible to realize the invariance property by identifying subsets of Ω that are (robustly) controlled invariant. Among all such subsets, it is of interest to determine the maximal one, which constitutes the domain of the invariance control strategy.

Definition 4.3. Let $\Omega \subseteq \mathbb{X}$. The set $\mathcal{I}_\infty^\delta(\Omega)$ is said to be the *maximal δ-robustly controlled invariant set* inside Ω for system $\mathcal{S}$, if it is δ-robustly controlled invariant and contains all δ-robustly controlled invariant sets inside Ω. Specifically for system $\mathcal{S}^0$, such a set is called the maximal controlled invariant set inside Ω and denoted by $\mathcal{I}_\infty(\Omega)$.

Even for a nominal system $\mathcal{S}^0$, finding the maximal controlled invariant set $\mathcal{I}_\infty(\Omega)$ is not always helpful in practice, because any degree of uncertainties involved in system dynamics will destroy the invariance property of $\mathcal{I}_\infty(\Omega)$. And it is still possible that some part of Ω can be controlled invariant under disturbances. Therefore, we consider the following robust version of controlled invariant set for $\mathcal{S}^0$.

Definition 4.4. A set $\Omega \subseteq \mathbb{X}$ is said to be a *δ-robustly controlled invariant set* ($\delta \geq 0$) for system $\mathcal{S}^0$ if

$$\Omega \subseteq \mathrm{Pre}(\Omega \ominus \mathcal{B}_\delta). \tag{4.3}$$

We call Ω robustly controlled invariant if $\delta > 0$. The supremum of δ satisfying (4.3) is called the *robust invariance margin* of Ω.

By Proposition 3.4 (iii), The δ-robustly controlled invariant set $\mathcal{I}_\infty^\delta(\Omega)$ is consistent with Proposition 4.1.

It is interesting to note that by definition the maximal controlled invariant set itself is not robustly controlled invariant. This also indicates that the determination of the maximal invariant set is numerically nontrivial because of approximation errors.

Proposition 4.2. Let $\Omega \subseteq \mathbb{R}^n$ be compact and $\mathcal{I}_\infty(\Omega)$ be the maximal controlled invariant set in Ω. Suppose that Assumption 3.1 or 3.2 holds and $\mathcal{I}_\infty(\Omega) \neq \Omega$. Then $\mathcal{I}_\infty(\Omega)$ is not robustly controlled invariant.

Proof. We prove this by showing that some boundary points of $\mathcal{I}_\infty(\Omega)$ will be mapped into the boundary of $\mathcal{I}_\infty(\Omega)$. We only consider the case $\overset{\circ}{\Omega} \neq \emptyset$; otherwise the conclusion trivially holds by Definition 4.4 because $\overset{\circ}{\mathcal{I}}_\infty(\Omega) = \emptyset$, where $\overset{\circ}{\mathcal{I}}_\infty(\Omega)$ denotes the interior of set $\mathcal{I}_\infty(\Omega)$.

For the purpose of contradiction, we assume that $x \in (\partial\mathcal{I}_\infty(\Omega) \cap \overset{\circ}{\Omega})$, and there exists a $u \in U$ such that $f(x,u) \in \overset{\circ}{\mathcal{I}}_\infty(\Omega)$. That implies there exists a $r > 0$ such that $f(x,u) \oplus \mathcal{B}_r \subseteq \mathcal{I}_\infty(\Omega)$. By continuity of $f(\cdot,u)$ (from Assumption 3.1 or 3.2), we can find a $\delta(r) > 0$ such that any $x' \in x \oplus B_{\delta(r)}$ satisfies $f(x',u) \in f(x,u) \oplus \mathcal{B}_r$, and thus $f(x \oplus \mathcal{B}_{\delta(r)}, u) \subseteq \mathcal{I}_\infty(\Omega)$, which means x is an interior point of $\mathcal{I}_\infty(\Omega)$. This is a contradiction. □

We now consider the determination of (robustly) controlled invariant sets, which is crucial in the construction of invariance control strategies.

Let Inv^{δ} ($\delta \geq 0$) be a map between subsets of $\mathbb{R}^n$ defined as

$$\mathrm{Inv}^{\delta}(Y) = \mathrm{Pre}^{\delta}(Y|Y), \quad Y \subseteq \mathbb{R}^n. \tag{4.4}$$

We show in the next proposition that the maximal δ-robustly controlled invariant set inside a given compact set $\Omega \subseteq \mathbb{R}^n$ can be obtained by using the following algorithm:

$$\begin{cases} \mathrm{Inv}_0^{\delta}(\Omega) = \Omega, \\ \mathrm{Inv}_j^{\delta}(\Omega) = \mathrm{Inv}^{\delta}(\mathrm{Inv}_{j-1}^{\delta}(\Omega)), \end{cases} \tag{4.5}$$

where Inv_j^{δ} ($j \in \mathbb{Z}^+$) is the jth iterate of the map Inv^{δ}.

Proposition 4.3. Let $\Omega \subseteq \mathbb{X}$ be closed and $\delta \geq 0$. Given Assumption 3.1 or 3.2,

$$\mathcal{I}_{\infty}^{\delta}(\Omega) = \lim_{j \to \infty} \mathrm{Inv}_j^{\delta}(\Omega) = \bigcap_{j=0}^{\infty} \mathrm{Inv}_j^{\delta}(\Omega), \tag{4.6}$$

where $\mathcal{I}_{\infty}^{\delta}(\Omega)$ is the maximal δ-robustly controlled invariant set in Ω. Furthermore, $\mathcal{I}_{\infty}^{\delta}(\Omega)$ is a maximal fixed point of Inv^{δ}.

Proof. According to Proposition 3.5, if Ω is closed, then $\mathrm{Pre}(\Omega \ominus \mathcal{B}_{\delta})$, and hence $\mathrm{Inv}_j^{\delta}(\Omega)$ ($\forall j \in \mathbb{Z}^+$), is closed. By (4.4) and (4.5), $\{\mathrm{Inv}_j^{\delta}\}_{j=0}^{\infty}$ is decreasing. Then Proposition 3.2 shows that $\lim_{j\to\infty} I^j(\Omega) = \bigcap_{j=1}^{\infty} \mathrm{Inv}_j^{\delta}(\Omega)$ is closed and nonempty if $\mathrm{Inv}_j^{\delta}(\Omega) \neq \varnothing$ for all $j \in \mathbb{N}$.

First, we claim $\bigcap_{j=1}^{\infty} \mathrm{Inv}_j^{\delta}(\Omega) \subseteq \mathcal{I}_{\infty}^{\delta}(\Omega)$ by showing that $\bigcap_{j=1}^{\infty} \mathrm{Inv}_j^{\delta}(\Omega)$ is δ-robustly controlled invariant. For all $j \in \mathbb{Z}^+$, we have

$$\mathrm{Inv}_j^{\delta}(\Omega) = \mathrm{Inv}_{j-1}^{\delta}(\Omega) \cap \mathrm{Pre}^{\delta}(\mathrm{Inv}_{j-1}^{\delta}(\Omega)) \subseteq \mathrm{Pre}^{\delta}(\mathrm{Inv}_{j-1}^{\delta}(\Omega)).$$

Then $\bigcap_{j=0}^{\infty} \mathrm{Inv}_j^{\delta}(\Omega) \subseteq \bigcap_{j=1}^{\infty} \mathrm{Pre}^{\delta}(\mathrm{Inv}_{j-1}^{\delta}(\Omega))$. Since the sequence $\left\{\mathrm{Inv}_j^{\delta}(\Omega)\right\}_{j=0}^{\infty}$ is decreasing, by Proposition 3.7,

$$\bigcap_{j=1}^{\infty} \mathrm{Pre}^{\delta}(\mathrm{Inv}_{j-1}^{\delta}(\Omega)) = \mathrm{Pre}^{\delta}(\bigcap_{j=0}^{\infty} \mathrm{Inv}_j^{\delta}(\Omega)) = \mathrm{Pre}(\bigcap_{j=0}^{\infty} \mathrm{Inv}_j^{\delta}(\Omega) \ominus \mathcal{B}_{\delta}).$$

Hence, $\bigcap_{j=0}^{\infty} \mathrm{Inv}_j^{\delta}(\Omega) \subseteq \mathrm{Pre}(\bigcap_{j=0}^{\infty} \mathrm{Inv}_j^{\delta}(\Omega) \ominus \mathcal{B}_{\delta})$, which proves the claim.

Next, we show that $\mathcal{I}_{\infty}^{\delta}(\Omega) \subseteq \bigcap_{j=1}^{\infty} \mathrm{Inv}_j^{\delta}(\Omega)$. We assume that $\mathcal{I}_{\infty}^{\delta}(\Omega) \ominus \mathcal{B}_{\delta} \neq \varnothing$, otherwise $\mathcal{I}_{\infty}^{\delta}(\Omega) = \varnothing$, which means the conclusion trivially holds. We now use induction. For $j = 0$, we have $\mathcal{I}_{\infty}^{\delta}(\Omega) \subseteq \mathrm{Inv}_0^{\delta}(\Omega) = \Omega$. Suppose that $\mathcal{I}_{\infty}^{\delta}(\Omega) \subseteq \mathrm{Inv}_j^{\delta}(\Omega)$ for some $j \in \mathbb{N}$. By Proposition 4.1, for any $x \in (\mathrm{Inv}_j^{\delta}(\Omega) \backslash \mathrm{Inv}_{j+1}^{\delta}(\Omega))$, $f(x, u) \notin (\mathrm{Inv}_j^{\delta}(\Omega) \ominus \mathcal{B}_{\delta})$ for all $u \in U$, which also means $f(x, u) \notin (\mathcal{I}_{\infty}^{\delta}(\Omega) \ominus \mathcal{B}_{\delta})$. By definition of $\mathcal{I}_{\infty}^{\delta}(\Omega)$, $x \notin \mathcal{I}_{\infty}^{\delta}(\Omega)$. It follows that $\mathcal{I}_{\infty}^{\delta}(\Omega) \subseteq I_r^{j+1}(\Omega)$. Hence, $\mathcal{I}_{\infty}^{\delta}(\Omega) \subseteq \bigcap_{j=1}^{\infty} \mathrm{Inv}_j^{\delta}(\Omega)$.

Last, to see that $\mathcal{I}_{\infty}^{\delta}(\Omega)$ is a maximal fixed point of Inv^{δ}, it is sufficient to show that a set $Y \subseteq \Omega$ is a fixed point of Inv^{δ} iff Y is a δ-robustly controlled invariant set. If $Y \subseteq \Omega$ is a δ-robustly controlled invariant set, i.e., $Y \subseteq \mathrm{Pre}(Y \ominus \mathcal{B}_{\delta})$, then $\mathrm{Inv}^{\delta}(Y) = \mathrm{Pre}(Y \ominus \mathcal{B}_{\delta} | Y) = Y$. On the other side, if $\mathrm{Inv}^{\delta}(Y) = \mathrm{Pre}(Y \ominus \mathcal{B}_{\delta} | Y) = Y$, then we have $Y \subseteq \mathrm{Pre}(Y \ominus \mathcal{B}_{\delta})$, which means that Y is a δ-robustly controlled invariant set. $\square$

Proposition 4.3 essentially gives a fixed-point algorithm for the determination of the maximal (robustly) controlled invariant set inside a given target set. The actual computation, however, relies on how sets are represented and set operations are performed.

Consider Linear Time Invariant (LTI) systems in which (2.1) is of the form

$$f(x_t, u_t) + d_t = Ax_t + Bu_t + d_t,$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and $d_t \in \mathbb{D}$, which is given in (2.3).

If both the subtracted target set $\Omega \ominus \mathbb{D}$ and the set of control inputs $\mathbb{U}$ are polyhedra that are given by

$$\begin{aligned} \Omega \ominus \mathbb{D} &= \{x \in \mathbb{R}^n : Hx \leq h\}, \quad H \in \mathbb{R}^{l_1 \times n},\ h \in \mathbb{R}^{l_1} \\ \mathbb{U} &= \{u \in \mathbb{R}^m : Gu \leq g\}, \quad G \in \mathbb{R}^{l_2 \times m},\ g \in \mathbb{R}^{l_2}, \end{aligned}$$

where $l_1, l_2 \in \mathbb{Z}^+$ are the numbers of inequalities determining the polyhedra Ω and $\mathbb{U}$, respectively, then the predecessor of Ω is

$$\mathrm{Pre}^{\delta}(\Omega) = \mathrm{Pre}(\Omega \ominus \mathbb{D}) = \left\{ x \in \mathbb{R}^n : \begin{bmatrix} HA & HB \\ 0 & G \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq \begin{bmatrix} h \\ g \end{bmatrix},\ u \in \mathbb{R}^m \right\}. \tag{4.7}$$

The set $\mathrm{Pre}^{\delta}(\Omega)$ and hence $\mathrm{Inv}^{\delta}(\Omega)$ is also polyhedral, because the intersection of polyhedra is still a polyhedron. Then the iterations of (4.5) can go on without losing the polyhedral properties. However, the maximal (robustly) controlled invariant set is not necessarily polyhedral. The following example illustrates such a case.

For a general nonlinear form of (2.1), the computation of $\mathrm{Pre}(\Omega)$ is not as easy as for the LTI case, let alone the possibility of terminating in a finite number of iterations.

Example 4.1. Consider an LTI system $x_{t+1} = Ax_t$, where

$$A = \begin{bmatrix} 1.0810 & 0.4517 \\ -0.0903 & 0.7197 \end{bmatrix}.$$

With a pair of complex eigenvalues $0.9003 \pm 0.0903i$, this LTI system is globally stable. Hence, there exists a (controlled) invariant set inside $\Omega = [-1, 1] \times [-1, 1]$. However, Ω itself is not (controlled) invariant. This is because the system trajectories are spiral and some of them will leave Ω provisionally although they will eventually converge to the origin $(0, 0)$.

Represented by a polyhedron, $\Omega = \{x \in \mathbb{R}^n : Hx \leq h\}$, where

$$H = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \quad h = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

By (4.7), we have

$$\text{Inv}_1(\Omega) = \text{Pre}(\Omega|\Omega) = \left\{ x \in \mathbb{R}^n : \begin{bmatrix} HA \\ H \end{bmatrix} x \leq \begin{bmatrix} h \\ h \end{bmatrix} \right\},$$

which is a new polyhedron $\{x \in \mathbb{R}^n : H_1 x \leq h_1\}$, where

$$H_1 = \begin{bmatrix} HA \\ H \end{bmatrix} = \begin{bmatrix} 1.0810 & 0.4517 \\ -1.0810 & -0.4517 \\ -0.0903 & 0.7197 \\ 0.0903 & -0.7197 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \quad h_1 = \begin{bmatrix} h \\ h \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

The polyhedral sets obtained within the first 4 iterations are shown in Figure 4.1. It can be observed that the polyhedral set $\text{Inv}_i(\Omega)$ $(i \in \mathbb{N})$ keeps shrinking towards the real maximal (controlled) invariant set $\mathcal{I}_\infty(\Omega)$, which is bounded by two red boundary lines. It is also clear that $\mathcal{I}_\infty(\Omega)$ is not a polyhedron, and this implies that we can never achieve $\mathcal{I}_\infty(\Omega)$ within a finite number of iterations.

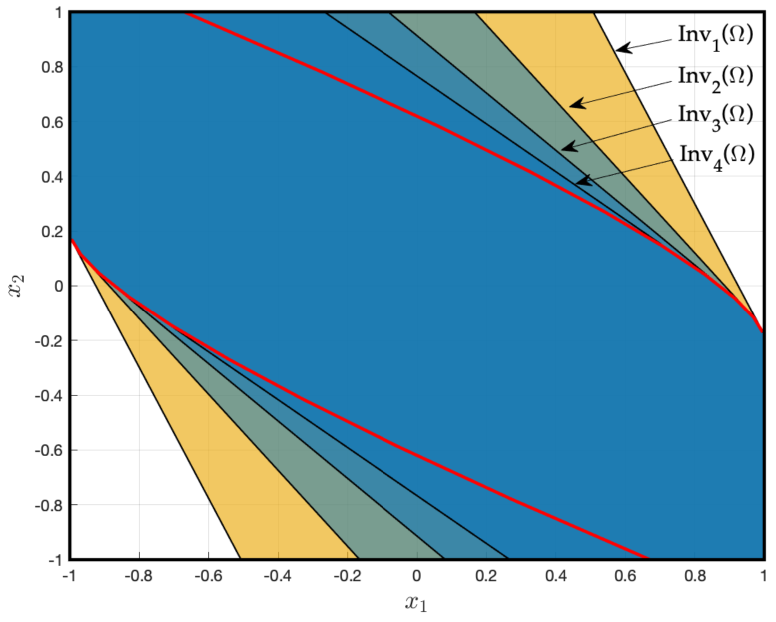


Figure 4.1: The set $\mathrm{Inv}_i(\Omega)$ for $i = 0, 1, 2, 3, 4$. The outermost black box represents the initial set $\mathrm{Inv}_0(\Omega) = \Omega$. The red lines are the real boundaries of the maximal (controlled) invariant set inside Ω.

Example 4.2. Consider a discrete-time version of a second-order nonlinear system taken from [69, Example 8.6] as follows:

$$\begin{aligned} x_{t+1} &= x_t + 0.1y_t \\ y_{t+1} &= -0.1x_t + 0.033x_t^3 + 0.9y_t. \end{aligned}$$

It has three isolated equilibrium points at $(0, 0)$, $(\sqrt{3}, 0)$ and $(-\sqrt{3}, 0)$. The region between the manifolds that pass through $(\sqrt{3}, 0)$ and $(-\sqrt{3}, 0)$ is the maximal positively invariant set, which is difficult to express analytically.

4.1.2 Robust Completeness

Finding the (robustly) maximal controlled invariant set is equivalent to determining the winning set for system $\mathcal{S}$ with respect to invariance specification φ_s. If we keep track of the valid control values during iterations, the corresponding control strategy can be constructed.

As we have discussed in Chapter 3, however, predecessors are not easy to compute precisely under nonlinear dynamics. So a workaround is to use inner approximations of predecessors instead so that the resulting control strategy is well defined for all the states inside the approximated winning set.

A risk of using inner approximations of predecessors is the possible loss of controlled invariance of the approximated winning set. The following theorem investigates the type of inner approximations that preserves the controlled invariance property.

Theorem 4.1 (Soundness and Robust Completeness)**.** Let $\Omega \subseteq \mathbb{X}$ be compact. Define a new map $\widehat{\text{Inv}} : 2^X \to 2^X$ with $\widehat{\text{Inv}}(Y) = \widehat{\text{Pre}}(Y) \cap Y$ for all $Y \subseteq \mathbb{X}$, where $\widehat{\text{Pre}}$ is an approximation of Pre that satisfies Assumption 3.3 for some $\delta > 0$. Consider the algorithm

$$\begin{cases} \widehat{\text{Inv}}_0(\Omega) = \Omega, \\ \widehat{\text{Inv}}_j(\Omega) = \widehat{\text{Inv}}(\widehat{\text{Inv}}_{j-1}(\Omega)). \end{cases} \tag{4.8}$$

Then (4.8) converges, i.e.,

$$\widehat{\mathcal{I}}_\infty \triangleq \lim_{j\to\infty} \widehat{\text{Inv}}_j(\Omega) = \bigcap_{j=0}^{\infty} \widehat{\text{Inv}}_j(\Omega). \tag{4.9}$$

Moreover, if $\widehat{\mathcal{I}}_\infty \neq \emptyset$, then $\widehat{\mathcal{I}}_\infty$ is controlled invariant, i.e., $\widehat{\mathcal{I}}_\infty \subseteq \text{Pre}(\widehat{\mathcal{I}}_\infty)$, and it is a fixed point of $\widehat{\text{Inv}}$ that satisfies

$$\mathcal{I}^\delta_\infty(\Omega) \subseteq \widehat{\mathcal{I}}_\infty(\Omega) \subseteq \mathcal{I}_\infty(\Omega), \tag{4.10}$$

where $\mathcal{I}_\infty(\Omega)$ and $\mathcal{I}^\delta_\infty(\Omega)$ denote the maximal and δ-robustly maximal controlled invariant set, respectively.

Proof. Let $\{Y_i\}_{i=0}^\infty$, $\{\widehat{Y}_i\}_{i=0}^\infty$, and $\left\{Y_i^\delta\right\}_{i=0}^\infty$ be the sequences of sets generated by (4.5) using Pre, $\widehat{\text{Pre}}$ and Pre^δ, respectively. The sequences $\{Y_i\}_{i=0}^\infty$ and $\left\{Y_i^\delta\right\}_{i=0}^\infty$ are decreasing by (4.4). Under Assumption 3.3, $\widehat{\text{Pre}}$ and hence $\{\widehat{Y}_i\}_{i=0}^\infty$ is also decreasing. Since $\widehat{Y}_i$ is compact for all $i \in \mathbb{N}$ given that $\widehat{Y}_0 = \Omega$ is compact, the $\lim_{i\to\infty} \widehat{Y}_i$ exists and is given by (4.9) by Proposition 3.2.

Next we show that $\widehat{\mathcal{I}}_\infty$ is a fixed point of $\widehat{\text{Inv}}$ and controlled invariant. For any $x \in \widehat{\mathcal{I}}_\infty$, if $x \notin \widehat{\text{Pre}}(\widehat{\mathcal{I}}_\infty)$, then there exists some $j \in \mathbb{N}$ such that $x \notin \widehat{\text{Pre}}(\widehat{\text{Inv}}_j(\Omega))$. By the definition of the map $\widehat{\text{Inv}}$, we have $x \notin \widehat{\text{Inv}}_{j+1}(\Omega)$, which implies $x \notin \widehat{\mathcal{I}}_\infty$. Therefore, $\widehat{\mathcal{I}}_\infty \subseteq \widehat{\text{Pre}}(\widehat{\mathcal{I}}_\infty)$, and it follows that $\widehat{\mathcal{I}}_\infty \subseteq \text{Pre}(\widehat{\mathcal{I}}_\infty)$ and $\widehat{\mathcal{I}}_\infty = \widehat{\text{Inv}}(\widehat{\mathcal{I}}_\infty)$.

We now prove (4.10) by induction. According to (4.5), initially $Y_0 = \widehat{Y}_0 = Y_0^\delta = \Omega$. By (3.12), we have $\mathrm{Pre}(Y_j \ominus \mathcal{B}_\delta | Y_j) \subseteq \widehat{\mathrm{Pre}}(Y_j|Y_j) \subseteq \mathrm{Pre}(Y_j|Y_j)$ for all $j \in \mathbb{Z}_{\geq 0}$, which means that $Y_1^\delta \subseteq \widehat{Y}_1 \subseteq Y_1$. Assume that $Y_i^\delta \subseteq \widehat{Y}_i \subseteq Y_i$ for some $i \in \mathbb{Z}^+$. Then

$$\begin{aligned} Y_{j+1}^\delta &= \mathrm{Pre}(Y_j^\delta \ominus \mathcal{B}_\delta | Y_j^\delta) \subseteq \mathrm{Pre}(\widehat{Y}_j \ominus \mathcal{B}_\delta | \widehat{Y}_j) \subseteq \widehat{\mathrm{Pre}}(\widehat{Y}_j | \widehat{Y}_j) = \widehat{Y}_{j+1} \\ \widehat{Y}_{j+1} &= \widehat{\mathrm{Pre}}(\widehat{Y}_j | \widehat{Y}_j) \subseteq \mathrm{Pre}(\widehat{Y}_j | \widehat{Y}_j) \subseteq \mathrm{Pre}(Y_j | Y_j) = Y_{j+1}. \end{aligned}$$

Hence, $Y_j^\delta \subseteq Y_j \subseteq Y_j$ for all $j \in \mathbb{N}$. If $\widehat{\mathcal{I}}_\infty(\Omega) \neq \emptyset$, then (4.10) trivially holds. If $\widehat{\mathcal{I}}_\infty(\Omega) = \bigcap_{j=0}^\infty Y_j = \emptyset$, then there exists some integer $N > 0$ such that $Y_N = \emptyset$. Otherwise $\widehat{\mathcal{I}}_\infty(\Omega)$ is nonempty since Ω is compact. It follows that $Y_N^\delta = \emptyset$ and (4.10) holds. $\square$

Theorem 4.1 additionally suggests that robustly controlled invariance is a sufficient condition for inner-approximating the maximal controlled invariant set. It can also be inferred that, for any nonempty controlled invariant set that is not robustly invariant, approximation of predecessors under any precision fails to give a solid approximation. This is because there does not exist a positive real number as the tolerance for the set approximation error. The following example is such a scenario.

Example 4.3. Consider a discrete-time system $x(t+1) = A_\theta x(t)$, where

$$A_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Every state moves on a circle centered at the origin. The approximated set $\widehat{\mathcal{I}}_\infty(\Omega)$ for any $\Omega \subseteq \mathbb{R}^2$ by using any approximation $\widehat{\mathrm{Pre}}$ of Pre will be an empty set, since Ω is not robustly invariant for this system.

Based on Theorem 4.1, let us now revisit Problem 2.2 with an invariance specification φ_s.

Corollary 4.1. Given system $\mathcal{S}^0$ with the labeling function (4.2) where $\Omega \subseteq \mathbb{X}$ is compact and an invariance specification $\varphi_s = \Box G$, consider algorithm (4.8) under Assumption 3.3 for some $\delta > 0$ and $\widehat{\mathcal{I}}_\infty$ defined in (4.9).

(i) If $\widehat{\mathcal{I}}_\infty(\Omega) \neq \emptyset$, then $\widehat{\mathcal{I}}_\infty(\Omega)$ is controlled invariant for system $\mathcal{S}^0$ with the control strategy:

$$\kappa(x) = \bigcap_{j=1}^{\infty} \Pi_{\widehat{\mathrm{Inv}}_j(\Omega)}(x), \quad \forall x \in \widehat{\mathcal{I}}_\infty(\Omega). \tag{4.11}$$

(ii) If $\widehat{\mathcal{I}}_\infty(\Omega) = \emptyset$, then Ω is not δ-robustly controlled invariant for system $\mathcal{S}^0$.

Proof. If $\widehat{\mathcal{I}}_\infty(\Omega) = \emptyset$, then $\mathcal{I}^\delta_\infty(\Omega) = \emptyset$ by (4.10), which implies (ii). Now we prove (i). Let $x \in \widehat{\mathcal{I}}_\infty(\Omega)$ be arbitrary. By Proposition 3.6, the set $\Pi_{\widehat{\mathrm{Inv}}_j(\Omega)}(x)$ is compact for all $j \in \mathbb{Z}^+$ and $x \in \widehat{\mathcal{I}}_\infty(\Omega)$. Thus, $\bigcap_{j=1}^\infty \Pi_{\widehat{\mathrm{Inv}}_j(\Omega)}(x)$ exists and is compact. For any control input $u \in \bigcap_{j=1}^\infty \Pi_{\widehat{\mathrm{Inv}}_j(\Omega)}(x)$, we have by algorithm (4.8) $f(x,u) \in \widehat{\mathrm{Inv}}_{j-1}(\Omega)$ for all $j \in \mathbb{Z}^+$, which means that $f(x,u) \in \bigcap_{j=1}^\infty \widehat{\mathrm{Inv}}_{j-1}(\Omega) = \bigcap_{j=0}^\infty \widehat{\mathrm{Inv}}_j(\Omega) = \widehat{\mathcal{I}}_\infty(\Omega)$. Hence, $\widehat{\mathcal{I}}_\infty(\Omega)$ a set inside Ω that is controlled invariant using the control strategy (4.11). $\square$

Corollary 4.1 essentially says that the control strategy (4.11) is sound and robustly complete if the approximations of the predecessors satisfy Assumption 3.3 during iterations.

4.2 Reachability Control

Reachability plays an important role in analysis and control of dynamical systems. For a control problem, a target set is given and the objective is to steer the system trajectories into the target set. The following is a formal description of the reachability property.

Definition 4.5. Let Ω be a subset of the state space $\mathbb{X}$ of system $\mathcal{S}$ with the labeling function (4.2). A solution $\mathbf{x} = \{x_t\}_{t=0}^\infty$ of the system $\mathcal{S}$ satisfies a *reachability* property with respect to Ω if there exists $k \in \mathbb{N}$ such that $x_k \in \Omega$. Such a property can be written in the LTL formula $\varphi_{\mathrm{r}} = \Diamond G$.

4.2.1 Robustly Backward Reachable Set

For the purpose of control, we wish to determine the winning set of system $\mathcal{S}$ with respect to the reachability specification $\varphi_{\mathrm{r}} = \Diamond G$, which can be specialized as the maximal δ-robustly backward reachable set defined below.

Definition 4.6. Let Ω be a subset of the state space $\mathbb{X}$ of system $\mathcal{S}$ with the labeling function (4.2). A set $\mathcal{BR}^\delta_\infty(\Omega) \subseteq \mathbb{X}$ is said to be the *maximal δ-robustly backward reachable set* of system $\mathcal{S}$ from Ω if it contains (and only contains) any initial state $x_0 \in \mathbb{X}$ that satisfies: for all δ-bounded sequences of disturbances $\mathbf{d} = \{d_t\}_{t=0}^\infty$, i.e., $d_t \in \mathbb{D}$ for all $t \in \mathbb{N}$, there exists a control signal $\mathbf{u} = \{u_t\}_{t=0}^\infty$ such that $\mathrm{Trace}(\mathbf{x}) \models \varphi_{\mathrm{r}}$ where $\mathbf{x} = \{x_t\}_{t=0}^\infty$ is the resulting solution of $\mathcal{S}$.

For nominal system $\mathcal{S}^0$, the winning set with respect to the reachability specification is the backward reachable set $\mathcal{BR}_\infty(\Omega)$.

It is worth noting that the integer k and the control signal $\mathbf{u}$ are dependent on the sequence of disturbances $\mathbf{d}$ in Definition 4.6. In other words, the minimum time step for any initial state $x_0 \in \mathcal{BR}^\delta_\infty(\Omega)$ to be controlled into Ω can be different given different sequences of disturbance.

Definition 4.7. A set $\Omega \subseteq \mathbb{X}$ is said to be δ*-robustly reachable* for system $\mathcal{S}$ (or *reachable* for system $\mathcal{S}^0$) if $\mathcal{BR}^\delta_\infty(\Omega) \neq \emptyset$.

We now introduce the following definition for the characterization of $\mathcal{BR}^\delta_\infty(\Omega)$.

Definition 4.8. The N*-step* δ*-robustly backward reachable set* of system $\mathcal{S}$ from a target set $\Omega \subseteq \mathbb{X}$ is a set of initial states from which Ω can be reached within N ($N \in \mathbb{N}$) steps for any possible sequence of disturbance, i.e.,

$$\begin{aligned}\mathcal{BR}^\delta_N(\Omega) = \{x \in \mathbb{X} : \forall \{d_i\}_{i=0}^N \, (d_i \in \mathbb{D}), \exists \{u_i\}_{i=0}^N \text{ s.t. } \{x_i\}_{i=0}^N \text{ by (2.2) satisfies} \\ x_0 = x, \; x_k \in \Omega, 0 \leq k \leq N\}.\end{aligned} \tag{4.12}$$

For nominal system $\mathcal{S}^0$, the N-step and maximal δ-robustly backward reachable set $\mathcal{BR}^\delta_N(\Omega)$ and $\mathcal{BR}^\delta_\infty(\Omega)$ are reduced to the N-step backward reachable set $\mathcal{BR}_N(\Omega)$ and maximal backward reachable set $\mathcal{BR}_\infty(\Omega)$, respectively.

Define Rch^δ as a map between subsets of $\mathbb{R}^n$:

$$\text{Rch}^\delta(Y) = \text{Pre}^\delta(Y) \cup Y, \quad Y \subseteq \mathbb{R}^n. \tag{4.13}$$

Let us now consider the algorithm ($i \in \mathbb{Z}^+$):

$$\begin{cases}\text{Rch}^\delta_0(\Omega) = \Omega, \\ \text{Rch}^\delta_j(\Omega) = \text{Rch}^\delta(\text{Rch}^\delta_{j-1}(\Omega)).\end{cases} \tag{4.14}$$

It can be seen straightforwardly that the sequence $\left\{\text{Rch}^\delta_j(\Omega)\right\}_{j=0}^\infty$ is increasing and with a slight use of induction, we can conclude the following result.

Proposition 4.4. Given system $\mathcal{S}$ and a subset $\Omega \subseteq \mathbb{X}$, we have

$$\mathcal{BR}^\delta_N(\Omega) = \text{Rch}^\delta_N(\Omega), \quad \forall N \in \mathbb{N}, \tag{4.15}$$

where Rch^δ and $\text{Rch}^\delta_N(\Omega)$ are defined in (4.13) and (4.14), respectively.

Proof. We show it by induction. The basic case holds because by (4.14) $\mathrm{Rch}_0^\delta(\Omega) = \Omega$, which is the set of states that can be controlled into Ω under any allowable disturbance in 0 steps. Suppose that $\mathrm{Rch}_j^\delta(\Omega)$ is the j-step δ-robustly backward reachable set. By (4.14) and Definition 3.3, we have

$$\mathrm{Rch}_{j+1}^\delta(\Omega) = \mathrm{Pre}^\delta(\mathrm{Rch}_j^\delta(\Omega)) \cup \mathrm{Rch}_j^\delta(\Omega),$$

which additionally includes all the states that can be controlled inside $\mathrm{Rch}_j^\delta(\Omega)$ in one step under any allowable disturbance. Hence, we have $\mathrm{Rch}_{j+1}^\delta(\Omega)$ is the $(j+1)$-step δ-robustly backward reachable set and the claim is proved. $\square$

As we have seen in Example 4.1, for LTI systems and a given polyhedral target set Ω, the set $\mathrm{Inv}_j^\delta(\Omega)$ in each iteration $j \in \mathbb{N}$ can be computed precisely. Robustly backward reachable sets $\mathrm{Rch}_j^\delta(\Omega)$, however, are not as easily obtained as the set $\mathrm{Inv}_j^\delta(\Omega)$ $(j \in \mathbb{N})$. This is because the set union in (4.13) for the computation of $\mathrm{Rch}_j^\delta(\Omega)$ does not keep the shape of polyhedra.

Example 4.4. Consider a discrete-time double integrator with disturbance [67]:

$$x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u_t + d_t, \tag{4.16}$$

where $u_t \in \mathbb{U} = \{u \in \mathbb{R}^2 : \|u\|_\infty \le 1\}$ and $d_t \in \mathbb{D} = \{d \in \mathbb{R}^2 : \|d\|_\infty \le 0.1\}$. The vector x represents the position and velocity. System (4.16) is a sampled-data version (with sampling time $\tau = 1$) of the following ODE that models acceleration of an object:

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \quad u \in [-1, 1]. \tag{4.17}$$

We consider a target reach set $\Omega = [-0.3, 0.3] \times [-0.3, 0.3]$ on the state space $\mathbb{X} = [-8, 8] \times [-4, 4]$. As in Example 4.1, the first 4 iterations are plotted in Figure 4.2. The 1-step δ-robustly backward reachable set $\mathrm{Rch}_1^\delta(\Omega) = \mathrm{Pre}^\delta(\Omega) \cup \Omega$ is the union of the yellow rectangle and the innermost polytope, which is concave and hence not a polytope. Specifically in this case, the backward reachable sets are not polyhedral until $\mathrm{Rch}_4^\delta(\Omega)$, which includes all the previous sets.

Based on (4.15), we can further characterize the maximal backward reachable set $\mathcal{BR}_\infty^\delta(\Omega)$ according to the following proposition.

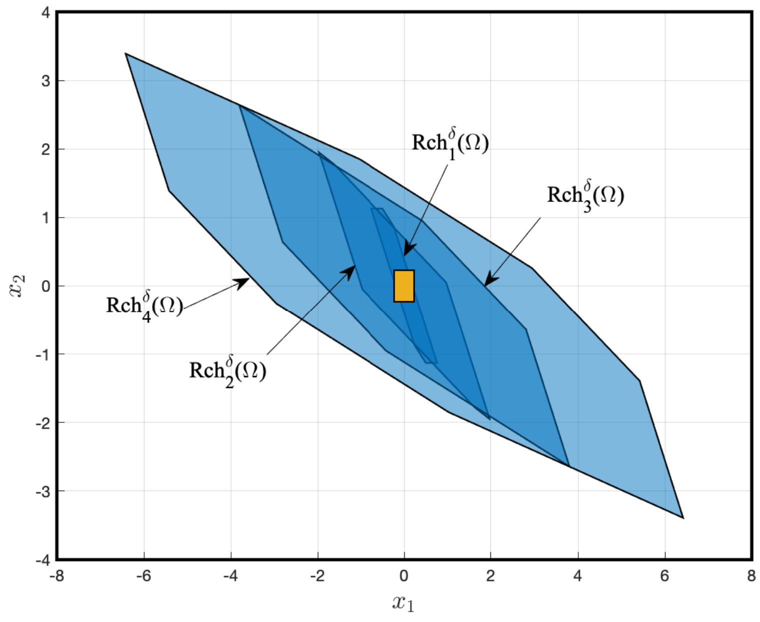


Figure 4.2: The 4-step δ-robustly backward reachable set. The yellow rectangle region in the center is the target set Ω.

Proposition 4.5. For system $\mathcal{S}$, let $\Omega \subseteq \mathbb{X}$ be open and $\delta \geq 0$. Then

$$\mathcal{BR}_\infty^\delta(\Omega) = \bigcup_{j=0}^{\infty} \mathrm{Rch}_j^\delta(\Omega), \tag{4.18}$$

and $\mathcal{BR}_\infty^\delta(\Omega)$ is a fixed point of the map Rch^δ.

Proof. The direction $\mathcal{BR}_\infty^\delta(\Omega) \supseteq \bigcup_{i=0}^{\infty} \mathrm{Rch}_j^\delta(\Omega)$ is clear because $\mathrm{Rch}_j^\delta(\Omega) \subseteq \mathcal{BR}_\infty^\delta(\Omega)$ for all $j \in \mathbb{N}$.

To show that $\mathcal{BR}_\infty^\delta(\Omega) \subseteq \bigcup_{j=0}^{\infty} \mathrm{Rch}_j^\delta(\Omega)$, we first claim that $\bigcup_{j=0}^{\infty} \mathrm{Rch}_j^\delta(\Omega)$ is a fixed point of Rch^δ. Let $A_j = \mathrm{Rch}_j^\delta(\Omega) \subseteq \mathbb{X}$. Then $\{A_j\}_{j=0}^{\infty}$ is open and increasing, and by Proposition 3.8, we have

$$\mathrm{Rch}^{\delta}(\bigcup_{j=0}^{\infty}\mathrm{Rch}_j^{\delta}(\Omega)) = \mathrm{Pre}^{\delta}\left(\bigcup_{j=0}^{\infty}A_j\right)\cup\bigcup_{j=0}^{\infty}A_j = \bigcup_{i=0}^{\infty}\mathrm{Pre}^{\delta}(A_j)\cup\bigcup_{j=0}^{\infty}A_j$$
$$= \bigcup_{j=0}^{\infty}\left(\mathrm{Pre}^{\delta}(A_j)\cup A_j\right) = \bigcup_{j=0}^{\infty}\mathrm{Rch}_{j+1}^{\delta}(\Omega) = \bigcup_{j=0}^{\infty}\mathrm{Rch}_j^{\delta}(\Omega).$$

We now show $x \notin \mathcal{BR}_{\infty}^{\delta}(\Omega)$ for all $x \notin \bigcup_{j=0}^{\infty}\mathrm{Rch}_j^{\delta}(\Omega)$. Let $x_0 \notin \bigcup_{j=0}^{\infty}\mathrm{Rch}_j^{\delta}(\Omega)$ be arbitrary. Then $x_0 \notin \mathrm{Pre}^{\delta}(\bigcup_{j=0}^{\infty}\mathrm{Rch}_j^{\delta}(\Omega))$ because $\bigcup_{j=0}^{\infty}\mathrm{Rch}_j^{\delta}(\Omega)$ is a fixed point of Rch^{δ}. This means that for all $u_0 \in \mathbb{U}$ there exists $d_0 \in \mathbb{D}$ (depending on u_0) such that $x_1 = f(x_0, u_0) + d_0 \notin \bigcup_{j=0}^{\infty}\mathrm{Rch}_j^{\delta}(\Omega)$ and thus $x_1 \notin \Omega$. Similarly for x_1, for all $u_1 \in \mathbb{U}$ we can find $d_1 \in \mathbb{D}$ such that $x_2 = f(x_1, u_1) + d_1 \notin \Omega$. Therefore, we can construct a sequence of disturbance $\{d_t\}_{t=0}^{\infty}$ such that $x_t \notin \Omega$ for all $\{u_t\}_{t=0}^{\infty}$ and all $t \in \mathbb{N}$, which implies that $x \notin \mathcal{BR}_{\infty}^{\delta}(\Omega)$.

Hence (4.18) is proved, and the result that $\mathcal{BR}_{\infty}^{\delta}(\Omega)$ is a fixed point of the map Rch^{δ} follows straightforwardly. □

Note that the target set Ω has to be open in order that algorithm (4.18) yields the maximal robustly backward reachable set, as opposed to Proposition 4.3. This is not surprising because reachability is the dual of invariance, which requires compactness of the target set. To explain why (4.15) does not apply to a closed target set in general, we give the following counter example.

Example 4.5. Let $a_1 \approx 0.1127$ and $a_2 \approx 0.8873$ be the roots of $a = a^2 + 0.1$. We consider a target set $\Omega = [0, 0.2] \cup \{a_2\}$ for the system

$$x_{t+1} = \begin{cases} x_t^2 + d_t & x_t \in [0, a_2], \\ a_2^2 + d_t & x_t \in (a_2, 1], \end{cases}$$

where $x_t \in \mathbb{X} = [0, 1]$, $d_t \in \mathbb{D} = [0, 0.1]$.

Then the trajectories with the initial condition $x_0 \in [0, a_2)$ will enter the region $[0, a_2]$ asymptotically under all possible sequences of disturbance. Hence, we have

$$\bigcup_{j=0}^{\infty}\mathrm{Rch}_j^{\delta}(\Omega) = [0, a_2).$$

However, the real maximal robustly backward reachable set is the entire state space $\mathbb{X}$ since all $x \in (a_2, 1]$ is mapped within $[0, a_2]$ and a_2 is the point backward reachable from Ω.

Controlling the system state into an open set is what usually required in the applications of reachability control. Even if sometimes the target set Ω is given as a closed set, it is always safer to design the reachability control strategy with respect to the interior part of Ω.

4.2.2 Robust Completeness

As demonstrated in Example 4.4, the exact N-step backward reachable set $\mathrm{Rch}_N^\delta(\Omega)$ are often difficult to obtain even for linear systems with polyhedral target set and state, control constraints. To deal with such a difficulty, especially for nonlinear dynamics and a general target set without particular shape, we resort to its approximation $\widehat{\mathrm{Rch}}_N^\delta(\Omega)$, which is based on an approximation $\widehat{\mathrm{Pre}}$ of Pre:

$$\widehat{\mathrm{Rch}}(Y) = \widehat{\mathrm{Pre}}(Y) \cup Y, \quad Y \subseteq \mathbb{X}.$$

Replacing Rch^δ in (4.14), we obtain the following modified algorithm, which can be proved to yield a sound and robustly complete reachability control strategy.

$$\begin{cases} \widehat{\mathrm{Rch}}_0(\Omega) = \Omega, \\ \widehat{\mathrm{Rch}}_j(\Omega) = \widehat{\mathrm{Rch}}(\widehat{\mathrm{Rch}}_{j-1}(\Omega)). \end{cases} \tag{4.19}$$

Similar to invariance control synthesis, we can also achieve sound and robustly complete control synthesis for reachability problems.

Theorem 4.2 (Soundness and Robust Completeness)**.** Let $\Omega \subseteq \mathbb{X}$ be an open set. If $\widehat{\mathrm{Pre}}(Y)$ in (4.19) is an approximation of $\mathrm{Pre}(Y)$ that satisfies Assumption 3.3 for some $\delta > 0$. Assume that $\bigcup_{j=0}^{\infty} \widehat{\mathrm{Rch}}_j(\Omega) \neq \emptyset$. Then (4.19) gives

$$\mathcal{BR}_\infty^\delta(\Omega) \subseteq \bigcup_{j=0}^{\infty} \widehat{\mathrm{Rch}}_j(\Omega) \subseteq \mathcal{BR}_\infty(\Omega). \tag{4.20}$$

Proof. By Proposition 4.5, we have $\mathcal{BR}_\infty^\delta(\Omega) = \bigcup_{j=0}^{\infty} \mathrm{Rch}_j^\delta(\Omega)$ and $\mathcal{BR}_\infty(\Omega) = \bigcup_{j=0}^{\infty} \mathrm{Rch}_j(\Omega)$. To prove (4.20), we only need to show $\mathrm{Rch}_j^\delta(\Omega) \subseteq \widehat{\mathrm{Rch}}_j(\Omega) \subseteq \mathrm{Rch}_j(\Omega)$ for all $j \in \mathbb{N}$.

For $j = 0$, we have $\mathrm{Rch}_0^\delta(\Omega) = \widehat{\mathrm{Rch}}_0(\Omega) = \mathrm{Rch}_0(\Omega) = \Omega$. For $j = 1$,

$$\begin{aligned} \mathrm{Rch}_1^\delta(\Omega) &= \mathrm{Pre}^\delta(\mathrm{Rch}_0^\delta(\Omega)) \cup \mathrm{Rch}_0^\delta(\Omega) = \mathrm{Pre}^\delta(\Omega) \cup \Omega, \\ \widehat{\mathrm{Rch}}_1(\Omega) &= \widehat{\mathrm{Pre}}(\widehat{\mathrm{Rch}}_0(\Omega)) \cup \widehat{\mathrm{Rch}}_0(\Omega) = \widehat{\mathrm{Pre}}(\Omega) \cup \Omega, \\ \mathrm{Rch}_1(\Omega) &= \mathrm{Pre}(\mathrm{Rch}_0(\Omega)) \cup \mathrm{Rch}_0(\Omega) = \mathrm{Pre}(\Omega) \cup \Omega. \end{aligned}$$

By (3.12), we can conclude $\mathrm{Rch}_1^\delta(\Omega) \subseteq \widehat{\mathrm{Rch}}_1(\Omega) \subseteq \mathrm{Rch}_1(\Omega)$. Assume that $\mathrm{Rch}_j^\delta(\Omega) \subseteq \widehat{\mathrm{Rch}}_j(\Omega) \subseteq \mathrm{Rch}_j(\Omega)$ for some $j \in \mathbb{Z}^+$. Then

$$\begin{aligned}\mathrm{Rch}_{j+1}^\delta(\Omega) &= \mathrm{Pre}^\delta(\mathrm{Rch}_j^\delta(\Omega)) \cup \mathrm{Rch}_j^\delta(\Omega)\\ &\subseteq \widehat{\mathrm{Pre}}(\widehat{\mathrm{Rch}}_j(\Omega)) \cup \widehat{\mathrm{Rch}}_j(\Omega) = \widehat{\mathrm{Rch}}_{j+1}(\Omega)\\ &\subseteq \mathrm{Pre}(\mathrm{Rch}_j(\Omega)) \cup \mathrm{Rch}_j(\Omega) = \mathrm{Rch}_{j+1}(\Omega).\end{aligned}$$

Hence, $\mathrm{Rch}_{j+1}^\delta(\Omega) \subseteq \widehat{\mathrm{Rch}}_{j+1}(\Omega) \subseteq \mathrm{Rch}_{j+1}(\Omega)$. The proof is now complete by induction. □

The relation (4.20) indicates that we can obtain an inner approximation of the maximal backward reachable set with a lower bound of δ-robustly backward reachable set if the approximations of predecessors can be made sufficiently precise so that Assumption 3.12 is satisfied.

If we consider the reachability control objective as an LTL formula φ_{r}, the maximal (δ-robustly) backward reachable set is essentially the winning set of system $\mathcal{S}$ with respect to φ_{r}. Then we can solve Problem 2.2 by using algorithm (4.19).

Corollary 4.2. Given system $\mathcal{S}^0$ with the labeling function (4.2) where $\Omega \subseteq \mathbb{X}$ is an open target set and an reachability specification $\varphi_{\mathrm{r}} = \Diamond G$, consider algorithm (4.19) under Assumption 3.3 for some $\delta > 0$.

(i) If $\bigcup_{j=0}^{\infty} \widehat{\mathrm{Rch}}_j(\Omega) \neq \emptyset$, then we have the reachability control strategy for $\mathcal{S}^0$:

$$\kappa(x) = \begin{cases} \Pi_{\widehat{\mathrm{Rch}}_j(\Omega)}(x) & \forall x \in \widehat{\mathrm{Rch}}_{j+1}(\Omega) \setminus \widehat{\mathrm{Rch}}_j(\Omega), \quad j = 0, 1, \dots \\ \mathbb{U} & \forall x \in \Omega = \widehat{\mathrm{Rch}}_0(\Omega). \end{cases} \tag{4.21}$$

(ii) If $\bigcup_{j=0}^{\infty} \widehat{\mathrm{Rch}}_j(\Omega) = \emptyset$, then Ω is not δ-robustly reachable for system $\mathcal{S}^0$.

Proof. For (ii), if $\bigcup_{j=0}^{\infty} \widehat{\mathrm{Rch}}_j(\Omega) = \emptyset$, then $\mathrm{Win}_{\mathcal{S}}^\delta(\varphi_{\mathrm{r}}) = \emptyset$ for $\delta > 0$ because $\mathrm{Win}_{\mathcal{S}}^\delta(\varphi_{\mathrm{r}}) = \mathcal{BR}_\infty^\delta(\Omega) \subseteq \bigcup_{j=0}^{\infty} \widehat{\mathrm{Rch}}_j(\Omega)$. Hence, Ω is not δ-robustly reachable for system $\mathcal{S}^0$.

We now consider (i). For $x \in \widehat{\mathrm{Rch}}_0(\Omega)$, the formula φ_{r} is always true and hence $\kappa(x) = \mathbb{U}$ for all $x \in \Omega$. The sequence $\left\{\mathrm{Rch}_j^\delta(\Omega)\right\}_{j=0}^{\infty}$ is increasing. Assume that $\kappa(x)$ can successfully achieve the reachability property eventually for any $x \in \widehat{\mathrm{Rch}}_j(\Omega)$ for some $j \in \mathbb{N}$. Then for $x \in \widehat{\mathrm{Rch}}_{j+1}(\Omega) \setminus \widehat{\mathrm{Rch}}_j(\Omega)$, any control value $u \in \Pi_{\widehat{\mathrm{Rch}}_j(\Omega)}(x)$ will steer system state into $\widehat{\mathrm{Rch}}_j(\Omega)$ under any disturbance in set $\mathbb{D}$. This implies that $\kappa(x)$ also realizes φ_{r} for any $x \in \widehat{\mathrm{Rch}}_{j+1}(\Omega)$. Therefore, (i) is also proved. □

4.3 Reach-and-Stay Control

Same as the previous two sections, we give the formal definition of the reach-and-stay property as follows first.

Definition 4.9. Let Ω be a subset of the state space $\mathbb{X}$ of system $\mathcal{S}$ with the labeling function (4.2). A *reach-and-stay* property of a solution $\mathbf{x} = \{x_i\}_{i=0}^{\infty}$ of system $\mathcal{S}$ with respect to Ω requires that there exists some $j \in \mathbb{N}$ such that $x_k \in \Omega$ for all $k \geq j$ and $k \in \mathbb{N}$, written as $\varphi_{\text{rs}} = \Diamond\Box G$ in form of a LTL formula.

Intuitively, reach-and-stay property is a combination of reachability and invariance. A control strategy that can control the system state to reach a controlled invariant set would serve this purpose. Such an idea for solving the reach-and-stay problem was first proposed in [16] and is shown as the following algorithm.

$$\begin{cases} \left.\begin{array}{l} X_0 = \Omega \\ X_{i+1} = \text{Pre}^{\delta}(X_i | X_i) \end{array}\right\} X_\infty \triangleq \bigcap_{i=0}^{\infty} X_i \\ \kappa(x) = \Pi_{X_\infty}(x), \forall x \in X_\infty \\ \left.\begin{array}{l} Z_0 = X_\infty \\ Z_{i+1} = \text{Pre}^{\delta}(Z_i) \\ \kappa(z) = \Pi_{Z_{i+1}}(z), \forall z \in Z_{i+1} \setminus Z_i \end{array}\right\} Z_\infty \triangleq \bigcup_{i=0}^{\infty} Z_i \end{cases} \tag{4.22}$$

Algorithm (4.22) is composed of two sequential fixed-point iterations. The completeness of (4.22) relies on the assumption that the target set is compact and convex. For general dynamics and compact target set without this assumption, (4.22) fails to yield the real winning set, which can be illustrated by the following example.

Example 4.6. Consider a target set $\Omega = [-0.3, 0.3] \cup [0.8, 1.1]$ and the dynamics

$$x_{t+1} = -x_t(x_t^2 - 2.05x_t + 0.05) + u_t + d_t, \tag{4.23}$$

where $x_t \in \mathbb{X} = [-0.65, 1.1]$, $u_t \in \mathbb{U} = \{0, 10\}$, and $d_t \in \mathbb{D} = [-5, 5] \times 10^{-4}$ ($\delta = 5 \times 10^{-4}$) for $t \in \mathbb{N}$.

For all state $x \in \mathbb{X}$, using the control value 10 will make the state in the next time step out of domain $\mathbb{X}$. Let $u_t = 0$ for all t. There are 3 fixed points 0, 1, and 1.05. The fixed points 0 and 1.05 are stable while 1 is unstable.

As shown in Figure 4.3, the interval $O = [-0.65, -0.6311)$ cannot be controlled inside the state space $\mathbb{X}$ under arbitrary disturbance, because for all $x_t \in O$ there exists $d \in \mathbb{D}$ such that

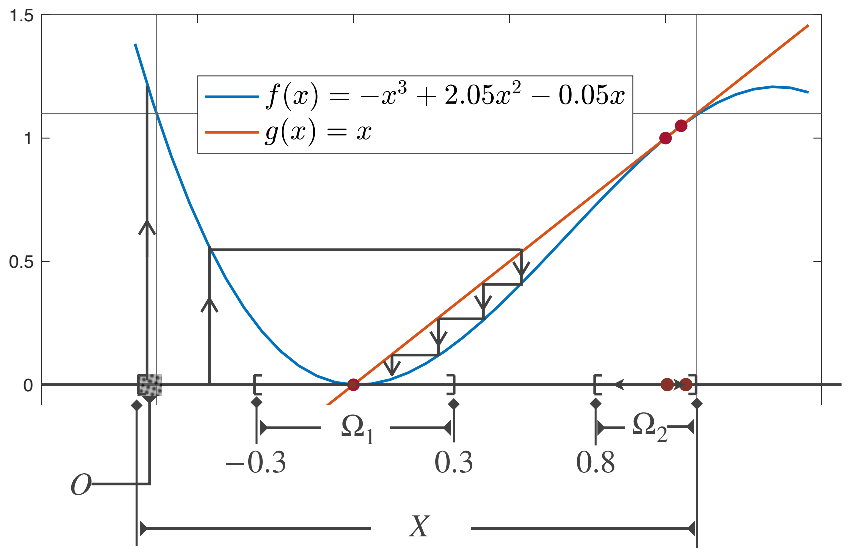


Figure 4.3: The evolution of the state of system (4.23) without disturbance term.

the system state at the next time step $x_{t+1} > 1.1$. For the nominal system of (4.23), system state x evolves to 0 for all $x \in (0, 1)$.

The target set is a union of two disconnected intervals $\Omega_1 = [-0.3, 0.3]$ and $\Omega_2 = [0.8, 1.1]$. The set Ω_1 is δ-robustly controlled invariant since $\Omega_1 \subseteq \mathrm{Pre}^{\delta}(\Omega_1) = [-0.3435, 0.4483]$. Since $x = 0.9914$ satisfies $-x(x^2 - 2.05x + 0.05) + d = x$ and the difference $x_t - x_{t+1}$, which is negative, between two sequential states is decreasing as x increases between 0.3414 and 1.0253, any state $x \in [0.3, 0.9914)$ can be controlled inside Ω_1. Because of the overlap between Ω_1 and $[0.3, 0.9914)$, we can see that $[-0.3, 1.1] \subseteq \mathrm{Win}^{\delta}_{\mathcal{S}}(\varphi_{\mathrm{rs}})$. In addition, $x_{t+1} \in (0, 1.1]$ for any state $x_t \in [-0.6311, 0)$. Hence, the real winning set is

$$\mathrm{Win}^{\delta}_{\mathcal{S}}(\varphi_{\mathrm{rs}}) = [-0.6311, 1.1].$$

However, algorithm (4.22) computes the maximal controlled invariant set inside Ω, i.e., $X_\infty = [-0.3, 0.3] \cup [1.0370, 1.1]$ firstly and finally

$$Z_\infty = [-0.6311, -0.6082) \cup (-0.6021, 0.9914) \cup (1.0135, 1.1].$$

Because X_∞ is a union of two disconnected intervals $[-0.3, 0.3]$ and $[1.0370, 1.1]$ with an unstable fixed point 1 in between, the interval $[0.9914, 1.0135]$ is not included in Z_∞. The interval $[-0.6082, -0.6021]$ is also missing because it is mapped to $[0.9914, 1.0135]$ by (4.23).

As opposed to (4.22), we now present the following algorithm (4.24) for reach-and-stay control synthesis, which consists of two nested fixed-point iterations.

$$\begin{cases} Y_0 = \emptyset, X_0^{\infty} = \emptyset \\ \left.\begin{aligned} &X_{i+1}^0 = Y_i \cup \Omega \\ &X_{i+1}^{j+1} = \mathrm{Pre}^{\delta}(X_{i+1}^j | X_{i+1}^j) \end{aligned}\right\} X_{i+1}^{\infty} \triangleq \bigcap_{j=0}^{\infty} X_{i+1}^j \\ \kappa(x) = \Pi_{X_{i+1}^{\infty}}(x), \forall x \in \Omega \cap \left(X_{i+1}^{\infty} \setminus X_i^{\infty}\right) \\ Y_{i+1} = \mathrm{Pre}^{\delta}(X_{i+1}^{\infty}) \\ \kappa(y) = \Pi_{X_{i+1}^{\infty}}(y), \forall y \in Y_{i+1} \setminus (Y_i \cup \Omega) \end{cases} \tag{4.24}$$

In the next proposition, we will show that the winning set $\mathrm{Win}_{\mathcal{S}}(\varphi_{\mathrm{rs}})$ can be obtained by using algorithm (4.24). The memoryless control strategy κ constructed along with the winning set computation suffices to realize the reach-and-stay objective.

Proposition 4.6. Consider an LTL formula $\varphi_{\mathrm{rs}} = \Diamond\Box G$ for system $\mathcal{S}$ with labeling functions (4.2). Suppose that $\Omega \subseteq \mathbb{X}$ is compact and Assumption 3.1 or 3.2 holds. Let $Y_\infty = \bigcup_{i=0}^{\infty} Y_i$ be a fixed point of (4.24). Then,

(i) $Y_\infty = \mathrm{Win}_{\mathcal{S}}^{\delta}(\varphi_{\mathrm{rs}})$ where $\delta \geq 0$ is the bound of disturbances, and

(ii) The strategy κ as defined in (4.24) is a memoryless control strategy that realizes φ_{rs}.

Proof. We only consider $\Omega \neq \emptyset$. Otherwise the results trivially hold.

We first show $Y_\infty \subseteq \mathrm{Win}_{\mathcal{S}}^{\delta}(\varphi_{\mathrm{rs}})$ by induction. Trivially $Y_0 = \emptyset \subseteq \mathrm{Win}_{\mathcal{S}}^{\delta}(\varphi_{\mathrm{rs}})$. The induction step aims to show that, for all $i \in \mathbb{Z}^+$, $Y_{i+1} \subseteq \mathrm{Win}_{\mathcal{S}}^{\delta}(\varphi_{\mathrm{rs}})$ if $Y_i \subseteq \mathrm{Win}_{\mathcal{S}}^{\delta}(\varphi_{\mathrm{rs}})$. Assume that X_i^{∞} is compact. Then $Y_i = \mathrm{Pre}^{\delta}(X_i^{\infty})$ and thus $X_{i+1}^0 = \Omega \cup Y_i$ is compact. The sequence $\{X_{i+1}^j\}_{j=0}^{\infty}$ is compact and decreasing by induction, using Proposition 3.4 (i) and Proposition 3.5 since $X_{i+1}^0 = \Omega \cup Y_i$ is compact. It is also easy to show that $\{Y_i\}_{i=0}^{\infty}$ is increasing by induction. Furthermore, by Proposition 3.2, we have

$$X_{i+1}^{\infty} = \lim_{j\to\infty} X_{i+1}^j = \bigcap_{j=0}^{\infty} X_{i+1}^j,$$

which is the maximal controlled invariant set inside $\Omega \cup Y_i$ by Proposition 4.3 and compact. If $Y_i \subseteq \mathrm{Win}_{\mathcal{S}}^{\delta}(\varphi_{\mathrm{rs}})$, then $X_{i+1}^{\infty} \subseteq \mathrm{Win}_{\mathcal{S}}^{\delta}(\varphi_{\mathrm{rs}})$ because X_{i+1}^{∞} is a controlled invariant set inside $\Omega \cup Y_i$, which gives

$$Y_{i+1} = \mathrm{Pre}^{\delta}(X_{i+1}^{\infty}) \subseteq \mathrm{Win}_{\mathcal{S}}^{\delta}(\varphi_{\mathrm{rs}})$$

by Definition 3.3. Hence, $Y_\infty = \bigcup_{i=0}^{\infty} Y_i \subseteq \text{Win}_{\mathcal{S}}^{\delta}(\varphi_{\text{rs}})$.

To see $\text{Win}_{\mathcal{S}}^{\delta}(\varphi_{\text{rs}}) \subseteq Y_\infty$, we aim to show that $x \notin \text{Win}_{\mathcal{S}}^{\delta}(\varphi_{\text{rs}})$ for all $x \notin Y_\infty$. Let $x \notin Y_\infty$ be arbitrary. Then $x \notin Y_\infty = \text{Pre}^{\delta}(\mathcal{I}_\infty(Y_\infty \cup \Omega))$, where $\mathcal{I}_\infty(Y_\infty \cup \Omega)$ is the maximal controlled invariant set inside $Y_\infty \cup \Omega$, since Y_∞ is a fixed point of (4.24). This means that for all $\{u_t\}_{t=0}^{\infty}$ there exists k and $\{d_t\}_{t=0}^{k}$ such that the resulting sequence of $\mathcal{S}$ satisfies $x_k \notin (\Omega \cup Y_\infty)$. Since $x_k \notin Y_\infty$, we can show in the same manner that for all $\{u_t\}_{t=k}^{\infty}$ there exists $k' \geq k$ and $\{d_t\}_{t=k}^{k'}$ such that the k'th state $x_{k'}$ of the resulting solution satisfies $x_{k'} \notin (\Omega \cup Y_\infty)$. In this way, for all $\{u_t\}_{t=0}^{\infty}$, we can find an infinite sequence $\{d_t\}_{t=0}^{\infty}$ for any $x \notin Y_\infty$ so that the resulting solution of $\mathcal{S}$ goes outside of Ω infinitely often. Hence, $x \notin \text{Win}_{\mathcal{S}}^{\delta}(\varphi_{\text{rs}})$, which shows $\text{Win}_{\mathcal{S}}^{\delta}(\varphi_{\text{rs}}) \subseteq Y_\infty$.

Now we prove (ii). The control strategy κ is constructed by $\Pi^{\delta}_{X_{i+1}^{\infty}}$, which is only dependent on the current state x of the system $\mathcal{S}$, and thus κ is memoryless. By the definition of $\Pi^{\delta}_{X_{i+1}^{\infty}}$ in (3.4), for all $x \in \Omega \cap \left(X_{i+1}^{\infty} \setminus X_i^{\infty}\right)$ and $x \in Y_{i+1} \setminus (Y_i \cup \Omega)$ ($i \in \mathbb{N}$), the state x will be controlled inside $\Omega \cup Y_i$ and Y_i in one step, respectively. That means any state $x \in Y_{i+1}$ will be controlled into Y_i until it enters $X_1^{\infty} = \mathcal{I}_\infty(\Omega) \subseteq \Omega$, which is controlled invariant. Hence, we have also shown that κ realizes φ_{rs}.

The proof is now complete. □

The major difference between (4.22) and (4.24) is that the information of the target set Ω is used for every iteration of Y_i in (4.24) while such information is lost after the computation of the maximal (robustly) controlled invariant set X_∞ in (4.22). Hence, some of the system states inside Ω, which would leave Ω but will be controlled back to X_∞ or stay inside Ω under some disturbance, will be missing if we use (4.22). The real winning set in Example 4.6 can be obtained by using algorithm (4.24).

Remark 4.1. The completeness result in Proposition 4.6, i.e., Y_∞ captures all the states that can be controlled to stay in Ω ($\text{Win}_{\mathcal{S}}^{\delta}(\varphi_{\text{rs}}) \subseteq Y_\infty$), relies on the assumption that Y_∞ is a fixed point of (4.24). To satisfy such an assumption, the following properties regarding Ω and the predecessor map are required:

$$\text{Pre}^{\delta}\left(\bigcup_{i=0}^{\infty} Y_i\right) = \bigcup_{i=0}^{\infty} \text{Pre}^{\delta}(Y_i), \tag{4.25}$$

$$\left(\bigcup_{i=0}^{\infty} \text{Pre}^{\delta}(Y_i \cup \Omega)\right) \cap \left(\bigcup_{i=0}^{\infty} (Y_i \cup \Omega)\right) = \bigcup_{i=0}^{\infty} \left(\text{Pre}^{\delta}(Y_i \cup \Omega) \cap (Y_i \cup \Omega)\right). \tag{4.26}$$

However, the condition (4.25) does not generally hold.

There are some computational redundancies in (4.24): the sequence $\{Y_i\}_{i=0}^{\infty}$ is increasing and so is $\{X_i^j\}_{i=0}^{\infty}$ for all $j \in \mathbb{N}$. Hence, it is only necessary to compute the incremental parts between two adjacent sets in the sequences. Also, considering that predecessors cannot be precisely computed, we present the approximated control synthesis algorithm (4.27).

$$\begin{cases} \widehat{Y}_0 = \widehat{X}_0^{\infty} = \emptyset, V_0 = X \setminus \Omega \\ \left.\begin{array}{l} W_i^0 = \Omega \setminus \widehat{Y}_i \\ \widehat{X}_{i+1}^j = \widehat{Y}_i \cup W_i^j \\ W_i^{j+1} = \widehat{\mathrm{Pre}}(\widehat{X}_{i+1}^j | W_i^j) \end{array}\right\} \begin{array}{l} W_i^{\infty} \triangleq \bigcap_{j=0}^{\infty} W_i^j \\ \widehat{X}_{i+1}^{\infty} \triangleq \bigcap_{j=0}^{\infty} \widehat{X}_{i+1}^j \end{array} \\ \kappa(x) \leftarrow \Pi_{X_{i+1}^{\infty}}(x), \forall x \in W_i^{\infty} \\ Z_i = \widehat{\mathrm{Pre}}(\widehat{X}_{i+1}^{\infty} | V_i) \\ \kappa(x) \leftarrow \Pi_{X_{i+1}^{\infty}}(x), \forall x \in Z_i \\ V_{i+1} = V_i \setminus Z_i \\ \widehat{Y}_{i+1} = X_{i+1}^{\infty} \cup Z_i \end{cases} \tag{4.27}$$

Theorem 4.3 (Soundness and Robust Completeness)**.** Consider an LTL formula $\varphi_{\mathrm{rs}} = \Diamond\Box G$ for system $\mathcal{S}$ with the labeling function (4.2). Let assumptions in Proposition 4.6 hold. Suppose that $\widehat{\mathrm{Pre}}$ in algorithm (4.27) satisfies Assumption 3.3 for some $\delta > 0$. Let $\widehat{Y}_{\infty} = \bigcup_{i=0}^{\infty} \widehat{Y}_i$. Then

$$\mathrm{Win}_{\mathcal{S}}^{\delta}(\varphi_{\mathrm{rs}}) \subseteq \widehat{Y}_{\infty} \subseteq \mathrm{Win}_{\mathcal{S}}(\varphi_{\mathrm{rs}}). \tag{4.28}$$

Proof. Let $\{\widetilde{X}_i^{\infty}\}$ ($\{\widetilde{X}_i^{r\infty}\}$) and $\{\widetilde{Y}_i\}$ ($\{\widetilde{Y}_i^r\}$) be the sequences of sets generated by algorithm (4.27) with $\widehat{\mathrm{Pre}} = \mathrm{Pre}$ ($\widehat{\mathrm{Pre}} = \mathrm{Pre}^r$). And also to simplify notation, we denote by Y_{∞} and Y_{∞}^{δ} ($\delta > 0$) the outputs of (4.24) with operator Pre and Pre^{δ}, respectively. We prove the theorem in the following two steps: (i) Show that (4.27) is equivalent to (4.24) when set computation is accurate, i.e., $\widetilde{X}_i^{\infty} = X_i^{\infty}$ ($\widetilde{X}_i^{r\infty} = X_i^{r\infty}$) and $\widetilde{Y}_i = Y_i$ ($\widetilde{Y}_i^r = Y_i^r$) for all $i \in \mathbb{N}$. (ii) Show $\widetilde{Y}_{\infty}^r \subseteq \widehat{Y}_{\infty} \subseteq \widetilde{Y}_{\infty}$ under the given condition.

First of all, we show that $Y_i \subseteq \mathrm{Pre}(Y_i)$ for all i. Since X_i^{∞} is a controlled invariant set, $X_i^{\infty} \subseteq \mathrm{Pre}(X_i^{\infty})$. By the definition of Y_i in (4.24) and monotonicity of Pre, $Y_i = \mathrm{Pre}(X_i^{\infty}) \subseteq \mathrm{Pre}(\mathrm{Pre}(X_i^{\infty})) = \mathrm{Pre}(Y_i)$. We now prove (i) by induction. The base case clearly holds since

$\widetilde{Y}_0 = Y_0 = \widetilde{X}_0^\infty = X_0^\infty = \emptyset$. Suppose that $\widetilde{X}_i^\infty = X_i^\infty$ and $\widetilde{Y}_i = Y_i$ for some $i \in \mathbb{Z}^+$. Then

$$\begin{aligned}
\widetilde{X}_{i+1}^0 &= \widetilde{Y}_i \cup (\Omega \setminus \widetilde{Y}_i) = \widetilde{Y}_i \cup \Omega = X_{i+1}^0, \\
\widetilde{X}_{i+1}^{j+1} &= \widetilde{Y}_i \cup W_i^{j+1} = \widetilde{Y}_i \cup (\mathtt{Pre}(\widetilde{X}_{i+1}^j) \cap W_i^j) \\
&= (\widetilde{Y}_i \cup \mathtt{Pre}(\widetilde{X}_{i+1}^j)) \cap (\widetilde{Y}_i \cup W_i^j) \\
&= (\widetilde{Y}_i \cup \mathtt{Pre}(\widetilde{X}_{i+1}^j)) \cap \widetilde{X}_{i+1}^j.
\end{aligned}$$

Also, $\mathtt{Pre}(\widetilde{X}_{i+1}^j) = \mathtt{Pre}(\widetilde{Y}_i \cup W_i^j) \supseteq \mathtt{Pre}(\widetilde{Y}_i) \supseteq \widetilde{Y}_i$, which implies that $\widetilde{X}_{i+1}^{j+1} = \mathtt{Pre}(\widetilde{X}_{i+1}^j) \cap \widetilde{X}_{i+1}^j$. This is the same as the iteration step in (4.24), and thus $\widetilde{X}_{i+1}^\infty = X_{i+1}^\infty$. Now consider the sequence $\{V_i\}_{i=0}^\infty$. We have $V_0 = X \setminus \Omega$ and

$$V_{i+1} = V_i \setminus (\mathtt{Pre}(\widetilde{X}_i^\infty) \cap V_i) = V_i \setminus \mathtt{Pre}(\widetilde{X}_i^\infty).$$

Unfolding V_i until V_0 and using that $\mathtt{Pre}(\widetilde{X}_i^\infty) \subseteq \mathtt{Pre}(\widetilde{X}_{i+1}^\infty)$, we can derive

$$V_i = X \setminus (\Omega \cup \mathtt{Pre}(\widetilde{X}_i^\infty)) = X \setminus (\Omega \cup Y_i) = X \setminus \widetilde{X}_{i+1}^0.$$

Then

$$\begin{aligned}
\mathtt{Pre}(\widetilde{X}_{i+1}^\infty) &= \mathtt{Pre}(\widetilde{X}_{i+1}^\infty) \cap (\widetilde{X}_{i+1}^0 \cup V_i) \\
&= \left[\mathtt{Pre}(\widetilde{X}_{i+1}^\infty) \cap \widetilde{X}_{i+1}^0\right] \cup \left[\mathtt{Pre}(\widetilde{X}_{i+1}^\infty) \cap V_i\right] \\
&= \widetilde{X}_{i+1}^\infty \cup \mathtt{Pre}(\widetilde{X}_{i+1}^\infty | V_i) = \widetilde{Y}_{i+1}.
\end{aligned} \tag{4.29}$$

The equality $\mathtt{Pre}(\widetilde{X}_{i+1}^\infty) \cap \widetilde{X}_{i+1}^0 = \widetilde{X}_{i+1}^\infty$ can be seen by contradiction. If there exists $A \subseteq \widetilde{X}_{i+1}^0 \setminus \widetilde{X}_{i+1}^\infty$ such that $A \subseteq \mathtt{Pre}(\widetilde{X}_{i+1}^\infty)$ then $\widetilde{X}_{i+1}^\infty \cup A \subseteq \mathtt{Pre}(\widetilde{X}_{i+1}^\infty \cup A)$, which indicates $A \cup \widetilde{X}_{i+1}^\infty$ is a larger controlled invariant set inside $\widetilde{X}_{i+1}^0$, but $\widetilde{X}_{i+1}^\infty$ is the maximal one. Therefore $Y_{i+1} = \widetilde{Y}_{i+1}$. The above argument also applies to prove $\widetilde{X}_i^{r\infty} = X_i^{r\infty}$ and $\widetilde{Y}_i^r = Y_i^r$.

To prove (ii), we aim to show $X_i^{r\infty} \subseteq \widehat{X}_i^\infty \subseteq X_i^\infty$ and $Y_i^r \subseteq \widehat{Y}_i \subseteq Y_i$ for all i. Clearly

$$\begin{gathered}
X_1^{r0} = \widehat{X}_1^0 = X_1^0 = \Omega, \text{ and} \\
\mathtt{Pre}^r(X_1^{r0} | W_1^0) \subseteq \widehat{\mathtt{Pre}}(\widehat{X}_1^0 | W_1^0) \subseteq \mathtt{Pre}(X_1^0 | W_1^0)
\end{gathered}$$

by $\mathtt{Pre}^r(X) \subseteq \widehat{\mathtt{Pre}}(X) \subseteq \mathtt{Pre}(X)$ and Proposition 3.4 (ii). This means $X_1^{r1} \subseteq \widehat{X}_1^1 \subseteq X_1^1$. By induction, we can easily achieve $X_1^{rj} \subseteq \widehat{X}_1^j \subseteq X_1^j$ for any $j \in \mathbb{N}$. Thus $X_1^{r\infty} \subseteq \widehat{X}_1^\infty \subseteq X_1^\infty$. As shown in (4.29), $\widehat{Y}_i = \widehat{\mathtt{Pre}}(\widehat{X}_i^\infty)$. Then

$$Y_1^r = \mathtt{Pre}^r(X_1^{r\infty}) \subseteq \mathtt{Pre}^r(\widehat{X}_1^\infty) \subseteq \widehat{Y}_1 \subseteq \mathtt{Pre}(\widehat{X}_i^\infty) \subseteq \mathtt{Pre}(X_i^\infty) = Y_1.$$

Therefore, (ii) can also be shown using induction. □

Similar to the previous two basic LTL formulas, we can arrive at the following robust completeness result for Problem 2.2.

Corollary 4.3. Given system $\mathcal{S}^0$ with the labeling function (4.2) where $\Omega \subseteq \mathbb{X}$ is compact and a reach-and-stay specification $\varphi_{\text{rs}} = \Diamond\Box G$, consider algorithm (4.27) under Assumption 3.3 for some $\delta > 0$.

(i) If $\widehat{Y}_\infty = \bigcup_{i=0}^{\infty} \widehat{Y}_i \neq \emptyset$, then the memoryless control strategy κ given in (4.27) realizes φ_{rs} at all $x \in \widehat{Y}_\infty$.

(ii) If $\widehat{Y}_\infty = \bigcup_{i=0}^{\infty} \widehat{Y}_i = \emptyset$, then there is no state within the state space $\mathbb{X}$ that is guaranteed to be controlled to Ω and stay there for all future time for system $\mathcal{S}^0$ with δ-bounded disturbance.

4.4 Summary

In this chapter, we revisited the traditional regulation problem from a set-theoretic point of view. Without assuming any form of the system dynamics nor stability properties, fixed-point algorithms for solving invariance, reachability, and reach-and-stay control problems were presented, which have formal guarantee of the correctness of the resulting control strategies. These algorithms all construct memoryless control strategies during the computation of winning sets.

For the invariance control problem, we showed that control synthesis can be sound and complete provided that the target set is compact and the computation of predecessors is precise. It has been addressed in [11] that the control synthesis is essentially a fixed-point algorithm to compute the maximal controlled invariant set inside the given target set. Research in this topic focused on linear discrete-time systems [18, 55, 117] because the numerical determination of maximal controlled invariant sets is not easy even for linear systems (see also Example 4.1). The proposed robustness margin is an extension of the λ-*contractivity* of the linear systems around a compact and convex set in [18] to general nonlinear systems. Another difference between our result and [18] is that we derived the sufficient condition for the approximation of predecessors so that the control synthesis can be sound and robustly complete while λ-contractivity is shown as a requirement for finite termination in [18]. The results for invariance control in this chapter have been published in [80, 78, 79]. In Chapter 5, we will illustrate the set approximation technique that can satisfy such a condition.

Similar to the invariance control problem, reachability control has been studied using set theory since 70's with fixed time horizon [13]. The *domain of attraction* to a target set is studied in [16] in terms of infinite horizon while [67] focuses on finite-time reachability. In this thesis, we do not fix a maximum reach time. For uncertain nonlinear systems, we proposed a sound and complete control synthesis algorithm (4.14) with respect to an open target set, which implies that reachability control is actually a dual of the invariance control. Closely related to our work is the research in [16, 102], where the reach time is also not given but to be minimized by solving minimum-time optimal control problems. Particularly in [16], a recursion that similar to our algorithm (4.14) is given as

$$Y_0 = \Omega, \quad Y_{i+1} = \mathrm{Pre}^{\delta}(Y_i). \tag{4.30}$$

The slight difference between (4.14) and (4.30) is that the set Y_{i+1} in (4.30) is computed only based on the previous set Y_i without considering the given initial set $Y_0 = \Omega$ for any $i \in \mathbb{Z}^+$. Hence, the set Y_N, which is called the *controllability set in N-steps* in [16] (or similar names as in [67, Definition 2.9] and [102, Definition 2.1]) is the set of states that are guaranteed to be controlled into a given target set in exactly N steps. Our N-step robustly backward reachable set (see Definition 4.8) obtained by (4.14) allows the uncertainty in the actual reach time and hence captures a larger set by considering the uncertainty of reach time.

Given that Ω is any subset of $\mathbb{X}$, Proposition 4.5 shows that the set $\bigcup_{i=0}^{\infty} \mathrm{Rch}_N^{\delta}(\Omega)$, where $\mathrm{Rch}_N^{\delta}(\Omega)$ $(i \in \mathbb{N})$ is obtained by (4.14), is equal to the real winning set with respect to the reachability specification for disturbed system $\mathcal{S}$ if Ω is open. We consider this result as one of our contributions since it has not been proved in the literature.

As an answer to the third question raised at the beginning of this chapter, we also showed that performing reachability control to a controlled invariant set inside the given target set, which is first proposed in [16] and presented as algorithm (4.22) in Section 4.3, is sound but not complete for solving reach-and-stay problems for systems with uncertainties. To improve on the work [16], we provided algorithm (4.24) and showed its completeness under the assumption that algorithm (4.24) returns a fixed point of (4.24). Related results on reach-and-stay control synthesis for switched systems is published in [81] and an improved version can be found in [82].

Considering that predecessors are usually difficult to compute precisely, we also analyzed the effects to the determination of winning sets by using inner approximations. Compactness and convexity are strong properties that make set computation practical [18, 16, 67, 56, 117]. In this chapter, we relaxed these assumptions and showed that if the approximation of predecessors can satisfy Assumption 3.3 in Chapter 3, then the control synthesis with respect to all three basic specifications is at least robustly complete.

A problem with (4.8), (4.19), and (4.27) is that they are not guaranteed to terminate in a finite number of steps under current assumptions. Suppose that $\{Y_i\}_{i=0}^{\infty}$ is the sequence of sets generated by (4.5). It is not necessary that we can always find an $N \in \mathbb{Z}^+$ so that $\widehat{\mathrm{Pre}}(Y_N|Y_N) = Y_N$, even if a proper approximation $\widehat{\mathrm{Pre}}$ of Pre can be implemented to satisfy Assumption 3.3. We will answer this question in Chapter 5 by giving a finitely terminating algorithm that proceeds by set approximations.

Chapter 5

Robust Completeness via Interval Analysis

We have discussed in Chapter 4 the robustly complete control synthesis algorithms for control problems with respect to invariance, reachability and reach-and-stay specifications. Implementation of these algorithms relies on a concrete method for the approximation of predecessors that satisfies Assumption 3.3, which requires the approximation error to be bounded from both below and above.

The questions left open in Chapter 4 are:

- *What is an efficient approximation $\widehat{Pre}$ of the predecessor Pre that satisfies Assumption 3.3?*
- *Does the use of $\widehat{Pre}$ guarantee the control synthesis algorithms with respect to invariance and reachability specifications finitely terminating?*

In this chapter, we present an *interval* implementation of the control synthesis algorithms with respect to the specifications discussed in Chapter 4. We use unions of interval vectors (or intervals) in the $\mathbb{R}^n$ space to approximate any compact set A in the state space of the system. The approximation of the predecessor $\mathrm{Pre}^{\delta}(A)$ of the set A is also a union of intervals, which can be obtained by solving a Constraint-Satisfaction Problem (CSP) with interval computation. In this way, the infinite state space $\mathbb{X}$ of system $\mathcal{S}$ is discretized into a finite union of intervals, and hence the winning sets can be approximated by intervals.

The complexity is a major concern for verification and control synthesis algorithms that run on a discrete state space S. The computational time increases as the cardinal number

$|S|$ increases. The bottleneck of the abstraction-based approach for the control synthesis of continuous-state systems is that a uniform discretization of the continuous state space often leads to an exceptionally large finite abstraction in order that the control synthesis is sound and robustly complete for the original system. To improve on this aspect, we apply an *adaptive partitioning scheme* that incorporates interval approximation of predecessors in each iteration, under which the state space are finely discretized only in the region where necessary.

We call the proposed control synthesis method the *specification-guided method via interval computation*. To show the effectiveness and efficiency of the proposed method in this chapter, we analyze its computational time complexity and test the algorithms implemented with intervals on several benchmarking examples.

5.1 Interval Analysis

Interval analysis, or interval computation, refers to the computational methods that use interval arithmetic with the aim to yield rigorous and reliable results. Such methods have been developed since the 1960s [92] and successfully applied in solving different problems [65], including computing reachable sets for continuous-time systems [28] by way of validated numerical solutions to initial value problems for ordinary differential equations [95].

A major advantage of using interval methods for the computation of predecessors is the flexibility to represent any compact set involved in the computation as unions of intervals. Computation of predecessors is essentially a CSP.

Definition 5.1 (CSP)**.** Let function $f : \mathbb{R}^n \to \mathbb{R}^m$. Given a set $B \subseteq \mathbb{R}^m$, find the set of states $A \subseteq \mathbb{R}^n$ such that $f(A) \subseteq B$.

The essence of interval methods lies in its ability to solve CSPs. A branch-and-bound technique is used to solve CSPs [54, 110] and, more recently, to enclose set boundaries [138]. It also applies in computing preimages under nonlinear maps. The corresponding algorithm is known as Set Inversion Via Interval Analysis (SIVIA) [65].

Definition 5.2. An *interval* $[a]$ is a set of real numbers, where

$$[a] \triangleq [\underline{a}, \overline{a}] = \{x : \underline{a} \leq x \leq \overline{a},\ \underline{a}, \overline{a} \in \mathbb{R}\},$$

where $\underline{a}$ and $\overline{a}$ represent the infimum and supremum of $[a]$, respectively. The space that contains any intervals is called the *interval space*, denoted by $\mathbb{IR}$.

By Definition 5.2, it is natural to consider the following qualities of intervals:

- width: $\text{wid}([a]) = \overline{a} - \underline{a}$;
- center: $\text{mid}([a]) = (\underline{a} + \overline{a})/2$;
- magnitude: $|[a]| = \max\{|\underline{a}|, |\overline{a}|\}$.

Similar to real numbers, for any intervals $[a]$ and $[b]$, we can also define the binary arithmetic operations $* \in \{+, -, \times, /\}$ by

$$[a] * [b] \triangleq \{x * y : \ x \in [a], y \in [b]\} .$$

A more specific definition for each operation are given as follows:

$$\begin{aligned}
[a] + [b] &= [\underline{a} + \underline{b}, \overline{a} + \overline{b}];\\
[a] - [b] &= [\underline{a} - \overline{b}; \overline{a} - \underline{b}];\\
[a] \times [b] &= [\min\left\{\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\right\}, \max\left\{\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\right\}];\\
[a]/[b] &= [\underline{a}, \overline{a}][1/\overline{b}, 1/\underline{b}];
\end{aligned}$$

An interval represents a set over reals in a specific form, and hence it inherits the set inclusion relation, which is defined specifically by

$$[a] \subseteq [b] \Leftrightarrow \underline{a} \geq \underline{b}, \text{and } \overline{a} \leq \overline{b}.$$

The interval-arithmetic operations are inclusion monotone, i.e.,

$$[a_1] \subseteq [a_2], \ [b_1] \subseteq [b_2] \Rightarrow [a_1] * [b_1] \subseteq [a_2] * [b_2].$$

Interval vectors and matrices can also be defined by replacing each element with an interval. An interval vector (or box) in $\mathbb{R}^n$ is denoted by

$$[x] \triangleq [x_1] \times \cdots \times [x_n] \subseteq \mathbb{R}^n,$$

where $[x_i] = [\underline{x}_i, \overline{x}_i] \in \mathbb{IR}$ for $i = 1, \cdots, n$.

The width of the interval $[x]$ is defined as $w([x]) \triangleq \max_{1\leq i\leq n}\{\overline{x_i} - \underline{x_i}\}$. Any matrix $[A] \in \mathbb{IR}^{n\times n}$, $[a_{ij}] \in \mathbb{IR}$, $1 \leq i, j \leq n$. The inclusion relation also applies to interval vectors:

$$[x] \subseteq [y] \Leftrightarrow [x_i] \subseteq [y_i], \ 1 \leq i \leq n,$$

The arithmetic operations involving interval vectors and matrices follow the same rules as for real numbers except that elementwise operations are between intervals, e.g.,

$$[c] = [A][b] : \quad [c_i] = \sum_{j=1}^{n} [a_{ij}][b_j].$$

This above operation is important for computation involve linear systems or operations.

To evaluate the system evolution using intervals, we need to define maps between intervals.

Definition 5.3. [65] Consider a function $f : \mathbb{R}^n \to \mathbb{R}^m$ and an interval function $[f] : \mathbb{IR}^n \to \mathbb{IR}^m$. The function $[f]$ is called a *convergent inclusion function* of f if the following two conditions hold:

(i) $f([x]) \subseteq [f]([x])$ for all $[x] \in \mathbb{IR}^n$;

(ii) $\lim_{w([x])\to 0} w([f]([x])) = 0$.

For a vector-valued function f, its convergent inclusion function counterpart is not unique. Methods varies in obtaining such inclusion functions. One can compute the infimum and supremum of $f([x])$ by performing optimizations on the interval $[x]$ if they are trivial. One of the straightforward inclusion function is called natural inclusion function, which is the result by replacing variables by interval variables and each operation by its interval counterpart. For higher precision, centered-form

$$[f]([x]) = f(\text{mid}([x])) + g([x] - \text{mid}([x])),\ g(x) = f(x) - f(\text{mid}([x]))$$

and mean-value form

$$[f]([x]) = f(\text{mid}([x])) + \nabla f([x])([x] - \text{mid}([x]))$$

with more precise expressions can be used according to the approximation accuracy requirements [65].

Example 5.1 (Example of Convergent Inclusion Functions)**.** Evaluate

$$f(x) = x^2 - x$$

on $[x] = [0, 1]$, $\bar{x} = 0.5$ using different convergent inclusion functions:

- $f([x]) = [-0.25, 0]$,
- $[f]_1([x]) = [x]^2 - [x] = [0, 1] \times [0, 1] - [0, 1] = [-1, 1]$,
- $[f]_2([x]) = \bar{x}^2 - \bar{x} + (2[x]-1)([x]-\bar{x}) = -0.25 + (2[0, 1]-1) \times [-0.5, 0.5] = [-0.75, 0.25]$,
- $[f]_3([x]) = f([x])$.

5.2 Bounded Approximation of Predecessors

To implement the map $\widehat{\text{Pre}}$ in control synthesis algorithms (4.8), (4.19), and (4.27), we use interval arithmetic. This is because interval operations are simple, and any compact set can be approximated by intervals with convergence guarantee under mild assumptions.

In order to evaluate the transition relation R in $\mathcal{S}$ over $\mathbb{IR}^n$, we introduce an interval-valued system

$$[\mathcal{S}] : \langle \mathbb{X}, \mathbb{U}, [R], AP, L \rangle, \tag{5.1}$$

where the set of states $\mathbb{X}$, the set of inputs $\mathbb{U}$, the set of atomic propositions AP, and the labeling function L are defined as in (2.4). The inclusion transition relation is defined as $[R]([x], u) \triangleq [f]([x], u)$ for all $[x] \subseteq \mathbb{X}$ and $u \in \mathbb{U}$, and $[f]$ is a convergent inclusion function of f.

Inspired by SIVIA algorithm, now we present Algorithm 5.1, which provides an interval approximation of $\text{Pre}(B|A)$ for any $A, B \subseteq \mathbb{X}$.

Algorithm 5.1 $[\underline{A}, \Delta A, A_c] = \text{PRE}([\mathcal{S}], B, A, \varepsilon)$

```
1:  $\underline{A} \leftarrow \emptyset, \Delta A \leftarrow \emptyset, A_c \leftarrow \emptyset$
2:  $List \leftarrow A$
3:  while $List \neq \emptyset$ do
4:      $[x] \leftarrow List.first$
5:      if $[R]([x], u) \cap B = \emptyset$ for all $u \in U$ then
6:          $A_c \leftarrow A_c \cup [x]$
7:      else if $[R]([x], u) \subseteq Y$ for some $u \in U$ then
8:          $\underline{A} \leftarrow \underline{A} \cup [x]$
9:      else
10:         if wid$([x]) < \varepsilon$ then
11:             $\Delta A \leftarrow \Delta A \cup [x]$
12:         else
13:             $\{Left[x], Right[x]\} = Bisect([x])$        ▷ Perform bisection to $[x]$.
14:             $List.add(\{Left[x], Right[x]\})$
15:         end if
16:     end if
17: end while
```

Algorithm 5.1 takes as input compact sets A, B, which are assumed to be intervals or unions of a finite number of intervals. This is without loss of generality, because any compact set can be arbitrarily approximated by a finite union of intervals because of the Borel-Lebesgue finite covering theorem.

During each iteration, Algorithm 5.1 checks if the image $[f]([x])$ of a particular box $[x]$ is contained in B, the outer approximation obtained in the previous iteration, or completely outside of Y. If neither, and the box size is greater than ε, then $[x]$ is deemed to be undetermined and divided into two subintervals $Left[x]$ and $Right[x]$ by bisection, which are given by

$$\begin{aligned} Left[x] &= [\underline{x}_1, \overline{x}_1] \times \cdots \times [\underline{x}_j, (\underline{x}_j + \overline{x}_j)/2] \times \cdots \times [\underline{x}_n, \overline{x}_n], \\ Right[x] &= [\underline{x}_1, \overline{x}_1] \times \cdots \times [(\underline{x}_j + \overline{x}_j)/2, \overline{x}_j] \times \cdots \times [\underline{x}_n, \overline{x}_n], \end{aligned}$$

where j is the dimension in which the box x attains its width. A box will go through subdivision if necessary until its size is less than the precision parameter ε.

In the outputs of Algorithm 5.1, $\underline{A}$ denotes the set of intervals that absolutely belong to $\mathrm{Pre}(B|A)$, A_c is the set of intervals that does not, and those intervals that partially intersect with $\mathrm{Pre}(B|A)$, i.e., undetermined intervals, are collected in ΔA. The parameter ε controls the minimum width of intervals for approximating $\mathrm{Pre}(B|A)$.

It is easy to see that any intervals in $\underline{A}$ is a subset of $\mathrm{Pre}(B|A)$ and $\mathrm{Pre}(B|A)$ can be covered by the union of intervals contained in $\underline{A}$ and ΔA. Let

$$[\underline{\mathrm{Pre}}]^{\varepsilon}(B|A) \triangleq \bigcup_{[x]\in\underline{A}} [x], \tag{5.2}$$

$$[\overline{\mathrm{Pre}}]^{\varepsilon}(B|A) \triangleq \bigcup_{\substack{[x]\in\underline{A},\text{or}\\ [x]\in\Delta A}} [x], \tag{5.3}$$

where $\underline{A}$ and ΔA are obtained by $\textsc{Pre}([\mathcal{S}], B, A, \varepsilon)$. Then $[\underline{\mathrm{Pre}}]^{\varepsilon}(B|A)$ and $[\overline{\mathrm{Pre}}]^{\varepsilon}(B|A)$ represent an inner and outer approximations of $\mathrm{Pre}(B|A)$, respectively, i.e.,

$$[\underline{\mathrm{Pre}}]^{\varepsilon}(B|A) \subseteq \mathrm{Pre}(B|A), \quad \mathrm{Pre}(B|A) \subseteq [\overline{\mathrm{Pre}}]^{\varepsilon}(B|A).$$

Remark 5.1. More generally, the input set B can also be defined by equations or inequalities, i.e.,

$$B \triangleq \{y \in \mathbb{R}^n : g(y) \le 0\}, \quad g : \mathbb{R}^n \to \mathbb{R}^l.$$

In this case, the condition $[f]([x], u) \cap B = \emptyset$ and $[f]([x], u) \subseteq B$ can be respectively tested by

$$\begin{aligned} &[g \circ f]([x], u) \subseteq [0, \infty]^l, \\ &[g \circ f]([x], u) \subseteq [-\infty, 0]^l, \end{aligned}$$

respectively, where $[g \circ f]([x], u)$ denotes the convergent inclusion function of the composite function $g(f([x], u))$.

By using Algorithm 5.1 to approximate the exact predecessor $\text{Pre}(B|A)$ $(A, B \subseteq \mathbb{X})$, it is often of great interest to know how close the returned approximations are to the real one and in which way the precision parameter ε affects the approximations. To this end, in the following two sections, we evaluate the bounds of approximation errors of the inner approximation $[\underline{\text{Pre}}]^{\varepsilon}(B|A)$ and the outer approximation $[\overline{\text{Pre}}]^{\varepsilon}(B|A)$ to $\text{Pre}(B|A)$ in terms of the precision parameter ε.

5.2.1 Finite Control Values

Let us consider a finite input space U first, i.e., Assumption 3.2 holds. Then system $\mathcal{S}$ can be treated as switched system (3.6), which has been discussed in Chapter 2.

Assumption 5.1 (Lipschitz in $\mathbb{X}$)**.** There exists a constant $\rho > 0$ for the function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ in (2.2) such that for all $u \in \mathbb{U}$

$$\|f(x,u) - f(y,u)\|_{\infty} \leq \rho \|x - y\|_{\infty}, \quad \forall x, y \in A \subseteq \mathbb{R}^n. \tag{5.4}$$

By Assumption 5.1, we can always construct the mean-value form convergent inclusion function for all $[x] \subseteq A \subseteq \mathbb{R}^n$:

$$[f]([x], u) = f(\text{mid}([x]), u) + \rho([x] - \text{mid}([x]))\mathbf{1}^n, \tag{5.5}$$

where $\mathbf{1}^n$ denotes the n-dimensional vector with all its elements 1.

The following lemma gives the error bounds of the inner and outer approximations of $\text{Pre}(B|A)$ under Assumption 5.1.

Lemma 5.1. Consider system $\mathcal{S}$. Let $B, A \subseteq \mathbb{X}$ be compact. If Assumption 5.1 holds in an neighborhood of A, then

$$\text{Pre}(B|A) \subseteq [\overline{\text{Pre}}]^{\varepsilon}(B|A) \subseteq \text{Pre}(B \oplus \mathcal{B}_{\rho\varepsilon}|A), \tag{5.6}$$
$$\text{Pre}(B \ominus \mathcal{B}_{\rho\varepsilon}|A) \subseteq [\underline{\text{Pre}}]^{\varepsilon}(B|A) \subseteq \text{Pre}(B|A). \tag{5.7}$$

Proof. It follows straightforwardly from Algorithm 5.1 that $\text{wid}([x]) < \varepsilon$ for all $[x] \in \Delta A$, and

$$[\underline{\text{Pre}}]^{\varepsilon}(B|A) \subseteq \text{Pre}(B|A) \subseteq [\overline{\text{Pre}}]^{\varepsilon}(B|A) \subseteq A.$$

By (5.4), we have $\text{wid}([f]([x], u)) \leq \rho\text{wid}([x]) < \rho\varepsilon$ for all $[x] \in \Delta A$ and $u \in \mathbb{U}$, where the inclusion function $[f]$ is given in (5.5). Then for any $[x] \in \Delta A$, there exists a $u \in \mathbb{U}$ such that

$[f]([x],u) \cap B \neq \emptyset$ and $[f]([x],u) \subseteq B \oplus \mathcal{B}_{\rho\varepsilon}$ by the definition of the Minkowski sum. Also, $\underline{A} \subseteq \text{Pre}(B|A)$. Hence,

$$\overline{A} = (\underline{A} \cup \Delta A) \subseteq \text{Pre}(B \oplus \mathcal{B}_{\rho\varepsilon}|A) \subseteq \text{Pre}(B \oplus \mathcal{B}_{\rho\varepsilon}|A),$$

which shows (5.6).

We now show that $\text{Pre}(B \ominus \mathcal{B}_{\rho\varepsilon}|A) \subseteq [\underline{\text{Pre}}]^{\varepsilon}(B|A)$. If not, there exists an $x \in \text{Pre}(B \ominus \mathcal{B}_{\rho\varepsilon}|A)$, but $x \notin [\underline{\text{Pre}}]^{\varepsilon}(B|A)$. Then x has to be in $\bigcup_{[x]\in\Delta A}[x]$, since $x \in \bigcup_{[x]\in A_c}[x]$ implies that $x \notin \text{Pre}(B \ominus \mathcal{B}_{\rho\varepsilon}|A)$, which is contradictory to the fact that $x \in \text{Pre}(B \ominus \mathcal{B}_{\rho\varepsilon}|A)$. Let $x \in [x] \in \Delta A$. By Proposition 3.1 (ii), there exists $u \in \mathbb{U}$ such that

$$f(x,u) \in [f]([x],u) \subseteq B \ominus \mathcal{B}_{\rho\varepsilon} \oplus \mathcal{B}_{\rho\varepsilon} \subseteq B.$$

It implies that $[x] \subseteq [\underline{\text{Pre}}]^{\varepsilon}(B|A)$, which is a contradiction. Hence, (5.7) holds. □

5.2.2 Infinite Control Values

Under Assumption 3.1, the compact set $\mathbb{U} \subseteq \mathbb{R}^m$ might contain an infinite number of elements in $\mathbb{U}$. In this case, Algorithm 5.1 becomes impractical because we cannot enumerate all the elements in $\mathbb{U}$.

To inner approximate $\text{Pre}(B|A)$, a straightforward way is to use an under-sampled set of controls, e.g., a set of uniformly sample points within $\mathbb{U}$ defined as

$$[\mathbb{U}]_\mu \triangleq \mu\mathbb{Z}^m \cap \mathbb{U}, \tag{5.8}$$

where $\mu\mathbb{Z}^m \triangleq \{\mu z : z \in \mathbb{Z}^m, \mu > 0\}$.

We define another system by replacing $\mathbb{U}$ in (5.1) by $[\mathbb{U}]_\mu$

$$[\mathcal{S}]_\mu : \langle \mathbb{X}, [\mathbb{U}]_\mu, [R], AP, L \rangle. \tag{5.9}$$

Denote by $[\underline{\text{Pre}_\mu}]^{\varepsilon}(B|A)$ the set of intervals given in (5.2) with $\underline{A}$ returned by $\textsc{Pre}([\mathcal{S}]_\mu, B, A, \varepsilon)$. To achieve a similar result to (5.7) in Lemma 5.1, we additionally require the following assumption.

Assumption 5.2 (Lipschitz in $\mathbb{U}$)**.** Consider system $\mathcal{S}$. There exists a Lipschitz constant $\rho > 0$ for the function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ in (2.2) such that for all $x \in A \subseteq \mathbb{R}^n$

$$\|f(x,u) - f(x,v)\|_\infty \leq \rho \|u - v\|_\infty, \quad \forall u, v \in \mathbb{U}. \tag{5.10}$$

Lemma 5.2. Consider system $[\mathcal{S}]_\mu$ where μ is a parameter given in (5.8). Let $B, A \subseteq \mathbb{X}$ be compact. If Assumption 5.1 and 5.2 hold in a neighborhood of A with Lipschitz constant $\rho_1 > 0$ and $\rho_2 > 0$, respectively, then

$$\text{Pre}(B \ominus \mathcal{B}_{\rho_1\varepsilon+\rho_2\mu}|A) \subseteq [\underline{\text{Pre}_\mu}]^\varepsilon(B|A) \subseteq \text{Pre}(B|A). \tag{5.11}$$

Proof. We define a new predecessor map

$$\text{Pre}_\mu(B) \triangleq \{x \in \mathbb{X} :\ \exists u \in [\mathbb{U}]_\mu,\ \text{s.t. } f(x,u) + d \in B, \forall d \in \mathbb{D}\}.$$

Let $Z = \text{Pre}(B|A)$, $Z_\mu = \text{Pre}_\mu(B|A)$, $\underline{Z} = [\underline{\text{Pre}_\mu}]^\varepsilon(B|A)$, and $\widetilde{B} = B \ominus \mathcal{B}_{\rho_2\frac{\mu}{2}}$.

We first claim that $\text{Pre}(\widetilde{B}|A) \subseteq Z_\mu \subseteq Z$. Trivially $Z_\mu \subseteq Z$ because $[\mathbb{U}]_\mu$ is a subset of $\mathbb{U}$. By Definition 3.3, for all $z \in \text{Pre}(\widetilde{B}|A)$, there exists a $u \in \mathbb{U}$ such that $f(z,u)+d \in \widetilde{B}$ for all $d \in \mathbb{D}$. With Assumption 5.2, for all $u \in \mathbb{U}$, there exists a $v \in [\mathbb{U}]_\mu$ such that $f(z,v) \in f(z,u) \oplus \mathcal{B}_{\rho_2\frac{\mu}{2}}$. Then by Proposition 3.1 (ii),

$$\begin{aligned} f(z,v) + d &\in f(z,u) \oplus \mathcal{B}_{\rho_2\frac{\mu}{2}} + d = (f(z,u)+d) \oplus \mathcal{B}_{\rho_2\frac{\mu}{2}} \\ &\in \widetilde{B} \oplus \mathcal{B}_{\rho_2\frac{\mu}{2}} = B \ominus \mathcal{B}_{\rho_2\frac{\mu}{2}} \oplus \mathcal{B}_{\rho_2\frac{\mu}{2}} \subseteq B, \end{aligned}$$

which means that $z \in Z_\mu$. Hence the claim holds.

By (5.7) in Lemma 5.1, $\text{Pre}_\mu(\widetilde{B} \ominus \mathcal{B}_{\rho_1\varepsilon}|A) \subseteq \underline{Z} \subseteq Z_\mu$. Applying the claim above, we have

$$\text{Pre}(B \ominus \mathcal{B}_{\rho_1\varepsilon+\rho_2\mu}|A) = \text{Pre}(\widetilde{B} \ominus \mathcal{B}_{\rho_1\varepsilon} \ominus \mathcal{B}_{\rho_2\frac{\mu}{2}}|A) \subseteq \text{Pre}_\mu(\widetilde{B} \ominus \mathcal{B}_{\rho_1\varepsilon}|A).$$

Therefore, $\text{Pre}(B \ominus \mathcal{B}_{\rho_1\varepsilon+\rho_2\mu}|A) \subseteq \underline{Z} \subseteq Z_\mu \subseteq \text{Pre}(B|A)$, which is (5.11). □

The outer approximation obtained from $\textsc{Pre}([\mathcal{S}]_\mu, B, A, \varepsilon)$ does not necessarily satisfy a relationship similar to (5.6) in Lemma 5.1, because the set of control values $[\mathbb{U}]_\mu$ in system $[\mathcal{S}]_\mu$ is only a finite subset of $\mathbb{U}$ in $\mathcal{S}$. Any evaluation of $[R]$ is only an inner approximation in terms of control input.

Remark 5.2. In most cases, we use a common Lipschitz constant $\rho = \max\{\rho_1, \rho_2\}$ for the purpose of simplicity. Then (5.11) becomes

$$\text{Pre}(B \ominus \mathcal{B}_{\rho(\varepsilon+\mu)}|A) \subseteq [\underline{\text{Pre}_\mu}]^\varepsilon(B|A) \subseteq \text{Pre}(B|A), \tag{5.12}$$

For control purposes, interval-valued outer approximation of predecessors cannot give information for constructing provably correct control strategies, but they are helpful in showing the convergence results, which will be discussed later.

5.3 Finite Termination and Robust Completeness

As have been discussed in previous chapters, a condition for robustly complete control synthesis with respect to a fundamental LTL specification φ, which indicates that a control strategy can be found by formal algorithms as long as φ is robustly realizable for system $\mathcal{S}$, is that the approximation of the predecessor map is both lower and upper bounded (Assumption 3.3).

Based on Lemmas 5.1 and 5.2, the interval-valued set $[\underline{\text{Pre}}]^{\varepsilon}$ or $[\underline{\text{Pre}_{\mu}}]^{\varepsilon}$ can be used as an inner approximation of Pre for robustly complete control synthesis, provided the precision parameter ε is properly chosen.

Before we advance to the criteria for choosing ε, let us first discuss the finite termination problem that is unsolved in Chapter 4.

5.3.1 Finite Termination

Without any requirement other than Assumption 3.3, the aforementioned algorithms do not necessarily terminate in a finite number of steps, although the approximated winning set is compact. This is because the predecessor is defined over the continuous state space, and the difference between two sets can be infinitesimal.

The inner approximation $[\underline{\text{Pre}}]^{\varepsilon}(B|A)$ of the predecessor $\text{Pre}(B|A)$ using Algorithm 5.1 for any compact set $B, A \subseteq \mathbb{X}$ is a union of intervals with minimum width greater than $\varepsilon/2$, where $\varepsilon > 0$ is the precision parameter of Algorithm 5.1.

We now formalize the finite termination conclusion in the following theorems.

Theorem 5.1 (Finite Termination)**.** Consider system $\mathcal{S}$ with the labeling function (4.2) where Ω is a compact subset of the state space $\mathbb{X}$. Let φ be an LTL formula from one of the classes:

- invariance ($\varphi_{\text{s}} = \Box G$),
- reachability ($\varphi_{\text{r}} = \Diamond G$), and
- reach-and-stay ($\varphi_{\text{rs}} = \Diamond\Box G$).

Let $\widehat{\text{Pre}} = [\underline{\text{Pre}}]^{\varepsilon}$ or $\widehat{\text{Pre}} = [\underline{\text{Pre}_{\mu}}]^{\varepsilon}$ in the corresponding control synthesis algorithm with respect to φ, i.e., (4.8) for φ_{s}, (4.19) for φ_{r}, and (4.27) for φ_{rs}. Then control synthesis with respect to φ terminate in a finite number of steps.

Proof. First of all, let $\varphi = \varphi_{\mathrm{s}}$. Suppose that $\widehat{\mathrm{Inv}}_i(\Omega) \neq \widehat{\mathrm{Inv}}_{i+1}(\Omega)$ for any $i \in \mathbb{N}$. As have shown in the proof of Theorem 4.1, the sequence of sets $\{\widehat{\mathrm{Inv}}_i(\Omega)\}_{i=0}^{\infty}$ is strictly decreasing. Under a given precision ε and the compactness of Ω, $\widehat{\mathrm{Inv}}_i(\Omega)$ contains a finite number of intervals. Then there must exists an $N \in \mathbb{Z}^+$ such that $\widehat{\mathrm{Inv}}_N(\Omega) = \emptyset$, which results in $\widehat{\mathrm{Inv}}_i(\Omega) = \emptyset$ for all $i \geq N$. Hence, algorithm (4.8) terminate in a finite number of iterations.

Next, consider that $\varphi = \varphi_{\mathrm{r}}$. Since we assume that $\mathbb{X}$ is compact, $\mathcal{BR}_{\infty}^{\delta}(\Omega)$ and $\mathcal{BR}_{\infty}(\Omega)$ are all bounded. It follows that $\bigcup_{i=0}^{\infty} \widehat{\mathrm{Rch}}_i(\Omega)$ is also bounded by (4.20). Suppose that $\widehat{\mathrm{Rch}}_i(\Omega) \neq \widehat{\mathrm{Rch}}_{i+1}(\Omega)$ for every $i \in \mathbb{N}$. Then the sequence of sets $\{\widehat{\mathrm{Rch}}_i(\Omega)\}_{i=0}^{\infty}$ is strictly increasing. Given that the minimum width of every interval in $\widehat{\mathrm{Rch}}_i(\Omega)$ for all $i \in \mathbb{N}$ is greater than $\varepsilon/2$, there must exist $N \in \mathbb{N}$ such that $\widehat{\mathrm{Rch}}_N(\Omega) \supseteq \mathbb{X}$, but all sets are bounded in $\mathbb{X}$. Hence, the sequence $\{\widehat{\mathrm{Rch}}_i(\Omega)\}_{i=0}^{\infty}$ will become stationary after a finite number of iterations, and algorithm (4.19) terminates.

Last, let $\varphi = \varphi_{\mathrm{rs}}$. Under a given precision $\varepsilon > 0$, the elements in of sequence $\{W_i\}_{i=0}^{\infty}$ generated in the inner loop in (4.27) must stay unchanged after some positive integer $N \in \mathbb{N}$ by the result for $\varphi = \varphi_{\mathrm{s}}$. Thus, the inner loop terminates within each outer loop. Likewise, the outer loop is also terminating as shown for $\varphi = \varphi_{\mathrm{r}}$. Hence, algorithm (4.27) also terminates in a finite number of steps. □

5.3.2 Robust Completeness Based on Interval Partitions

In Lemmas 5.1 and 5.2, we have established lower and upper bounds for $[\underline{\mathrm{Pre}}]^{\varepsilon}(B|A)$ ($[\underline{\mathrm{Pre}}_{\mu}]^{\varepsilon}$), which is the interval-valued inner approximation of $\mathrm{Pre}(B|A)$. This implies that control synthesis with respect to LTL specifications can be made sound and robustly complete by choosing a proper precision parameter so that Assumption 3.3 is satisfied.

In this section, we study the condition of being robustly complete for the control synthesis algorithms that use $[\underline{\mathrm{Pre}}]^{\varepsilon}$ ($[\underline{\mathrm{Pre}}_{\mu}]^{\varepsilon}$) as an inner approximation of Pre as well as partition-based control strategies that realize the corresponding specifications.

As a result of using Algorithm 5.1, the state space $\mathbb{X}$ of the system will be partitioned into a finite number of intervals, which can be treated as a partition $\mathcal{P}$ of $\mathbb{X}$ by Definition 2.4 and each interval is a cell in $\mathcal{P}$. So is the inner approximation of the winning set with respect to a given specification. The extracted control strategies are defined on intervals, since computation of every approximated predecessor is performed over intervals instead of single points in the continuous state space. In other words, we can use the same set of valid control values at any state inside the same interval (or cell) so that the given specification can be realized.

We first investigate in the following theorem the existence of partition-based control strategies provided that the given specification is robustly realizable for system $\mathcal{S}$. Concrete control strategies are given in the constructive proof of the theorem.

Lemma 5.3. Consider system $\mathcal{S}^0$ with the labeling function (4.2) where Ω is a compact subset of the state space $\mathbb{X}$. Let φ be one of the classes considered in Theorem 5.1. Additionally assume that ε can be chosen so that $[\underline{\mathrm{Pre}}]^{\varepsilon}$ given in (5.2) satisfies Assumption 3.3 for some $\delta > 0$. If φ is δ-robustly realizable for $\mathcal{S}^0$, then there exists a partition $\mathcal{P} = \{P_1, P_2, \cdots, P_N\}$ of $\mathbb{X}$ and a memoryless control strategy $\kappa : \mathbb{X} \to 2^{\mathbb{U}}$ with

$$\kappa(x) = \bigcup_{i=1}^{N} \psi_{P_i}(x), \quad x \in \mathbb{X}, \tag{5.13}$$

that realizes φ for system $\mathcal{S}^0$ at any state in its domain.

The map ψ_{P_i} in (5.13) is given by

$$\psi_{P_i}(x) = \begin{cases} \emptyset & x \notin P_i, \\ \pi_i & x \in P_i, \end{cases} \tag{5.14}$$

where $\pi_i \subseteq \mathbb{U}$ for $i \in \{1, \cdots, N\}$.

Proof. Since $[\underline{\mathrm{Pre}}]^{\varepsilon}$ satisfies Assumption 3.3 for some ε, algorithms (4.8), (4.19), and (4.27) with $\widehat{\mathrm{Pre}} = [\underline{\mathrm{Pre}}]^{\varepsilon}$ all return a nonempty subset of the corresponding winning set by Theorems 4.1, 4.2, and 4.3.

Let Y_i ($i \in \mathbb{N}$) denote the set obtained by the ith iteration of (4.8), (4.19), or (4.27). By the finite termination property given in Theorem 5.1, there exists a positive integer J such that $Y_J = Y_{J+1}$. By using Algorithm 5.1 for the approximation of predecessors, the state space $\mathbb{X}$ is represented by a partition $\mathcal{P} = \{P_1, P_2, \cdots, P_N\}$, where $N \in \mathbb{Z}^+$ is the number of intervals, i.e., $\mathbb{X} = \bigcup_{i=1}^{N} P_i$. The inner approximation Y_J of the exact winning set can be characterized by

$$Y_J = \bigcup_{[x] \in \mathcal{P}'} [x], \quad \mathcal{P}' = \{P_{i_1}, \cdots, P_{i_{N'}}\} \subseteq \mathcal{P},$$

where N' denotes the number of intervals in $\mathcal{P}'$.

Recall that every member $[x]$ in the output set $\underline{A}$ of $\textsc{Pre}([\mathcal{S}], B, A, \varepsilon)$ in Algorithm 5.1 can be controlled to set $B \subseteq \mathbb{X}$ under some control inputs. We write $\underline{A} = \{[x]_1, \cdots, [x]_{|\underline{A}|}\}$. Define

$$[\Pi]_B([x]) \triangleq \{u \in \mathbb{U} : [f]([x], u) \subseteq B\}. \tag{5.15}$$

Then the valid control values for any state inside the same interval is the same.

For invariance formula φ_s, we have $Y_J = \widehat{\text{Inv}}_J(\Omega) = [\underline{\text{Pre}}]^{\varepsilon}(Y_J|Y_J) \subseteq \text{Pre}(Y_J|Y_J)$. Define the subset of control values π_i given in (5.14) by

$$\pi_i = \begin{cases} [\Pi]_{Y_J}(P_i), & P_i \in \mathcal{P}', \\ \emptyset & P_i \notin \mathcal{P}'. \end{cases} \tag{5.16}$$

Then $[\Pi]_{Y_J}(x) = [\Pi]_{Y_J}(P_i) \neq \emptyset$ for all $x \in P_i \in \mathcal{P}'$ and

$$\kappa(x) = \begin{cases} [\Pi]_{Y_J}(x) & x \in Y_J, \\ \emptyset & x \in \mathbb{X} \setminus Y_J, \end{cases}$$

which is consistent with the invariance control strategy (4.11). Any state $x \in Y_J$ can be controlled inside $Y_J \subseteq \Omega$ for all future time using the control strategy in the form of (5.13).

For reachability formula φ_r, we use $\widehat{\text{Rch}}(Y) = [\underline{\text{Pre}}]^{\varepsilon}(Y|\mathbb{X}) \cup Y$. The resulting sequence of sets $\{\widehat{\text{Rch}}_j(\Omega)\}$ is increasing and $Y_J = \widehat{\text{Rch}}_J(\Omega)$. Let $\mathcal{P}_j = \{P_{j\oplus 1}, \cdots, P_{j\oplus N_j}\}$ $(j \in \{1, \cdots, J\})$ be the set of intervals inside $\widehat{\text{Rch}}_j(\Omega)$ but outside $\widehat{\text{Rch}}_{j-1}(\Omega)$, where N_j is the number of intervals in $\mathcal{P}_j$ and $j \oplus i = \sum_{l=1}^{j} N_l + i$ for $i \in \{1, \cdots, N_j\}$ is the index used for sorting intervals in $\mathcal{P}'$. Then $\sum_{j=1}^{J} N_j = N'$. Replace π_i in (5.14) by

$$\pi_{j\oplus i} = \begin{cases} [\Pi]_{\widehat{\text{Rch}}_{j-1}(\Omega)}(P_{j\oplus i}) & P_{j\oplus i} \in \mathcal{P}_j \\ \emptyset & \text{o.w.} \end{cases} \tag{5.17}$$

Then the control strategy in (5.13) becomes

$$\kappa(x) = \begin{cases} [\Pi]_{\widehat{\text{Rch}}_{j-1}(\Omega)}(x) & x \in P_{j\oplus i}, \\ \emptyset & x \in \mathbb{X} \setminus Y_J. \end{cases}$$

By algorithm (4.19) and definition (5.15) for valid control values based on Algorithm 5.1, any state $x \in P_{j\oplus i}$ will be controlled to $\widehat{\text{Rch}}_{j-1}(\Omega)$ for one step. This is consistent with (4.21).

For reach-and-stay formula φ_{rs}, control strategies are defined separately on $W_j^{\infty} \subseteq \Omega$ and $Z_j \subseteq \mathbb{X} \setminus \Omega$ for $j \in \{0, \cdots, J-1\}$. By (4.27), we have $W_j^{\infty} \cap W_{j'}^{\infty} = \emptyset$ and $Z_j \cap Z_{j'} = \emptyset$ for $j \neq j'$. Let $\mathcal{P}_{w,j}$ and $\mathcal{P}_{z,j}$ be the set of intervals returned by Algorithm 5.1 as the approximation of W_j^{∞} and Z_j, respectively. We concatenate the intervals in $\mathcal{P}_{w,j}$ and $\mathcal{P}_{z,j}$ in a way that the intervals belong to $\mathcal{P}_{w,j}$ always goes before the ones in $\mathcal{P}_{z,j}$, and denote by N_j and N_{J+j} the number of intervals in $\mathcal{P}_{w,j}$ and $\mathcal{P}_{z,j}$, respectively. Same as for the reachability formula φ_{r}, we assign indices of intervals so that

$$\mathcal{P}_{w,j} = \left\{P_{j\oplus 1}, \cdots, P_{j\oplus N_j}\right\}, \quad \mathcal{P}_{z,j} = \left\{P_{(J+j)\oplus 1}, \cdots, P_{(J+j)\oplus N_{J+j}}\right\},$$

and $\mathcal{P} = \{\mathcal{P}_{w,0}, \cdots, \mathcal{P}_{w,J-1}, \mathcal{P}_{z,0}, \cdots, \mathcal{P}_{z,J-1}\}$.

Let $j \in \{0, \cdots, 2J-1\}$ and

$$\pi_{j\oplus i} = \begin{cases} [\Pi]_{\widehat{X}^{\infty}_{j+1}}(P_{j\oplus i}) & P_{j\oplus i} \in \mathcal{P}, \\ \emptyset & \text{o.w. .} \end{cases} \tag{5.18}$$

Then the control strategy (5.13) realizes φ_{rs}. □

Based on Lemma 5.3, we present the following robust completeness results for control synthesis using interval computation.

Theorem 5.2 (Robust Completeness via Interval Analysis)**.** Consider system $\mathcal{S}^0$ with the labeling function (4.2) where Ω is a compact subset of the state space $\mathbb{X}$. The set of control inputs $\mathbb{U}$ is finite. Suppose that Assumption 5.1 holds with the Lipschitz constant $\rho_1 > 0$. Let φ be a specification considered in Theorem 5.1 and $[Y]^\varepsilon$ denote the approximated winning set with respect to φ by letting $\widehat{\mathrm{Pre}} = [\underline{\mathrm{Pre}}]^\varepsilon$. If

$$\rho_1 \varepsilon \leq \delta \tag{5.19}$$

for some $\delta > 0$, then the following conclusions hold:

(i) If $[Y]^\varepsilon \neq \emptyset$, then a control strategy in the form of (5.13) can be constructed to realize φ at any state $x \in [Y]^\varepsilon$, which is a finite union of intervals.

(ii) If $[Y]^\varepsilon = \emptyset$, then φ is not δ-robustly realizable for system $\mathcal{S}^0$.

Proof. If $\rho_1\varepsilon \leq \delta$, then by Lemma 5.1, for all $B, A \subseteq \mathbb{X}$

$$\mathrm{Pre}(B \ominus \mathcal{B}_\delta | A) \subseteq \mathrm{Pre}(B \ominus \mathcal{B}_{\rho_1\varepsilon} | A) \subseteq [\underline{\mathrm{Pre}}]^\varepsilon(B|A) \subseteq \mathrm{Pre}(B|A),$$

which means that $[\underline{\mathrm{Pre}}]^\varepsilon$ satisfies Assumption 3.3. If $[Y]^\varepsilon \neq \emptyset$, then we have

$$\mathrm{Win}^\delta_{\mathcal{S}}(\varphi) \subseteq [Y]^\varepsilon \subseteq \mathrm{Win}_{\mathcal{S}}(\varphi) \tag{5.20}$$

by Theorems 4.1, 4.2, and 4.3. By Lemma 5.3, there exists a control strategy in the form of (5.13) to realize φ.

If $[Y]^\varepsilon = \emptyset$, then $\mathrm{Win}^\delta_{\mathcal{S}}(\varphi) = \emptyset$ by (5.20), which implies that φ is not δ-robustly realizable for system $\mathcal{S}^0$. □

For system with infinitely many available control inputs, a similar result can be established as follows.

Theorem 5.3. Consider system $\mathcal{S}^0$ with the labeling function (4.2) where Ω is a compact subset of the state space $\mathbb{X}$. The set of control inputs $\mathbb{U}$ is compact. Suppose that Assumption 5.1 and 5.2 are satisfied with the Lipschitz constant $\rho_1 > 0$ and $\rho_2 > 0$, respectively. Let φ be a specification considered in Theorem 5.1 and $[Y]^\varepsilon$ denote the approximated winning set with respect to φ by letting $\widehat{\mathrm{Pre}} = [\underline{\mathrm{Pre}}_\mu]^\varepsilon$, where μ is the granularity of the input space $\mathbb{U}$ defined in (5.8). Then we have the same results as in Theorem 5.2 if

$$\rho_1\varepsilon + \rho_2\mu \leq \delta \tag{5.21}$$

for some $\delta > 0$.

Proof. The proof is the same as the one for Theorem 5.2 except that (5.20) is achieved by Lemma 5.2 instead of Lemma 5.1. □

Theorems 5.2 and 5.3 essentially reveal that using Algorithm 5.1, which is based on interval arithmetic and a branch-and-bound scheme, as an implementation of $\widehat{\mathrm{Pre}}$ solves Problem 2.2 at least for fundamental LTL control synthesis problems. The conditions (5.19) and (5.21) serve as criteria for choosing the precision parameter ε if the bound of disturbance δ and the Lipschitz constant ρ_1 and ρ_2 over the state space can be trivially determined. Using such a criterion in actual computation is usually too conservative due to the evaluation of the Lipschitz constants over the entire state space.

A practical benefit of Theorems 5.2 and 5.3 is the guarantee that the winning set can be approximated more precisely by using a sufficiently precision parameter. On the other hand, if we start computation with a large ε and iteratively reducing it until the algorithm achieves a nonempty result, algorithms (4.8), (4.19), and (4.27) can also estimate the bound of the disturbances that can be tolerated without breaking the realizability of the given specification.

To show how well the proposed specification-guided method performs in terms of computational time, in the following sections of this chapter, we compare it with abstraction-based methods, which are commonly used for control synthesis with respect to LTL specifications.

5.4 Complexity Analysis

As we have seen in Chapter 1, by using abstraction-based methods, a finite abstraction (transition system) that bisimulates or over approximates the original dynamical system on a continuous state space is first constructed, then computer algorithms that have been developed for discrete state systems are adopted for control synthesis. Therefore, the overall computational time for an abstraction-based method includes the time for both abstraction construction and discrete control synthesis.

Let ε and η ($\varepsilon, \eta > 0$) be the grid sizes of the state space $\mathbb{X}$ and the input space $\mathbb{U}$, respectively. Assume that c_1, $c_2 > 0$ are some constants related to the width of the state and input spaces, respectively, and the cost in terms of running time for each computation of the reachable set of a cell in the discretized state space is some constant $c > 0$. The integers n and m represent the dimension of the state and input spaces of system $\mathcal{S}$, respectively.

To deal with general nonlinear dynamics without any stability assumptions, a finite abstraction that over approximates the behaviors of the original system is often constructed over a uniformly discretized state space. For such cases, the number of discrete states N_S and inputs N_U are

$$N_S = \left\lceil \frac{c_1}{\varepsilon} \right\rceil^n, \quad N_U = \left\lceil \frac{c_2}{\eta} \right\rceil^m,$$

where $\lceil \cdot \rceil$ is the ceiling function. Hence, the time complexity for computing abstractions is $\mathcal{O}(cN_SN_U)$.

Since finite abstractions can be viewed as finite graphs or a two-player game arena [10], the time complexity of discrete control synthesis for abstraction-based methods can be easily concluded by using the related results from the area of two-player games and model checking. Let N_T be the number of transitions in the finite abstraction, which is also the number of edges of the graph that is equivalent to the abstraction. Under Assumption 5.1, the number of transitions N_T can be estimated by

$$N_T = (\lceil \rho \rceil + 1)^n N_S N_U,$$

where ρ is the Lipschitz constant.

As reported in the literature, the time complexity for achieving reachability objective $\varphi_{\mathrm{r}} = \Diamond G$ is $\mathcal{O}(N_T)$. Since invariance objective $\varphi_{\mathrm{s}} = \Box G$ is a dual to φ_{r}, the time complexity is also $\mathcal{O}(N_T)$. For reach-and-stay control objective $\varphi_{\mathrm{rs}} = \Diamond\Box G$, the time complexity is $\mathcal{O}(N_SN_T)$. Hence, the overall time complexity of abstraction-based control synthesis with respect to invariance or reachability is

$$\mathcal{O}(cN_SN_U + (\lceil \rho \rceil + 1)^n N_S N_U), \tag{5.22}$$

and the complexity with respect to reach-and-stay specifications is

$$\mathcal{O}(cN_SN_U + (\lceil \rho \rceil + 1)^n N_S^2 N_U). \tag{5.23}$$

We now analyze the time complexity of the proposed control synthesis algorithms (4.8), (4.19), and (4.27) with interval approximation of the map Pre, which is given in Algorithm 5.1. Using the branch-and-bound technique in Algorithm 5.1, the state space is adaptively partitioned with respect to system dynamics and the satisfaction of the specification. As a result, a non-uniform partition of the state space will be generated, which can be implemented by using the binary tree data structure[1].

If the minimum width of an interval is still ε, then we need to set the precision parameter 2ε in Algorithm 5.1. This is due to the subdivision scheme that any interval with width larger than the precision parameter will be bisected if it can not be fully controlled inside a given set in the next time step. Then the greatest depth of the binary tree is

$$h_{\max} = \left\lceil n \log\left(\frac{c_1}{\varepsilon}\right)\right\rceil \approx \log N_S[2].$$

The operations $[f]([x], u) \cap B = \emptyset$ and $[f]([x], u) \subseteq Y$ in Algorithm 5.1 involves the computation of interval inclusion function $[f]$ (i.e., the computation of reachable set for interval $[x]$) and set membership test, which is performed by searching the binary tree. Assume that the set membership test for intervals costs one operational time compared to c. Then in the worst case where the state space of the tree is of depth $h_{\max}$, determining whether an interval $[x] \subseteq A$ belongs to the predecessor $\text{Pre}(B|A)$ takes approximately $(\log N_S + c)N_U$ operational time.

Let N_G be the number of the set of intervals that represents the target set Ω. Then the number of intervals outside of Ω is $N_S - N_G$. Normally for a regulation problem, the target area Ω is rather small compared to the state space $\mathbb{X}$, which means that $N_G << N_S$.

For invariance control algorithm (4.8), computation is confined to the target set. Assuming that $\widehat{\text{Inv}}_j(\Omega)$ and $\widehat{\text{Inv}}_{j-1}(\Omega)$ $(j \in \mathbb{Z}^+)$ only differ by one interval and $N_G >> 1$, the overall computational complexity is

$$\mathcal{O}\left(\frac{c}{2}N_U N_G^2 + \frac{1}{2}N_U N_G^2 \log N_S\right). \tag{5.24}$$

If the target set is small enough so that $N_G^2 \log N_S < N_S$, the worst case complexity of the invariance control algorithm with interval implementation is lower than the one for abstraction-based methods (5.22).

[1]A tree is an abstract data structure that has a root node which is linked by children nodes. A binary tree is a tree data structure in which each node has at most two children.

[2]The logarithm is with base 2.

For reachability algorithm (4.19), we consider the worst case in which only one more interval is included in $\widehat{\mathrm{Rch}}_j$ than $\widehat{\mathrm{Rch}}_{j-1}$ and $N_G, 1 << N_S$. Then we have the computational complexity

$$\mathcal{O}\left(\frac{c}{2}N_U N_S^2 + \frac{1}{2}N_U N_S^2 \log N_S\right), \tag{5.25}$$

which is higher than (5.22).

Similarly, in the worst case for algorithm (4.27), the set elements in the sequences $\{Y_i\}_{i=0}^{\infty}$ and $\{X_i^j\}_{j=0}^{\infty}$ differ by one interval. Then the number of iterations N_I is

$$N_I = \frac{1}{2}\sum_{i=1}^{N_G}(i^2+i) + \sum_{i=1}^{N_S-N_G} i = \frac{N_G^3 + 3N_G^2 + 8N_G}{12} + \frac{(N_S-N_G)^2 + (N_S-N_G)}{2}.$$

If $N_G << N_S$, then $N_I \approx (N_S^2 + N_S)/2$. Hence, the time complexity of the algorithm (4.27) is

$$\mathcal{O}\left(\frac{\log N_S + c}{2}(N_S N_U + N_S^2 N_U)\right). \tag{5.26}$$

By comparing (5.26) with (5.23) , the time complexity of algorithm (4.27) is of $\mathcal{O}(N_U N_S^2 \log N_S)$ while the abstraction-based methods is quadratic in N_S. The term $\log N_S$ in (5.26) is contributed by the overhead of using Algorithm 5.1, which primarily comes from the set inclusion tests by searching the binary tree, i.e., the part induced by $h_{\max}$.

For both reachability and reach-and-stay control objectives, the abstraction-based methods have better efficiency than the proposed specification-guided method in the worst case, in which the state space $\mathbb{X}$ is assume to be partitioned to intervals of the smallest size (determined by the precision parameter) and the sets, or unions of intervals to be more precise, computed in each iteration varies by only one interval. In such worst cases, the overhead run time of Algorithm 5.1 is relatively large.

The worst case, however, rarely exists in practical control problems. On the other hand, the use of a non-uniform partitioning scheme avoids partitioning the region in the state space without helping in control synthesis. This usually leads to fewer discrete states for a given precision. In this sense, the proposed specification-guided method is less sensitive to the state discretization precisions than abstraction-based control synthesis methods. We will show by some practical control examples in the following section that the experimental run time is far better than ones predicted by (5.24), (5.25) and (5.26).

From the relationship between system dimension and the time complexity as discussed above, the main limitation of the proposed method, which also exists in abstraction-based methods, is that it still suffers from the *curse of dimensionality*.

5.5 Experiments on Benchmarking Examples

In this section, we illustrate the effectiveness of the interval-based control synthesis algorithms on several benchmarking examples that have been used in the literature.

We also compare the run time of solving invariance, reachability and reach-and-stay control synthesis scenarios for different systems by using the proposed specification-guided method and abstraction-based methods. Although theoretical complexity analysis shows that the proposed method takes more time in the worst case because of the overhead for managing non-uniform partitions, the proposed method outperforms abstraction-based methods in those empirical experiments.

Control synthesis in all the examples are performed on a 3.6 GHz processor (Intel Core i3) using ROCS [83], which is a self-developed toolbox for nonlinear system LTL control synthesis. We include in this thesis a detailed presentation of ROCS in Appendix A.

5.5.1 Boost DC-DC Converter

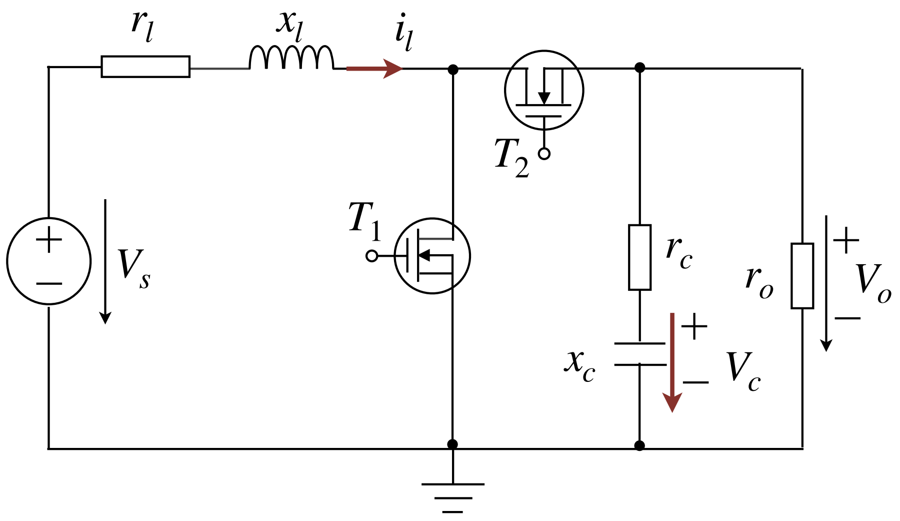


Figure 5.1: The circuit of a boost DC-DC converter.

Consider a boost DC-DC converter [53], which operates in two modes, as shown in Figure 5.1. Let the state x of the converter be a vector of the inductor current i_l and the capacitor voltage V_c. Then the state space model of the boost DC-DC converter is linear affine:

$$\begin{aligned}
&\dot{x} = A_p x + b, \quad p = 1, 2, \\
&x = \begin{bmatrix} i_l \\ V_c \end{bmatrix}, \quad b = \begin{bmatrix} \frac{V_s}{V_l} \\ 0 \end{bmatrix}, \\
&A_1 = \begin{bmatrix} -\frac{r_l}{x_l} & 0 \\ 0 & -\frac{1}{x_c(r_c+r_o)} \end{bmatrix}, \quad A_2 = \begin{bmatrix} -\frac{1}{x_l}(r_l + \frac{r_o r_c}{r_o+r_c}) & -\frac{r_o}{x_l(r_o+r_c)} \\ \frac{r_o}{x_c(r_o+r_c)} & -\frac{1}{x_c(r_o+r_c)} \end{bmatrix}.
\end{aligned} \tag{5.27}$$

The parameters in (5.27) is provided in Table 5.1.

Table 5.1: The parameters in (5.27), and "p.u."= *per unit.*

Parameters	Value (p.u.)	Physical meaning
x_c	70	The capacity of the capacitor
r_c	0.005	The resistance of the capacitor
x_l	3	The inductance of the inductor
r_l	0.05	The resistance of the inductor
r_o	1	The load resistance
V_s	1	The source voltage

Although the physical model is ODEs, the following exact discrete-time model of (5.27) can be derived via integrating 5.27 according to $x_{t+1} = e^{A_p \tau_s} x_t + \int_0^{\tau_s} e^{\tau_s - s} b \, \mathrm{d}s$ by a sampling time $\tau_s > 0$.

$$\begin{aligned}
&\text{Mode 1:} \quad x_{t+1} = \begin{bmatrix} 0.9917 & 0 \\ 0 & 0.9929 \end{bmatrix} x_t + \begin{bmatrix} 0.1660 \\ 0 \end{bmatrix}, \\
&\text{Mode 2:} \quad x_{t+1} = \begin{bmatrix} 0.9903 & -0.1645 \\ 0.0070 & 0.9923 \end{bmatrix} x_t + \begin{bmatrix} 0.1659 \\ 0.0006 \end{bmatrix}.
\end{aligned} \tag{5.28}$$

A typical function of a boost DC-DC converter is to regulate the output voltage V_o within a certain range. Depending on whether the initial state x_0 of the system falls inside this range or not, such a control objective can be described as an invariance specification $\varphi_s = \Box G$ or a reach-and-stay specification $\varphi_{rs} = \Diamond\Box G$ for system $\mathcal{S}$ with transition relation determined by (5.27) and labeling function (4.2).

Hence, we consider two scenarios for both of the specifications in our simulation. The state space for this example is $\mathbb{X} = [0.6490, 1.6500] \times [0.9898, 1.1900]$. In the first scenario, we aim to maintain system state inside a target region $\Omega_1 = [1.15, 1.55] \times [1.09, 1.17]$ (labeled as G), and the second case is to control a state in the state space $\mathbb{X}$ to reach an target set $\Omega_2 = [1.10, 1.6] \times [1.08, 1.18]$ (labeled as G) and stay there for all future time.

A sampling time $\tau_s = 0.5$ s is used for constructing the discrete-time model (5.28). The precision parameter $\varepsilon = 0.001$ is used for the interval version $[\underline{\mathrm{Pre}}]^{\varepsilon}$ of $\widehat{\mathrm{Pre}}$ in both invariance control synthesis algorithm (4.8) and reach-and-stay algorithm (4.27).

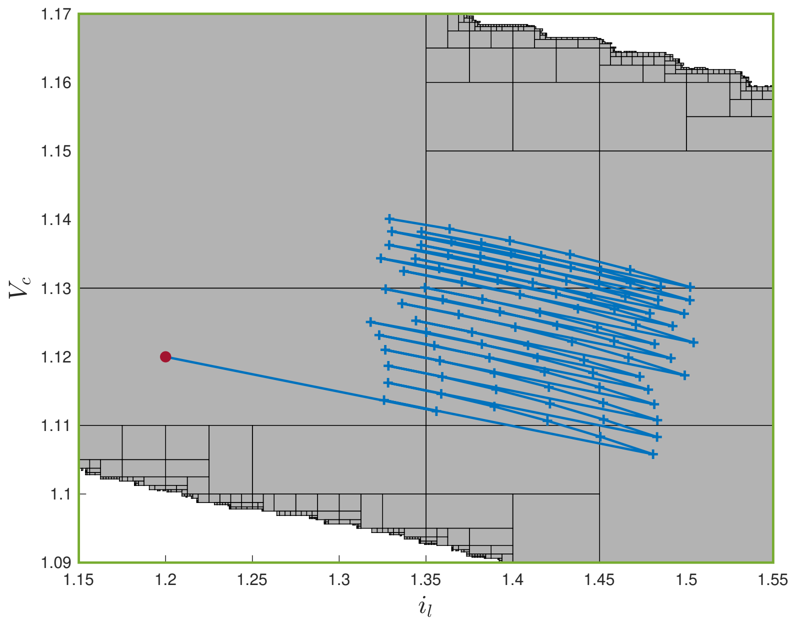


Figure 5.2: The phase portrait of a closed-loop trajectory that satisfies the invariance specification φ_{s}. The area marked by the outermost green rectangle is the target set Ω_1.

In the first case, we approximate the maximal controlled invariant set inside Ω_1, which is the winning set $\mathrm{Win}_{\mathcal{S}}(\varphi_{\mathrm{s}})$ for system $\mathcal{S}$ with respect to the invariance specification φ_{s}, by the union of intervals marked as the shaded area in Figure 5.2. The Lipschitz constant $\rho_1 = \max\{0.9929, 1.0737\} = 1.0737$. Implied by Theorem 5.2, the target set Ω_1 is not δ-robustly controlled invariant for any $\delta > \rho_1\varepsilon = 0.0010737$, because the approximated winning set does not cover Ω_1.

Applying the constructed partition-based memoryless control strategy in the form of (5.13) and (5.14), a closed-loop system trajectory from the initial state $x_0 = (1.2, 1.12)$ is shown in Figure 5.3. Such a control strategy returns (possibly) multiple valid control values for a state in the winning set and any one of the control values realizes the invariance specification. In this

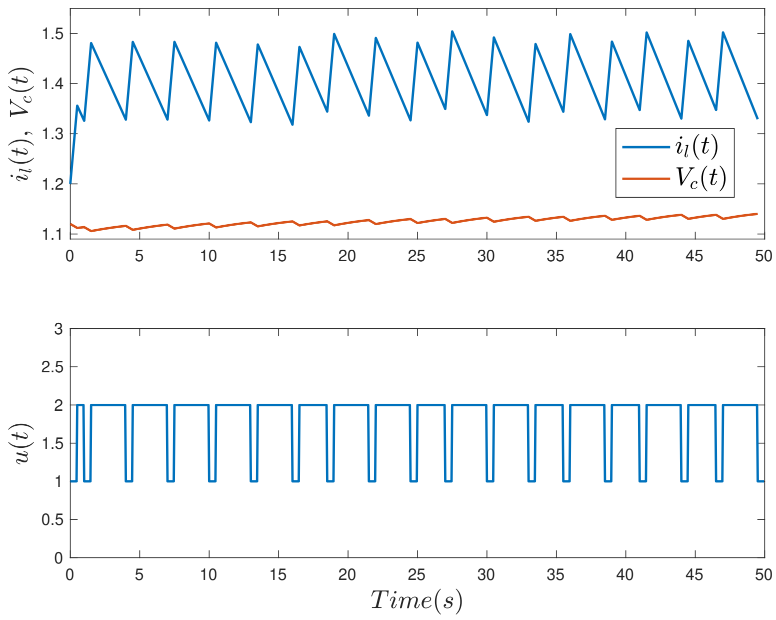


Figure 5.3: The corresponding time history of closed-loop states and control variables in Figure 5.2.

example, the valid control values can be both mode 1 and 2. The only one of the control values that we select in our closed-loop control simulation is the one closest to the last used control value, which results in less mode switching.

It can be seen that the whole trajectory is confined to the controlled invariant set inside Ω_1 as required. Figure 5.3 displays the time history of system states and control inputs. Since we perform control synthesis automatically by formal algorithms on discretized state and input spaces, the curves in Figure 5.3 show discontinuity.

In the second case, we run the reach-and-stay control synthesis algorithm (4.27), and the winning set $\text{Win}_{\mathcal{S}}(\varphi_{\text{rs}})$ is approximated by the shaded area in Figure 5.4, which also shows a closed-loop trajectory from with initial condition $x_0 = (0.7, 1.08)$. Similar to the first case, the target set is not δ-robustly controlled invariant itself as $\Omega_2 \nsubseteq \text{Win}_{\mathcal{S}}(\varphi_{\text{rs}})$.

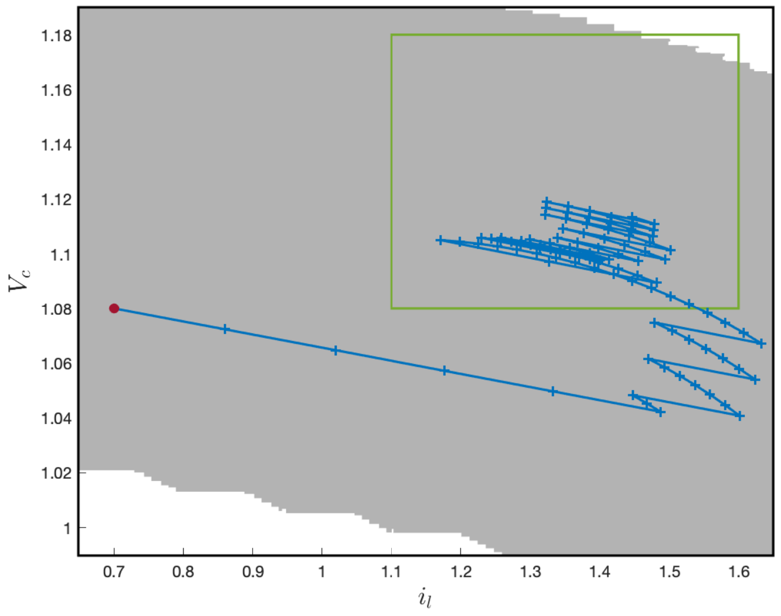


Figure 5.4: A closed-loop trajectory of the converter using the control strategy that realizes the reach-and-stay specification φ_{rs}. The target set Ω_2 is marked as the green rectangle.

5.5.2 Parallel Parking

We now consider an automatic parallel parking problem in which the goal is to control a vehicle to park along the curb between two other vehicles. Such an objective can be expressed by a reach-and-stay specification. The following vehicle model [5] is used:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\gamma+\theta)\cos(\gamma)^{-1} \\ v\sin(\gamma+\theta)\cos(\gamma)^{-1} \\ v\tan(\phi) \end{bmatrix}, \tag{5.29}$$

where (x, y) is the planar position of center of the vehicle, θ is its orientation, the control variable v represents the velocity, and ϕ is the steering angle command.

The vehicle structure is shown in Figure 5.5, and the variable $\gamma = \arctan(a\tan(\phi)/b)$, where a is the distance from the gravity center to the rear wheels of the vehicle, and b is the distance between its front and rear wheels. We use $a/b = 1/2$ in the simulation.

Considering constant control inputs during each sampling period, we can obtain the exact

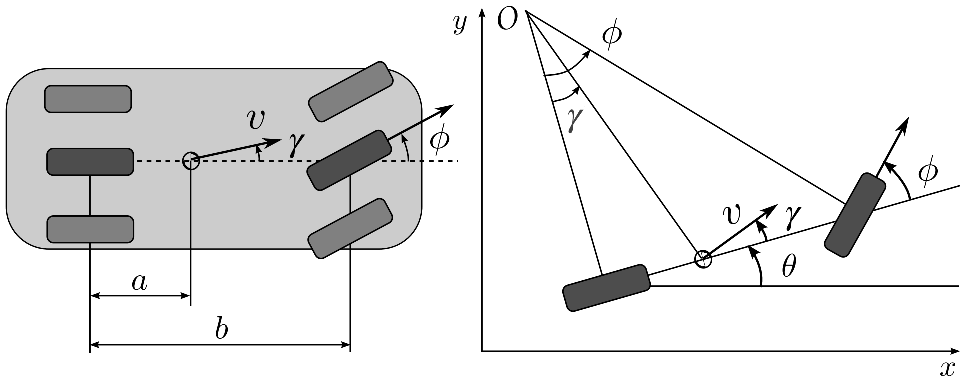


Figure 5.5: The vehicle structure [5].

discrete-time model for $\phi \neq 0$:

$$\begin{cases} x_{t+1} = \frac{\sin(\gamma_t + \tau_s v_t \tan\phi_t + \theta_t) - \sin(\gamma_t + \theta_t)}{\cos(\gamma_t)\tan\phi_t} + x_t, \\ y_{t+1} = \frac{-\cos(\gamma_t + \tau_s v_t \tan\phi_t + \theta_t) + \cos(\gamma_t + \theta_t)}{\cos(\gamma_t)\tan\phi_t} + y_t, \\ \theta_{t+1} = \tau_s v_t \tan\phi_t + \theta_t. \end{cases} \tag{5.30}$$

For $\phi = 0$, the discrete-time model becomes

$$\begin{cases} x_{t+1} = v_t \cos\theta_t \tau_s + x_t, \\ y_{t+1} = v_t \sin\theta_t \tau_s + y_t, \\ \theta_{t+1} = \theta_t. \end{cases} \tag{5.31}$$

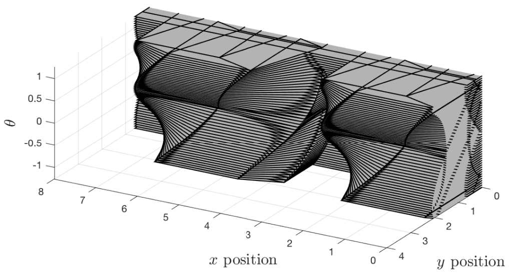
In our simulation, the state space is $\mathbb{X} = [0, 8] \times [0, 4] \times [-72°, 72°]$, sampling time is $\tau_s = 0.3$s, and the set of control values is $\mathbb{U} = \{\pm 0.9, \pm 0.6, \pm 0.3, 0\}$, which is sampled by uniform discretization of the space $[-1, 1] \times [-1, 1]$ with grid width $\eta = 0.3$.

The discrete-time model can be readily verified Lipschitz continuous over the state space $\mathbb{X}$ and the input space $\mathbb{U}$: For (5.31) when the steering angle $\phi = 0$, $\rho_1 = 1.3$ and $\rho_2 = 0.3$ are the Lipschitz constants with respect to state and input, respectively. Letting $\phi \to 0$, (5.30) and (5.31) will be almost equivalent.

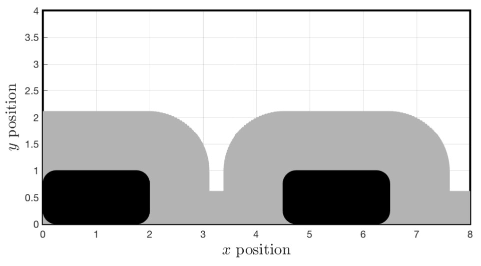
Suppose that the length and width of the vehicle be $L = 2$ and $H = 1$, respectively. For the purpose of analysis, we consider two problem settings: parking with a wide marginal space $\Delta = L = 2$ and a narrow marginal space $\Delta = 0.5$. The marginal space is the distance between the front and rear vehicles in addition to L. For both cases, the rear vehicle center is at $(1, 0.5)$, and thus the front vehicle center is at $(1 + 3L/2 + \Delta, 0.5)$. The target area is

$$\Omega = [1 + L, 1 + L + \Delta] \times [0.5, 0.6] \times [-3°, 3°].$$

The collision area (the center position and orientation of the vehicle that causes collision with the parked vehicles and the curb) needs to be determined before control synthesis. We assume that vehicles and the curb are rectangles. Then the collision area can be interpreted by inequalities of the form $g(x) \leq 0$, which is derived by checking if two polyhedra intersect. It is clear that the center of the vehicle has different admissible regions with different orientations. Hence, the collision area is not simply a hyper-rectangle in $\mathbb{R}^3$, as shown in Figure 5.6 (a). The free workspace (the admissible position of the vehicle center in $\mathbb{R}^3$) determined by such a constraint can be handled by algorithm (4.27).



(a) The $x - y - \theta$ view.



(b) The $x - y$ view.

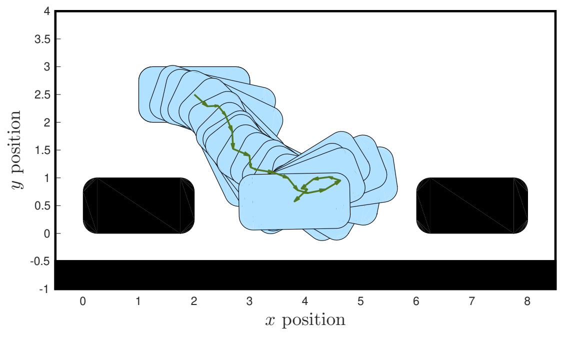
Figure 5.6: Collision area when $\Delta = 0.5$. In (b), the gray area is the $x - y$ plane projection of the 3D collision area, and the two black rectangles represent the bodies of rear and front vehicle.

By Theorem 5.3, if parallel parking is robustly realizable with the given marginal space, we can always synthesize a control strategy using a sufficiently small precision without calculating the Lipschitz constant. To see if the specifications in these two parking scenarios are realizable, we use different precision control parameters. The corresponding control synthesis results regarding the number of partitions ($\#\mathcal{P}_{1,2}$) and the run time ($t_{1,2}$) are summarized in Table 5.2.

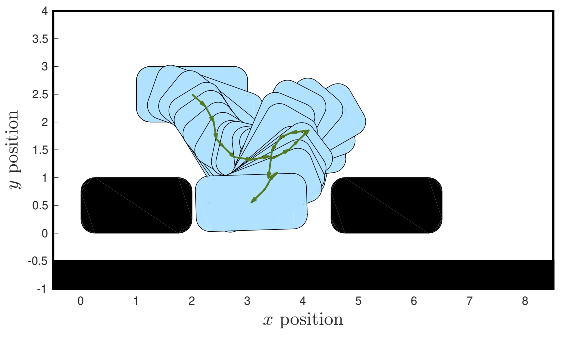
For both scenarios, the vehicle can be successfully parked into the target spot from any point of the free workspace. The controlled parking trajectories with the resulting memoryless control strategies are presented in Figure 5.7, which all meet the parallel parking specification.

Table 5.2: Control synthesis of the parallel parking problem with different precisions.

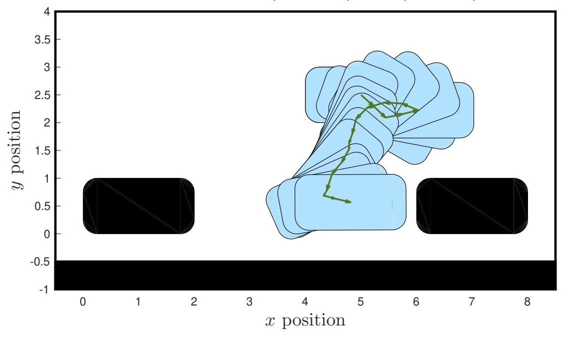
ε	$\#\mathcal{P}_1$	t_1 (s)	$\#\mathcal{P}_2$	t_2 (s)
0.07	176786	102.93	–	–
0.06	176666	103.19	1,797,027	295.68
0.02	203166	127.44	1,832,589	327.50
0.01	274694	176.20	1,920,929	427.48



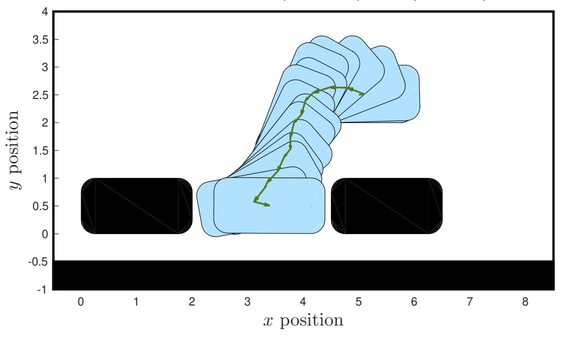
(a) $\Delta = 2$, $(x_0, y_0) = (2, 2.5)$.



(b) $\Delta = 0.5$, $(x_0, y_0) = (2, 2.5)$.



(c) $\Delta = 2$, $(x_0, y_0) = (5, 2.5)$.



(d) $\Delta = 0.5$, $(x_0, y_0) = (5, 2.5)$.

Figure 5.7: Controlled parking trajectories from an initial condition (x_0, y_0) with wide and narrow marginal parking spaces.

When the marginal parking space Δ is 0.5, we need a control synthesis precision no greater than 0.06 so that a memoryless control strategy can be generated. Additionally for this specific example, using a smaller ε only increases the winning set by adding intervals close to the boundary of the free workspace.

Such a parallel parking task can also be solved by using a piecewise-affine controller defined on a pre-designed triangular partition of the configuration space [99], which contains the initial states of the car. The main advantage of using the proposed method is that the partition of the state space is performed automatically. As a result, we do not need to re-design the partition for a different parking scenario. In addition, the control design is based on the nonlinear model as opposed to different linearizations of the nonlinear model on different polytopes in the state space.

5.5.3 Motion Planning

For the same vehicle model (5.29), we now consider a motion planning problem: steer the vehicle to the target set while avoiding obstacles in a maze.

Let the workspace $\mathbb{X}$ of the maze be $\mathbb{X} = [0, 10] \times [0, 10] \times [-3.4, 3.4]$ and the range of input controls be $\mathbb{U} = [-1, 1] \times [-1, 1]$. There are static obstacles distributed over the entire workspace $\mathbb{X}$. The 2-D view of the maze is shown in Figure 5.8. The exit of the maze is the area at the bottom right corner, which is given as $\Omega = [9, 9.5] \times [0, 0.5] \times [-3.4, 3.4]$. Then this motion planning problem can be considered as a control synthesis problem with respect to $\varphi = \Diamond G$ for the system $\mathcal{S}$ with transition relation R determined by (5.30) and (5.31) and labeling function (4.2).

Same as in the parallel parking example, we use a sampling time $\tau_s = 0.3$s and a grid size $\eta = 0.3$ for the control space $\mathbb{U}$. The precision parameter ε of Algorithm 5.1 is set to be 0.2 in executing the proposed algorithm (4.19), and a closed-loop path of the vehicle with the resulting memoryless control strategy in the form of (5.13) is shown in Figure 5.8.

5.5.4 Comparison on Performance

We may find the proposed specification-guided method less efficient than abstraction-based methods from the complexity analysis for worst cases. In this section, we will show how well the proposed method performs in practice.

First of all, we compare in Table 5.3 the run time of invariance control synthesis for the boost DC-DC converter using our algorithm with the ones by using abstraction-based methods (reported in [118]). In terms of efficiency, our algorithm outperforms other existing methods.

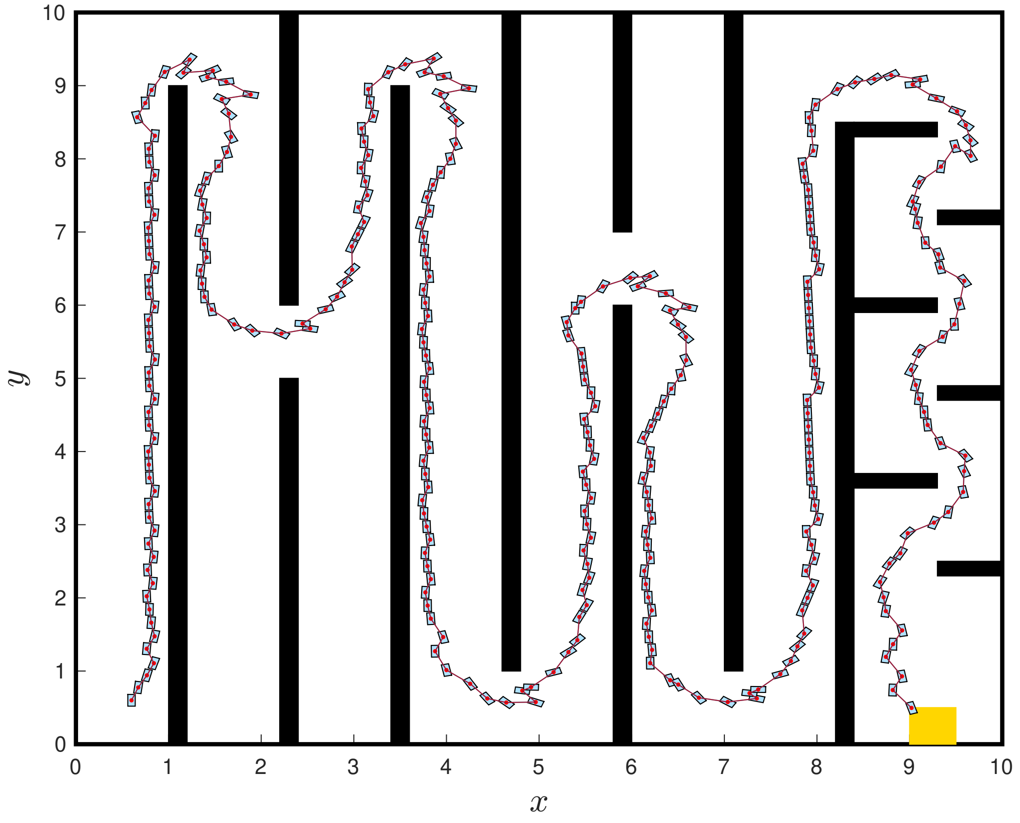


Figure 5.8: Motion planning in a planar area with obstacles: a controlled 2-D trajectory of the vehicle that leads from the initial condition $x_0 = 0.6, y_0 = 0.6, \theta_0 = \pi/2$. The obstacles are represented by black areas.

Table 5.3: Comparison of run times of invariance control synthesis for the boost DC-DC converter. "t_{abs}"=*the time for computing abstractions*, and "t_{syn}"=*the time for control synthesis.*

	CPU [GHz]	t_{abs}(s)	t_{syn}(s)
Pessoa[66]	i7 3.5	478.7	65.2
SCOTS[118]	i7 3.5	18.1	75.4
CoSyMA[93]	N/A	N/A	8.32
ROCS[83]	i3 3.6	0	0.077

We also compare the performance of solving reachability and reach-and-stay problems using our proposed method with abstraction-based methods (implemented in SCOTS [118]) on solving different benchmarking examples. The results are shown in Table 5.4. The column of #Iter indicates the number of outer loops and the total number of inner loops (in the bracket) running (4.24). The time for abstraction-based control methods is split into the part for abstraction (indicated as *Abst*) and the one for synthesis (as *Syn*).

Table 5.4: Performance comparison tests. TO=*time out* ($>$ 86400s) and "–" = *control synthesis fails.*

Examples	Parameters			ROCS			SCOTS				
	n	N_U	ε	N_S	#Iter	time(s)	N_S	N_T	#Iter	time(s)	
										Abst	Syn
DC-DC converter	2	2	0.005	22433	76(529)	0.53	40401	291068	84(671)	0.69	15.90
			0.001	162261	76(272)	3.48	1002001	7243320	77(431)	29.83	481.97
Motion planning	3	49	0.2	280291	381(1)	151.01	91035	3.73×10^7	–	82.80	–
			0.1	1850830	297(1)	1062.97	724271	2.95×10^8	313(2266)	2004.66	17568.2
Parallel parking $\Delta = 0.5$	3	49	0.02	1832589	133(1)	327.50	10075125	TO	TO	TO	TO
Parallel parking $\Delta = 2$			0.07	167155	123(8)	94.32	83025	3.277×10^7	–	73.14	–

Our proposed method outperforms abstraction-based methods in those examples. In the *motion planning* example, using a grid size of 0.1 succeeds in synthesis while using 0.2 fails for abstraction-based methods because abstractions are more over-approximated for larger grid size. In contrast, our proposed method solves the problem in 151 seconds by using $\varepsilon = 0.2$. This is because the minimum width of the partitions can be less than 0.2 by the subdivision scheme of Algorithm 5.1. As opposed to the *motion planning* case where obstacles are distributed evenly across the state space, the constraints for parallel parking are highly nonlinear and only posed to a corner of the state space, and varying the discretization precision of the state space will save computational time in a great deal. Such a difference in those two case settings explains why the gain in time efficiency by using our method is more profound in the parallel parking cases.

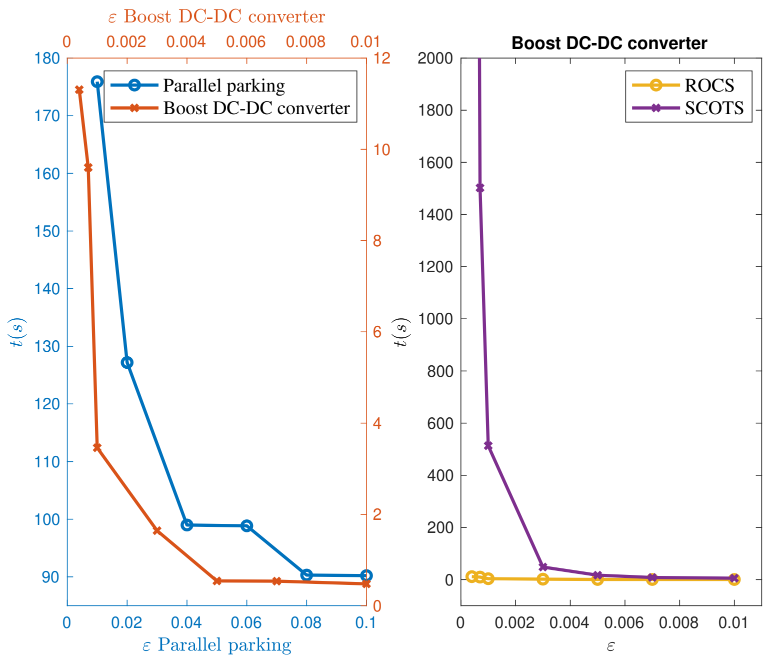


Figure 5.9: Changes of run time under different precisions.

As seen in (5.23) and (5.26), both methods are equivalently sensitive to the size of the discretized systems. The experimental results shows that the worst case as in (5.26) is rather pessimistic in practice and our proposed method is more scalable to the discretization precision than abstraction-based methods. Analyzing the example of *boost DC-DC converter*, we can observe in the right-hand side of Figure 5.9 that the run time of the proposed method changes slowly while the one for abstraction-based method explodes as precision ε decreases. The left-hand side of Figure 5.9 compares the run time of the proposed method for two cases of different sizes, which indicates the dimensionality problem of the proposed method.

5.6 Convergence in Set Approximations

Previous sections in this chapter focus on robust completeness in control synthesis, which is guaranteed to find a memoryless control strategy if the given specification is robustly realizable for system $\mathcal{S}$. It is a relaxation concerned with difficulty of approximation error of predecessors under nonlinear dynamics. In this section, we discuss the convergence of using the outer interval approximation (5.3) in computing maximal controlled invariant set. We can show that the maximal controlled invariant set can be outer approximated with arbitrary precision.

Theorem 5.4. Let $\Omega \subseteq \mathbb{X}$ be compact and Assumption 3.1 or 3.2 holds. Then (4.8) with $\widehat{\text{Pre}} = [\overline{\text{Pre}}]^\varepsilon$ (or $\widehat{\text{Pre}} = [\overline{\text{Pre}}_\mu]^\varepsilon$), which is given in (5.3), terminates in a finite number of steps with an output $[\overline{\mathcal{I}}]^\varepsilon_\infty$. Furthermore, the output $[\overline{\mathcal{I}}]^\varepsilon_\infty$ is an union of intervals satisfying the following properties:

(i) If $0 < \varepsilon_1 < \varepsilon_2$, $\mathcal{I}_\infty(\Omega) \subseteq [\overline{\mathcal{I}}]^{\varepsilon_1}_\infty \subseteq [\overline{\mathcal{I}}]^{\varepsilon_2}_\infty$;

(ii) $\mathcal{I}_\infty(\Omega) = \lim_{\varepsilon \to 0} [\overline{\mathcal{I}}]^\varepsilon_\infty$.

Proof. We use subscript j to denote the jth iteration ($j \in \mathbb{N}$) of algorithm (4.8). The corresponding sets of Algorithm 5.1 in the jth iteration are denoted by $\underline{A}_j$ $A_{c,j}$, ΔA_j, and B_j, respectively.

To see (i), we first prove $\mathcal{I}_\infty(\Omega) \subseteq [\overline{\mathcal{I}}]^\varepsilon_\infty$ for all $\varepsilon > 0$. For the sake of contradiction, let $y \in \mathcal{I}_\infty(\Omega)$ but $y \notin B_N$ for a $N \in \mathbb{Z}^+$. Then $y \in \Omega \setminus B_N$. According to the algorithm, $\forall z \in \Omega \setminus B_N$, there must be a step $0 < j \leq N$ such that $[f](z,u) \cap \Omega = \emptyset$ for all $u \in \mathbb{U}$. This indicates that $z \notin \mathcal{I}_\infty(\Omega)$, which is a contradiction. Thus $\mathcal{I}_\infty(\Omega) \subseteq [\overline{\mathcal{I}}]^\varepsilon_\infty$. Next we prove $[\overline{\mathcal{I}}]^{\varepsilon_1}_\infty \subseteq [\overline{\mathcal{I}}]^{\varepsilon_2}_\infty$ by induction. Consider the first two steps: $B_0^{\varepsilon_1} = B_0^{\varepsilon_2} = \Omega$. Since $0 < \varepsilon_1 < \varepsilon_2$, some intervals in $\Delta A_1^{\varepsilon_2}$ will be divided into finer boxes and are possible to be included in $A^{\varepsilon_1}_{c,1}$, and thus $A^{\varepsilon_2}_{c,1} \subseteq A^{\varepsilon_1}_{c,1}$. Together with $B_1^{\varepsilon_1} = B_0^{\varepsilon_1} \setminus A^{\varepsilon_1}_{c,1}$, and $B_1^{\varepsilon_2} = B_0^{\varepsilon_2} \setminus A^{\varepsilon_2}_{c,1}$, we have $B_1^{\varepsilon_1} \subseteq B_1^{\varepsilon_2}$. Assume $B_j^{\varepsilon_1} \subseteq B_j^{\varepsilon_2}$ for any step $1 \leq j < N$. Then $A^{\varepsilon_2}_{c,j} \subseteq A^{\varepsilon_1}_{c,j}$, which gives $B^{\varepsilon_1}_{j+1} \subseteq B^{\varepsilon_2}_{j+1}$. Hence (i) is proved.

To show (ii), we consider a decreasing sequence $\{\varepsilon_j\}_{j=1}^\infty$ with $\varepsilon_j > 0$ and $\lim_{j\to\infty} \varepsilon_j = 0$. Since $[\overline{\mathcal{I}}]^{\varepsilon_j}$ is compact, $\lim_{\varepsilon_j \to 0} [\overline{\mathcal{I}}]^{\varepsilon_j}$ exists and is given by the compact set $\bigcap_{j=1}^\infty [\overline{\mathcal{I}}]^{\varepsilon_j}$. Let $[\overline{\mathcal{I}}] = \bigcap_{j=1}^\infty [\overline{\mathcal{I}}]^{\varepsilon_j}$. If every $[\overline{\mathcal{I}}]^{\varepsilon_j}$ is nonempty, then $[\overline{\mathcal{I}}]$ is nonempty. By (i), $\mathcal{I}_\infty(\Omega) \subseteq [\overline{\mathcal{I}}]^{\varepsilon_j}$ for all $j \geq 1$. Then it is clear that $\mathcal{I}_\infty(\Omega) \subseteq [\overline{\mathcal{I}}]$.

We claim that $[\overline{\mathcal{I}}] \subseteq \mathcal{I}_\infty(\Omega)$. If this is not true, then there exists $y \in [\overline{\mathcal{I}}]$ such that $f(y,u) \notin [\overline{\mathcal{I}}]$ for all $u \in \mathbb{U}$, i.e., $f(y,u) \in [\overline{\mathcal{I}}]^c$, which is the complement of $[\overline{\mathcal{I}}]$ and is open. Then it follows that there exists $r > 0$ such that $\mathcal{B}_r(f(y,u)) \subseteq [\overline{\mathcal{I}}]^c$ for all $u \in \mathbb{U}$. Furthermore, by Definition 3.2, there exists a J_1 sufficiently large such that $\mathcal{B}_r(f(y,u)) \cap [\overline{\mathcal{I}}]^{\varepsilon_j} = \emptyset$ for all $u \in \mathbb{U}$ and $j \geq J_1$. Then it is only possible that $y \in [x] \in \Delta A_j, j \geq J_1$. Since $f(\cdot,u)$ is a continuous function (under Assumption 3.1 or 3.2), there exists a J_2 such that $[f]([x],u) \subseteq \mathcal{B}_r(f(y,u))$ for all $u \in \mathbb{U}$ and $[x] \in \Delta A_j, j \geq J_2$. Then for all $j \geq \max\{J_1, J_2\}$, we have $[f]([x],u) \cap B_N = \varnothing$, which is contradictory with the fact that $y \in \Delta A_j$. Hence, (ii) is true. $\square$

Theorem 5.4 indicates that the exact maximal invariant sets can be outer approximated in an arbitrary precision, as illustrated in the following example.

Example 5.2. Consider again the system in Example 4.1. The maximal positively invariant set within Ω is bounded by two trajectories, which is marked by the two red curves. Figure 5.10 shows the approximation results with different choices of precision ε (ε =0.05, 0.01, 0.0063, 0.001, respectively). It can be observed that the approximation error decreases as ε becomes smaller.

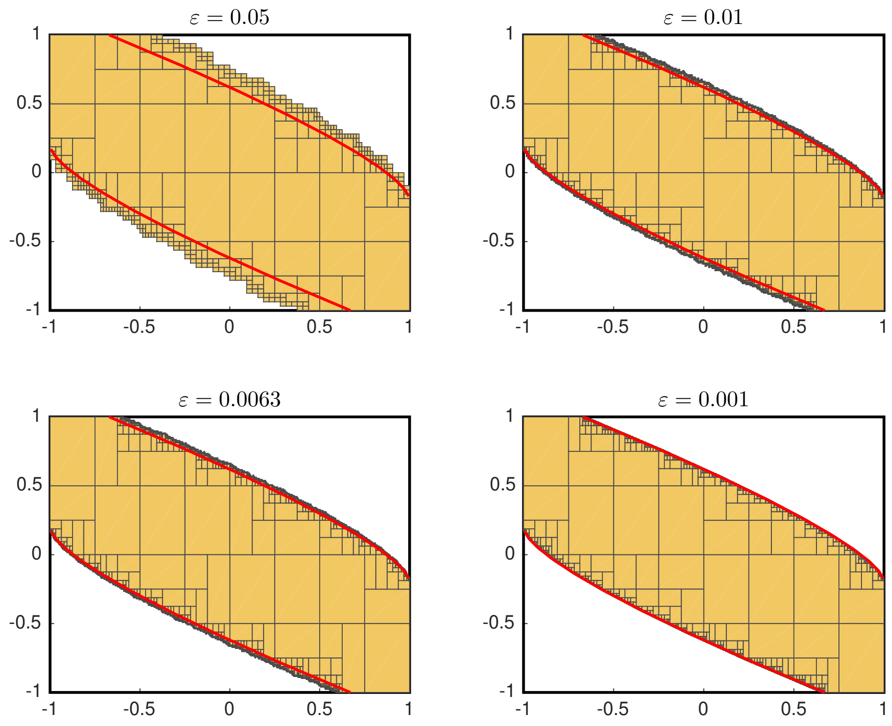


Figure 5.10: Outer approximations of $\mathcal{I}_\infty(\Omega)$ with different precision parameters.

5.7 Summary

We presented in this chapter an adaptive state space partition scheme integrating interval implementation of the predecessor map so that the control synthesis algorithms (4.8), (4.19) and (4.27) in Chapter 4 can terminate in finite numbers of iterations and are guaranteed to be sound and robustly complete.

Due to the generality of set representation using intervals, assumptions on the form of the nonlinear dynamics (e.g. polynomial or linear affine $f(x, u) = Ax + g(x)u$) and the stability properties (e.g. incremental stability [4]) are not necessary for the approximation of predecessors in the proposed control synthesis algorithms. Similar to our work in control synthesis, in the verification of dynamical system properties, interval arithmetic has been used to prove the quasi-decidability of safety properties [112, 111] and δ-complete analysis for bounded reachability [46] of hybrid systems.

Specifically in invariance control, computation of maximal controlled invariant sets is not easy even for linear systems because maximal controlled invariant sets are not guaranteed to be finitely determined, except for the special cases where the linear system is λ-*contractive* in a compact and convex set around the origin [18]. For control purposes, invariant inner approximations are more desirable, because, different from outer approximations, they are subsets of states that can be controlled invariant, for which an invariance controller exists. Finitely determined invariant inner approximations can be obtained by computing the *null-controllable sets* (i.e., the set of states that can be controlled to the origin in finite time) [55]. The δ-complete inner approximation of the maximal controlled invariant set $\mathcal{I}_\infty(\Omega)$ inside a given set Ω for linear disturbed systems proposed in [117] relaxes the requirement that Ω must surround the origin. The idea in [117] is similar to our proposed robustly complete method, but it only applies to linear systems. For nonlinear systems, invariance control via analytical methods include constructing barrier certificates [134] by sum-of-squares (SOS) techniques [64]. These methods usually work for polynomial dynamics or particular forms of feedback control functions.

For reachability and reach-and-stay control problems, integrating constraints and bounded perturbations in the stage of controller design was studied in [16] for linear systems and extended to nonlinear systems [102] where the reach-and-stay requirement is relaxed to reach a robustly controllable super set containing the target set when the target set is not controlled invariant. As a result, the guarantee of the reach-and-stay property is lost in [102], which is different from the reach-and-stay problem considered in this thesis. Most of works in this topic are dependent on the assumption that the target set is controlled invariant [32, 133].

As explained in Section 2.4, abstraction-based methods are applicable in the control synthesis problems discussed so far. These methods are systematic and usually rely on milder assumptions than analytical methods. Similar to abstraction-based methods, our method based on interval computation also work on finite partitions of the continuous state space. The major difference between abstraction-based methods and our method is that the partition is adaptively performed with respect to both dynamics and specification under the proposed scheme. Compared with the works with abstraction refinement mechanisms [49, 61, 98], in which parameters need to be chosen empirically or synthesis does not always terminate in finite time, we devise a scheme for adaptive tuning of discretization precision under a given threshold related to system robustness level.

With the numerical experiments shown in this chapter, the advantages of the proposed method can be concluded as:

(i) Compared with the existing abstraction-based methods, it has better practical time efficiency because of the adaptive partitioning scheme.

(ii) Easy operation on set unions and intersections for the approximation of predecessors. Many of the geometric representations such as polyhedra, and ellipsoids can not be preserved under set union and/or intersection operation.

Convergence of outer approximations of maximal controlled invariant sets is shown based on interval computation, which is consistent with the general conclusion given in [31] that maximal controlled invariant sets are outer computable and reachable sets are inner computable.

Chapter 6

Robustly Complete Control Synthesis with LTL Formulas

In many situations, control specifications beyond simply invariance and reachability need to be considered. For example, specifications such as “visiting different work areas in order infinitely often and avoid obstacles” are frequently considered in motion planning. Control of an elevator or a network of distributed resources involves a request-response pattern. This motivates the study of control synthesis for dynamical systems to realize the properties that require ordering, liveness or reactivity. Such properties can be well captured by general LTL formulas [8].

Current solution to a general LTL control synthesis problem is based on abstractions: a finite transition system (or abstraction) that approximates the continuous-state dynamics is often constructed first. Discrete control synthesis algorithms, which are rooted in graph or game theory [89, 142], are applied to the product system of the abstraction and the Büchi Automaton (BA) (see [131]) translated from the LTL specification afterwards [10]. A control synthesis algorithm is sound and complete if it flawlessly determines the winning set. Soundness and completeness can be achieved at the abstraction level as discrete control synthesis is a direct application of infinite game problems. The gap between the real infinite-state dynamical systems and their finite abstractions leaves the following question open:

Is it possible to make LTL control synthesis sound and complete based on nonlinear dynamics?

As we have shown in Chapter 4, memoryless control strategies are sufficient for control synthesis with respect to the specifications that are restricted to invariance or reachability, but general LTL control synthesis requires finite memories [104, 19]. It is then natural to ask

What is the controller structure for a general LTL formula? How do the memories reflected in the structure?

Time complexity is a major concern in LTL control synthesis. A promising finite abstraction of nonlinear dynamics is usually huge in size, and control synthesis on a product system would be intractable because the number of states is the multiplication of the sizes of both the abstraction and the BA. On the other hand, the reduction in the size of a specification can also reduce the complexity. Research has been focusing on system decomposition [91, 70], hierarchical abstractions [62] and parallel computation [68, 34], but not at the specification level. This motivates us to raise another question:

Can we reduce the complexity by exploring the hierarchy in the specification if the system cannot be further decomposed?

The goal of this chapter is to answer these questions for the control synthesis with respect to a general class of LTL formulas. As opposed to using the product system of a finite abstraction of the original system and the automaton translated from the given LTL formula, we look at this control synthesis problem more directly: characterize the winning set by a fixed-point algorithm based on the automata structure of the given formula. We also show that such an algorithm leads to a similar controller structure to the feedback control automaton in [71, 10] and the LTL control synthesis can be made sound and robustly complete for the situations where accurate computation of predecessors is nontrivial.

6.1 From LTL To Büchi Automata

As we have seen in Chapter 2, LTL is a formalism describing sets of words over an alphabet Σ that share common properties (e.g. invariance, and liveness). An automaton is a machine that accepts words with certain patterns. In this sense, LTL formulas can be represented by automata.

Definition 6.1. A Finite Automaton (FA) is a quintuple $\mathcal{A} = (Q, \Sigma, r, q_0, F)$, where

- Q is a finite set of states,
- Σ is a finite alphabet,
- $r : Q \times \Sigma \to 2^Q$ is the state transition function,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is a set of accepting states.

An automaton always proceeds from the initial state q_0 by reading an input word over Σ. A *run* of an automaton $\mathcal{A}$ under an input word $\mathbf{w} = \sigma_0 \cdots \sigma_l$ ($l \in \mathbb{N}$) is a sequence of states in Q, denoted by $\varrho = v_0 \cdots v_l$, that satisfies $v_{i+1} \in r(v_i, \sigma_0)$ and $v_i \in Q$ for all $i \in \{1, \cdots, l\}$. Denote

$$\varrho[i] \triangleq v_i, \quad \varrho[i,j] \triangleq v_i \ldots v_j, 0 < i < j.$$

If $r(q, \sigma)$ is a singleton for all $q \in Q$ and $\sigma \in \Sigma$, then $\mathcal{A}$ is *deterministic*. Otherwise $\mathcal{A}$ is said to be *nondeterministic*.

Although an FA contains only a finite number of states, it can run over finite or infinite words. Specifically if an automaton $\mathcal{A}$ runs over infinite words, then $\mathcal{A}$ is called an ω-*automaton*. An input word $\mathbf{w}$ is said to be *accepted* by $\mathcal{A}$ if the resulting run $\varrho = v_0 v_1 \cdots$ satisfies the accepting condition of $\mathcal{A}$, and the run ϱ is said to be *successful* for $\mathcal{A}$.

Definition 6.2 (Büchi Automaton)**.** An ω-automaton $\mathcal{A}$ is a BA if the accepting condition is: A run ϱ is successful for $\mathcal{A}$ if and only if ϱ visits at least one of the states in F infinitely many times, i.e.,

$$\mathrm{Inf}(\varrho) \cap F \neq \emptyset, \tag{6.1}$$

where $\mathrm{Inf}(\varrho) = \{v \in Q : \forall i, \exists j > i, \text{s.t. } v = \varrho[j]\}$ represents the set of states occurring infinitely many times during the run ϱ.

The set of all accepted words of a BA $\mathcal{A}$ forms the *language* of $\mathcal{A}$, denoted by $\mathcal{L}(A)$. By the accepting condition, termination of a BA is considered a failure.

A BA $\mathcal{A}$ is called a Deterministic Büchi Automaton (DBA) if $\mathcal{A}$ is deterministic and Nondeterministic Büchi Automaton (NBA) if $\mathcal{A}$ is nondeterministic. The language of an NBA $\mathcal{A}_{\mathrm{D}}$ is a super class of the one of an DBA $\mathcal{A}_{\mathrm{N}}$, i.e., $\mathcal{L}(\mathcal{A}_{\mathrm{D}}) \subseteq \mathcal{L}(\mathcal{A}_{\mathrm{N}})$. Therefore, NBA are more expressive than DBA.

An automaton $\mathcal{A} = (Q, \Sigma, r, q_0, F)$ can be presented as a *directed graph* $\mathcal{G} = (V, E)$, where

- $V = Q$ is a set of nodes, and
- $E = \{(v, \sigma, v') : \exists \sigma \in \Sigma, v' \in r(v, \sigma), v, v' \in V\}$ is a set of directed edges.

The direction of an edge (v, σ, v') is determined by the transition relation r, and the node v should proceed before v' because $v' \in r(v, \sigma)$. The input letter σ is considered as the *label* of the edge (v, σ, v'). The set of the labels of outgoing edges of a node v is defined as $\mathrm{Out}(v) = \{\sigma \in \Sigma : r(v, \sigma) \neq \emptyset\}$.

Let V' be a subset of V. The new graph $\mathcal{G}' = (V', E')$ is called a *subgraph* of $\mathcal{G}$, where $E' \subseteq E$ is the set of edges between nodes in V'. A sequence of nodes connected by directed edges is called a *path* in a directed graph. The graph is said to be *strongly connected* if there exists a path between any two nodes of the graph. When the graph $\mathcal{G}$ itself is not strongly connected, it is still possible that there exists a strongly connected subgraph. Such a subgraph is called a *Strongly Connected Component (SCC)* of $\mathcal{G}$. A single node with a self loop can be considered as a trivial SCC. If the graph $\mathcal{G}$ has no directed cycles, then we call $\mathcal{G}$ a Directed Acyclic Graph (DAG).

Figure 6.1 demonstrates the graph representation of a BA. The nodes (or states) are pictured as circles and the accepting states are specifically marked by double circles. Directed edges are represented by arrowed lines pointing from a node v to v' that satisfy the transition function. The state evolution of an automaton always start from the initial state q_0.

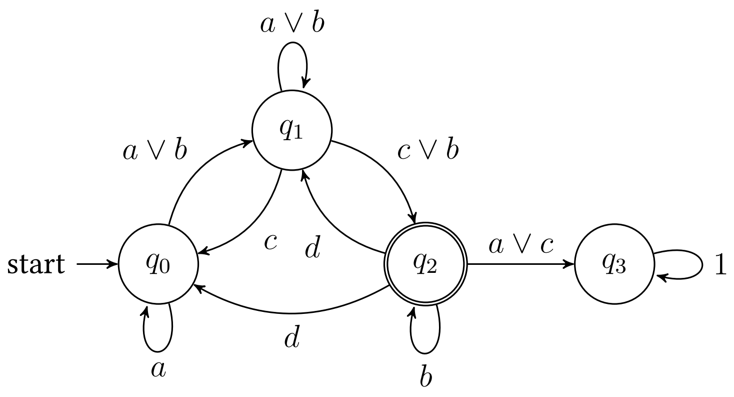


Figure 6.1: Graph representation of an NBA with the alphabet $\Sigma = \{a, b, c, d\}$. The node q_2 is a unique accepting state. The edge from q_2 to q_3 will incur an unsuccessful run of the NBA.

Every LTL formula φ built on a set AP of atomic propositions has an equivalent BA with an input alphabet $\Sigma = 2^{AP}$, which only accepts the words specified by φ, i.e., $\mathcal{L}(\varphi) \subseteq \mathcal{L}(\mathcal{A})$. Methods such as tableau construction [48] and the algorithm based on the conversion to generalized Büchi automata [47] have been developed to translate an LTL formula into a BA. Not all LTL formulas can be translated into DBA.

Example 6.1. The reach-and-stay objective $\varphi_{\text{rs}} = \Diamond\Box G$ is also called *co-Büchi* in the field of two player infinite games. It is a dual of Büchi objective $\varphi = \Box\Diamond G$, which requires that G holds infinitely often. Reach-and-stay formulas can not translated into a DBA, but only an NBA. Figure 6.2 shows the equivalent NBA of the formula φ_{rs}.

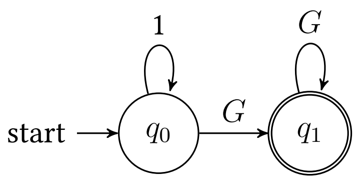
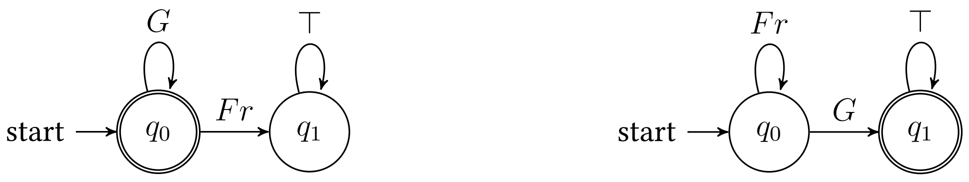


Figure 6.2: The equivalent NBA of φ_{rs} with $\Sigma = \{G, Fr\}$. The nondeterminism exists in the out edges of q_0: after reading an input letter G, the state of the NBA can either stay at q_0 or transit to q_1.

Even though DBA is insufficient in characterizing a general LTL formula, among 55 most expressive LTL patterns identified in [38], there are 52 of them belong to the language of DBA.

Example 6.2. The invariance and reachability formulas $\varphi_{\mathrm{s}} = \Box G$ and $\varphi_{\mathrm{r}} = \Diamond G$ can be translated into DBA, which is shown in Figure 6.3.



(a) The equivalent DBA for φ_{s}.

(b) The equivalent DBA for φ_{r}.

Figure 6.3: The DBA translations for the invariance and reachability formulas. The input alphabet is $\Sigma = \{G, Fr\}$.

6.2 Control Structure with Finite Memory

In this chapter, we limit our scope to the LTL formulas that can be translated into DBA, which are called *DBA-recognizable*, and denote by $\mathcal{A}_\varphi = (Q, \Sigma, r, q_0, F)$ the equivalent DBA of an LTL specification φ, where especially

- $Q = \left\{q_0, \cdots, q_{|Q|-1}\right\}$ and $|Q|$ is the number of states in $\mathcal{A}_\varphi$,
- $\Sigma = 2^{AP}$ is the input alphabet. An input symbol $\sigma \in \Sigma$ is usually represented by a propositional formula over the set AP of atomic propositions for φ.

As opposed to the conventional invariance and reachability control problems, we will see in this section that a winning control strategy for a DBA-recognizable LTL formula φ requires finite memories, which is induced by the ordering property specified by φ.

Without loss of generality, we assume that $\mathcal{A}_\varphi$ is *nonblocking*, i.e., $\text{Out}(q) \neq \emptyset$ for all $q \in Q$, since we can always construct a nonblocking one for any BA [8].

6.2.1 $\mathcal{S}$-Domains of Automaton States

Let $\mathcal{A}_\varphi$ be the equivalent DBA of an LTL formula φ and $q \in Q$ be an arbitrary state of $\mathcal{A}_\varphi$. Then by the determinism of $\mathcal{A}_\varphi$ and the nonblocking property, every state has at least one outgoing edge and

$$\sigma \wedge \sigma' = \bot, \forall \sigma, \sigma' \in \text{Out}(q), \qquad \bigvee_{\sigma \in \text{Out}(q)} \sigma = \top. \tag{6.2}$$

We consider traces of system $\mathcal{S}$ as input words to $\mathcal{A}_\varphi$. Hence, given a control signal $\mathbf{u} = \{u_t\}_{t=0}^{\infty}$ and a sequence of disturbance $\mathbf{d} = \{d_t\}_{t=0}^{\infty}$, the resulting run $\varrho = \{v_t\}_{t=0}^{\infty}$ of $\mathcal{A}_\varphi$ is obtained explicitly by (for all $t \in \mathbb{Z}^+$)

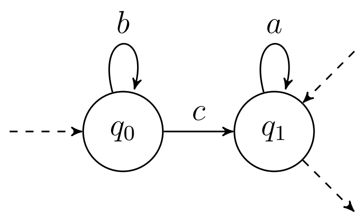
$$\begin{cases} v_0 = q_0, \ v_t = r(v_{t-1}, L(x_{t-1})), \ v_t \in Q \\ x_t = f(x_{t-1}, u_{t-1}) + d_{t-1}, \ x_i \in \mathbb{X}. \end{cases} \tag{6.3}$$

Note that if $L(x_0) = \sigma$, the state v of $\mathcal{A}_\varphi$ changes from q_0 to $r(q_0, \sigma)$ immediately, and $v = r(q_0, \sigma)$ until the next time step where the state v changes instantly according to the input symbol, which is the label of the system state x. An intuitive illustration of (6.3) is given in Figure 6.4.

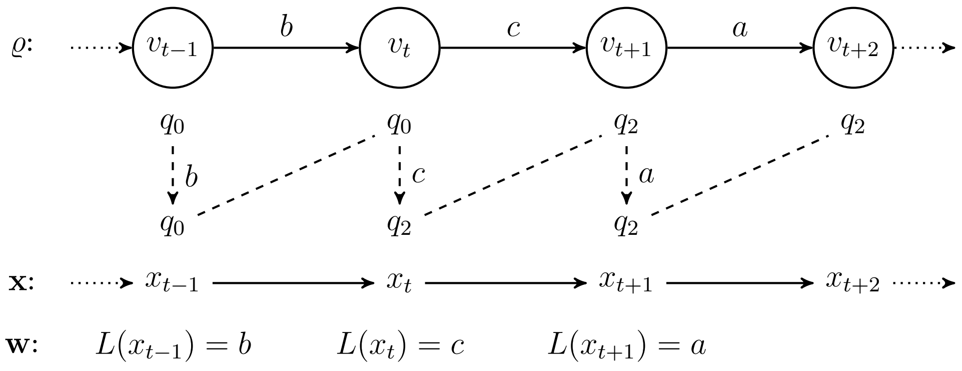
In order to control system $\mathcal{S}$ so that the resulting traces are accepted by $\mathcal{A}_\varphi$, each transition along the successful runs of $\mathcal{A}_\varphi$ needs to be executed sequentially. Those transitions, however, cannot be assigned deliberately as they have to satisfy the transition relation R of system $\mathcal{S}$. This also implies that, for each $q \in Q$, the corresponding system state x is restricted to a certain subset of the state space $\mathbb{X}$. To capture such a set, we introduce the following definition.

Definition 6.3. Let $q \in Q$ and $x \in \mathbb{X}$ be a state of DBA $\mathcal{A}_\varphi$ and system $\mathcal{S}$ at some time instance $j \geq 0$, respectively. Then x belongs to the $\mathcal{S}$-*domain* of q, written as $W_{\mathcal{S}}(q)$, iff there exists a control strategy κ in the form of (2.7) such that any run $\varrho = \{q_t\}_{t=0}^{\infty}$ of $\mathcal{A}_\varphi$ with $q_j = q$ generated by (6.3) under a control signal conform to κ satisfies that $\text{Inf}(\varrho) \cap F \neq \emptyset$.

The winning set of an LTL formula φ is, by definition, the $\mathcal{S}$-domain of the initial state q_0 of $\mathcal{A}_\varphi$, i.e., $\text{Win}_{\mathcal{S}}(\varphi) = W_{\mathcal{S}}(q_0)$. Therefore, the problem of computing $\text{Win}_{\mathcal{S}}(\varphi)$ can be reduced to computing $W_{\mathcal{S}}(q_0)$.



(a) Part of a DBA $\mathcal{A}_\varphi$.



(b) The relationship between a run ϱ of A_φ and a solution $\mathbf{x}$ of system $\mathcal{S}$.

Figure 6.4: The connection between system $\mathcal{S}$ and the equivalent DBA $\mathcal{A}_\varphi$ of a given LTL formula φ. Assume that at some time $t-1 \in \mathbb{N}$, the state of $\mathcal{A}_\varphi$ is at q_0 and the label of x_{t-1} is $b \in \Sigma$, i.e., $v_{t-1} = q_0$ and $L(x_{t-1}) = b$. Part (b) shows how the partial sequence $v_{t-1}v_tv_{t+1}v_{t+2}$ is driven by $x_{t-1}x_tx_{t+1}x_{t+2}$ according to the relevant part of $\mathcal{A}_\varphi$ shown in part (a).

6.2.2 Fixed-Point Characterization of $\mathcal{S}$-Domains

It is easy to see from (6.3) that the connection between dynamical system $\mathcal{S}$ and the targeted DBA $\mathcal{A}_\varphi$ is through the labeling function L. Since the outgoing edges satisfy (6.2), which is the same as (2.5), the set $\text{Out}(q)$ of every state $q \in Q$ forms a partition $\mathcal{P}(q) = \{L^{-1}(\sigma)\}_{\sigma \in \text{Out}(q)}$ of the state space $\mathbb{X}$ through the labeling function L as defined in (2.6).

Because of the graph structure of a DBA $\mathcal{A}_\varphi$, $\mathcal{S}$-domains of different automaton states are related with one another by the transitions among them. Any state $x \in W_\mathcal{S}(q)$ can be controlled to the $\mathcal{S}$-domain of one of the succeeding states of q in $\mathcal{A}_\varphi$. Suppose that a state q of a DBA $\mathcal{A}_\varphi$ has three outgoing edges σ_1, σ_2 and σ_3 as shown in Figure. 6.5. Any state $x \in \mathbb{X}$ belongs to $W_\mathcal{S}(q)$ if it can be controlled to any one of the following regions in the next time step:

$$L^{-1}(\sigma_2) \cap W_\mathcal{S}(q'),\ L^{-1}(\sigma_3) \cap W_\mathcal{S}(q''),\ L^{-1}(\sigma_1) \cap W_\mathcal{S}(q).$$

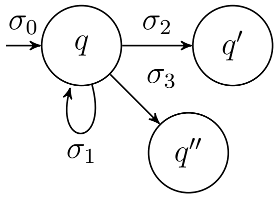


Figure 6.5: Transitions in a DBA.

Let $\mathbf{M}$ be an n_1 by n_2 ($n_1, n_2 > 0$) matrix of symbols from Σ and

$$V = \begin{bmatrix} V_1 \\ \vdots \\ V_{n_2} \end{bmatrix}, \quad W = \begin{bmatrix} W_1 \\ \vdots \\ W_{n_2} \end{bmatrix}$$

be two vectors of subsets of $\mathbb{X}$. Denote by m_{ij} the element at the ith row and $j$the column of $\mathbf{M}$. Define

$$W + V \triangleq \begin{bmatrix} W_1 \cup V_1 \\ \vdots \\ W_{n_2} \cup V_{n_2} \end{bmatrix}, \tag{6.4}$$

$$W - V \triangleq \begin{bmatrix} W_1 \setminus V_1 \\ \vdots \\ W_{n_2} \setminus V_{n_2} \end{bmatrix}, \tag{6.5}$$

$$V \preceq W \triangleq V_i \subseteq W_i,\ i = 1, \ldots, n_2, \tag{6.6}$$

$$W = V \triangleq W_i = V_i,\ i = 1, \ldots, n_2, \tag{6.7}$$

$$W' = \begin{bmatrix} W'_1 \\ \vdots \\ W'_{n_1} \end{bmatrix} = T^\delta(\mathbf{M}, W), \tag{6.8}$$

where

$$W'_i = \text{Pre}^\delta \left(\bigcup_{j=1}^{n_2} L^{-1}(m_{ij}) \cap W_j \right),\ i = 1, \ldots, n_1.$$

For nominal system $\mathcal{S}$ where $\delta = 0$, we use T in replacement of T^0. In the remaining of the section, we denote by $V[i]$ the ith element of a vector V of size $|V|$ ($i \in \{1, \cdots, |V|\}$).

Based on the properties of predecessor maps given in Proposition 3.4, the operator T^δ satisfies the following properties.

Proposition 6.1. Given a matrix $\mathbf{M}$ of symbols and vectors V, W of subsets of $\mathbb{X}$ that match in dimension for operator T^δ defined in (6.8) with $\delta \geq 0$,

(i) $T^\delta(\mathbf{M}, V) \preceq T^\delta(\mathbf{M}, W)$ if $V \preceq W$,

(ii) $T^\delta(\mathbf{M}, V) + T^\delta(\mathbf{M}, W) \preceq T^\delta(\mathbf{M}, W + V)$,

(iii) $T^\delta(\mathbf{M}_1, V) + T^\delta(\mathbf{M}_2, W) \preceq T^\delta(\begin{bmatrix}\mathbf{M}_1 & \mathbf{M}_2\end{bmatrix}, \begin{bmatrix}V \\ W\end{bmatrix})$, and

(iv) $T^{\delta_2}(\mathbf{M}, W) \preceq T^{\delta_1}(\mathbf{M}, W) \preceq T(\mathbf{M}, W)$ for $0 \leq \delta_1 \leq \delta_2$.

Proof. To show (i), assume that $\mathbf{M}$ is of size $n_1 \times n_2$ and W, V of size $n_2 \times 1$. As defined in (6.6), $V \preceq W$ means $V[j] \subseteq W[j]$ and hence $(L^{-1}(m_{ij}) \cap V[j]) \subseteq (L^{-1}(m_{ij}) \cap W[j])$ for all $i, j \in \{1, \cdots, n_2\}$. By the monotonicity of Pre^δ, $\text{Pre}^\delta(\bigcup_{i=1}^{n_1} L^{-1}(m_{ij}) \cap V[j]) \subseteq \text{Pre}^\delta(\bigcup_{i=1}^{n_1} L^{-1}(m_{ij}) \cap W[j])$, which gives $T^\delta(\mathbf{M}, V) \preceq T^\delta(\mathbf{M}, W)$.

We now prove (ii). Let $i \in \{1, \ldots, n_1\}$ be arbitrary. Then by (6.8) and Proposition 3.3, we have

$$\begin{aligned}\text{Pre}^\delta\left(\bigcup_{j=1}^{n_2} L^{-1}(m_{ij}) \cap (W[j] \cup V[j])\right) &= \text{Pre}^\delta\left(\bigcup_{j=1}^{n_2} (L^{-1}(m_{ij}) \cap W[j]) \cup (L^{-1}(m_{ij}) \cap V[j])\right) \\ &\supseteq \text{Pre}^\delta\left(\bigcup_{j=1}^{n_2} L^{-1}(m_{ij}) \cap W[j]\right) \cup \text{Pre}^\delta\left(\bigcup_{j=1}^{n_2} L^{-1}(m_{ij}) \cap V[j]\right).\end{aligned}$$

Hence, $T^\delta(\mathbf{M}, V) + T^\delta(\mathbf{M}, W) \preceq T^\delta(\mathbf{M}, W + V)$.

For (iii), let $\mathbf{M} = \begin{bmatrix}\mathbf{M}_1 & \mathbf{M}_2\end{bmatrix}$ with $\mathbf{M}_1$ and $\mathbf{M}_2$ of size $n_1 \times n_2$ and $n_1 \times n_3$, respectively. The element at the ith row and $j$the column of $\mathbf{M}$ is denoted by m_{ij}, and the ith element of $T^\delta(\mathbf{M}_1, V)$ and $T^\delta(\mathbf{M}_2, W)$ are denoted by V'_i and W'_i, respectively. Then

$$\begin{aligned}V'_i \cup W'_i &= \text{Pre}^\delta\left(\bigcup_{j=1}^{n_2} L^{-1}(m_{ij}) \cap V[j]\right) \cup \text{Pre}^\delta\left(\bigcup_{j=1}^{n_3} L^{-1}(m_{i(n_2+j)}) \cap W[j]\right) \\ &\subseteq \text{Pre}^\delta\left(\left(\bigcup_{j=1}^{n_2} L^{-1}(m_{ij}) \cap V[j]\right) \cup \left(\bigcup_{j=1}^{n_3} L^{-1}(m_{i(n_2+j)}) \cap W[j]\right)\right),\end{aligned}$$

which is the ith element of $T^{\delta}(\begin{bmatrix}\mathbf{M}_1 & \mathbf{M}_2\end{bmatrix}, \begin{bmatrix}V \\ W\end{bmatrix})$. Hence, (iii) is proved.

Property (iv) is straightforward by the fact that $\text{Pre}^{\delta}(A \ominus \mathcal{B}_{\delta_2}) \subseteq \text{Pre}^{\delta}(A \ominus \mathcal{B}_{\delta_1}) \subseteq \text{Pre}^{\delta}(A)$ for all $A \subseteq \mathbb{X}$. $\square$

The graph representation of a DBA can be coded into a matrix of symbols, which is given in the following definition.

Definition 6.4. Given a DBA $\mathcal{A}_{\varphi}$ with the set of states Q, the *transition matrix* $\mathbf{M}_{\varphi}$ of $\mathcal{A}_{\varphi}$ is a $|Q|$ by $|Q|$ matrix of symbols from Σ. The element m_{ij} in the ith row and jth column ($i, j \in \{1, \cdots, |Q|\}$) of $\mathbf{M}_{\varphi}$ is given by

$$m_{ij} = \begin{cases} \sigma & Q[j] = r(Q[i], \sigma), \sigma \in \Sigma, \\ e & \text{o.w.}, \end{cases} \tag{6.9}$$

where $e \in \Sigma$ denotes an empty symbol with $L^{-1}(e) = \emptyset$.

Intuitively, if the symbol m_{ij} at ith row and jth column is σ, then the automaton state jumps from $Q[i]$ to $Q[j]$ under the input symbol σ.

Remark 6.1. The transition matrix formulation in Definition 6.4 not only applies to DBA but also to any automaton.

As defined in (6.8), the operator T^{δ} computes predecessors according to the transition relation provided in $\mathbf{M}_{\varphi}$.

To track the control values that can activate the transitions, we further define a vector $\mathcal{K} = \begin{bmatrix}\kappa_1 & \dots & \kappa_{n_1}\end{bmatrix}$ of maps (2.8), where ($i = 1, \dots, n_1$)

$$\kappa_i(x) = \Pi_{S_i}(x),\ \forall x \in W'[i], \tag{6.10}$$

where $S_i = \bigcup_{j=1}^{n_2} L^{-1}(m_{ij}) \cap W[j]$.

For a DBA $\mathcal{A}_{\varphi}$, the dependencies among the $\mathcal{S}$-domains can be captured by using the operator T^{δ} and the transition matrix $\mathbf{M}_{\varphi}$.

Proposition 6.2. Let $\mathbf{M}_{\varphi}$ be the transition matrix of a DBA $\mathcal{A}_{\varphi}$ and $\mathbf{W}_{\mathcal{S}}$ be a vector of $\mathcal{S}$-domains of all the states in $\mathcal{A}_{\varphi}$, where $\mathcal{S}$ is of the form (2.4). Then $\mathbf{W}_{\mathcal{S}} = T^{\delta}(\mathbf{M}_{\varphi}, \mathbf{W}_{\mathcal{S}})$.

Proof. Let $V = T^{\delta}(\mathbf{M}_{\varphi}, \mathbf{W}_{\mathcal{S}})$, $i \in \{1, \cdots, |Q|\}$ be arbitrary, and B be a vector with

$$B[i] = \bigcup_{j=1}^{|Q|} L^{-1}(m_{ij}) \cap \mathbf{W}_{\mathcal{S}}[j].$$

We first show that $V \preceq \mathbf{W}_{\mathcal{S}}$. Any state $x \in V[i] = \text{Pre}^{\delta}(B[i])$ can be controlled under some $u \in \mathbb{U}$ into $B[i]$ in one step under any bounded disturbance $d \in \mathbb{D}$. By Definition 6.3, for any state $x \in W_{\mathcal{S}}[j]$ (any $j \in \{1, \cdots, |Q|\}$), there exists a run ϱ with $\varrho[t] = Q[j]$ at some $t \in \mathbb{Z}^{+}$, which is generated according to (6.3), such that ϱ visits F infinitely often. Hence, $x \in W_{\mathcal{S}}(Q[i])$ by Definition 6.3, and $V[i] \subseteq W_{\mathcal{S}}(Q[i])$. Since i is arbitrary, $V \preceq \mathbf{W}_{\mathcal{S}}$.

Next we show that $\mathbf{W}_{\mathcal{S}} \preceq V$. Suppose that there is an $x \in W_{\mathcal{S}}(Q[i])$ but $x \notin V[i] = \text{Pre}^{\delta}(B[i])$. Then by Definition 3.3, for all $u \in \mathbb{U}$ there exists $d \in \mathbb{D}$ such that $x' = f(x, u) + d \notin L^{-1}(m_{ij}) \cap W_{\mathcal{S}}(Q[j])$ for all $j \in \{1, \ldots, |Q|\}$. It implies that there is no solution of $\mathcal{S}$ that passes through x and x' with its trace visiting F infinitely many times under all disturbances, and hence, $x \notin W_{\mathcal{S}}(Q[i])$, which contradicts the assumption.

Therefore, by (6.6) and (6.7), $V \preceq \mathbf{W}_{\mathcal{S}}$ and $\mathbf{W}_{\mathcal{S}} \preceq V$ gives $\mathbf{W}_{\mathcal{S}} = V$. □

Remark 6.2. Proposition 6.2 is a necessary condition for a vector W to be $\mathbf{W}_{\mathcal{S}}$, and $\mathbf{W}_{\mathcal{S}}$ may not be the unique fixed point of T^{δ} with respect to a transition matrix $\mathbf{M}_{\varphi}$.

Without loss of generality, we assume that the indices of DBA states are rearranged so that the indices of accepting states are greater than non-accepting ones. The transition matrix is also rearranged correspondingly.

Based on the transition matrix form of a DBA $\mathcal{A}_{\varphi}$, we now present a fixed-point algorithm (6.11) to characterize $\mathcal{S}$-domains of $\mathcal{A}_{\varphi}$ and show that a memoryless control strategy is sufficient to activate a transition in $\mathcal{A}_{\varphi}$.

$$n_1 = |Q| - |F|\,,\; n_2 = |F|$$
$$\mathbf{M}_{\varphi} = \begin{bmatrix} (\mathbf{M}_1)_{n_1 \times |Q|} \\ (\mathbf{M}_2)_{n_2 \times |Q|} \end{bmatrix},\; \mathcal{K} = \begin{bmatrix} \kappa_1 \\ \vdots \\ \kappa_{|Q|} \end{bmatrix},\; Z_0 = \begin{bmatrix} \mathbb{X} \\ \vdots \\ \mathbb{X} \end{bmatrix}_{n_2 \times 1}$$

$$\begin{cases} Y_\nu^0 = \begin{bmatrix} \bigcup_{j=1}^{n_2} L^{-1}(m_{1(j+n_1)}) \cap Z_\nu[j] \\ \vdots \\ \bigcup_{j=1}^{n_2} L^{-1}(m_{n_1(j+n_1)}) \cap Z_\nu[j] \end{bmatrix}_{n_1 \times 1} & \text{(6.11a)} \\ Y_\nu^{l+1} = Y_\nu^l + T^\delta\left(\mathbf{M}_1, \begin{bmatrix} Y_\nu^l \\ Z_\nu \end{bmatrix}\right) & \text{(6.11b)} \\ \kappa_i(x) \text{ by (6.10) } \forall x \in Y_\nu^{l+1}[i] \setminus Y_\nu^l[i],\ i \in \{1, \cdots, n_1\} & \text{(6.11c)} \\ Z_{\nu+1} = T^\delta\left(\mathbf{M}_2, \begin{bmatrix} Y_\nu \\ Z_\nu \end{bmatrix}\right) \quad Y_\nu \triangleq \bigcup_{l=0}^{\infty} Y_\nu^l & \text{(6.11d)} \\ \kappa_{n_1+i}(x) \text{ by (6.10) } \forall x \in Z_\nu[i],\ i \in \{1, \cdots, n_2\} & \text{(6.11e)} \end{cases}$$

The input arguments of algorithm (6.11) are the transition matrix $\mathbf{M}_\varphi$ of $\mathcal{A}_\varphi$ and an operator T^δ that reflects the transition relation of system $\mathcal{S}$. We assume that the nodes of $\mathcal{A}_\varphi$ are sorted so that accepting nodes rank after nonaccepting ones, and the transition matrix $\mathbf{M}_\varphi$ is divided into 2 matrix blocks $\mathbf{M}_1$, $\mathbf{M}_2$ which represent the transitions from nonaccepting and accepting nodes, respectively.

The major iterations of (6.11) are (6.11b) and (6.11d), in which sequences of vectors $\{Y_\nu^l\}_{l=0}^\infty$ and $\{Z_\nu\}_{\nu=0}^\infty$ are generated, respectively. Let $Z_\nu[i]$ be the ith element of the vector Z_ν. For any fixed ν, let $Y_\nu^l[i]$ denote the ith element of Y_ν^l.

The initial condition Z_0 is a vector of $\mathbb{X}$s with size n_2 and Z_ν is computed by applying operator T^δ with respect to the prior $Y_{\nu-1}$ and $Z_{\nu-1}$. The vector Y_ν (for any fixed ν) of subsets of $\mathbb{X}$ is obtained as the infinite unions of $\{Y_\nu^l\}_{l=0}^\infty$ by iteration (6.11b) and it is trivial that $Y_\nu^l \preceq Y_\nu^{l+1}$ by (6.4) for all $l \in \mathbb{N}$.

Intuitively, the sequences $\{Z_\nu\}_{\nu=0}^\infty$ and $\{Y_\nu\}_{\nu=0}^\infty$ approach the $\mathcal{S}$-domains of the accepting and nonaccepting nodes, respectively. Define a vector of subsets of $\mathbb{X}$:

$$W = \begin{bmatrix} Y \\ Z \end{bmatrix} = \bigcap_{\nu=0}^{\infty} \begin{bmatrix} Y_\nu \\ Z_\nu \end{bmatrix} = \bigcap_{\nu=0}^{\infty} W_\nu, \tag{6.12}$$

where $W_\nu = \begin{bmatrix} Y_\nu \\ Z_\nu \end{bmatrix}$. We then show as follows that Y is a vector of the $\mathcal{S}$-domains of nonaccepting nodes while Z is a vector of $\mathcal{S}$-domains of accepting nodes.

Theorem 6.1 (Conditional Soundness and Completeness)**.** Consider system $\mathcal{S}$ and a DBA $\mathcal{A}_\varphi$. Denote by $\mathbf{M}_\varphi$ the transition matrix of $\mathcal{A}_\varphi$. Let $\mathbf{W}_\mathcal{S}$ be a vector of $\mathcal{S}$-domains of $\mathcal{A}_\varphi$. Assume that

$$Y_\nu = Y_\nu + T^\delta\left(\mathbf{M}_1, W_\nu\right)\ \forall \nu \in \mathbb{N}, \tag{6.13}$$

$$Z = T^\delta\left(\mathbf{M}_2, W\right), \tag{6.14}$$

where W_ν and W are defined in (6.12). Then $W = \mathbf{W}_{\mathcal{S}}$ and each element $\mathcal{K}[i]$ of $\mathcal{K}$ is a memoryless control strategy defined on W.

Proof. We first show that both $\{Z_\nu\}$ and $\{Y_\nu\}$ are decreasing by induction. The initial condition is $Z_1 \preceq Z_0$ ($Z_0[i] = \mathbb{X}$ for $i = 1, \ldots, n_2$). Suppose that $Z_\nu \preceq Z_{\nu-1}$ and $Y_\nu \preceq Y_{\nu-1}$ for some $\nu \in \mathbb{Z}^+$. Then $Y^0_{\nu+1} \preceq Y^0_\nu$ because $Y^0_{\nu+1}[i] = \bigcup_{j=1}^{n_2} L^{-1}(m_{i(j+n_1)}) \cap Z_{\nu+1} \subseteq \bigcup_{j=1}^{n_2} L^{-1}(m_{i(j+n_1)}) \cap Z_\nu[j][j] = Y^0_\nu[i]$ for all i. According to the algorithm and the monotonicity of T^δ,

$$\begin{aligned}
&Z_{\nu+1} = T^\delta\left(\mathbf{M}_2, W_\nu\right) \preceq T^\delta\left(\mathbf{M}_2, W_{\nu-1}\right) = Z_\nu,\\
&Y^0_{\nu+1} \preceq Y^0_\nu, \quad \text{Assume } Y^l_{\nu+1} \preceq Y^l_\nu:\\
&Y^{l+1}_{\nu+1} = Y^l_\nu + T^\delta\left(\mathbf{M}_1, W_\nu\right) \preceq Y^l_{\nu-1} + T^\delta\left(\mathbf{M}_1, W_{\nu-1}\right) = Y^{l+1}_\nu.
\end{aligned}$$

Hence, $Y_{\nu+1} = \bigcup_{l=0}^\infty Y^l_{\nu+1} \preceq \bigcup_{l=0}^\infty Y^l_\nu = Y_\nu$, and we can conclude that $\{Z_\nu\}$ and $\{Y_\nu\}$ are decreasing.

We next claim that $Y_\nu[i] = \bigcup_{l=0}^\infty Y^l_\nu[i] = \mathcal{BR}^\delta_\infty(Y^0_\nu[i])$ for all $i = 1, \cdots, n_1$ and $\nu \in \mathbb{N}$. Expanding the operator T^δ, we have

$$\begin{aligned}
Y^0_\nu[i] &= \bigcup_{j=1}^{n_2} L^{-1}(m_{i(j+n_1)}) \cap Z_\nu[j],\\
Y^{l+1}_\nu[i] &= Y^l_\nu[i] \cup \mathrm{Pre}^\delta\left(\left(\bigcup_{j=1}^{n_1} L^{-1}(m_{ij}) \cap Y^l_\nu[j]\right) \cup \left(\bigcup_{j=1}^{n_2} L^{-1}(m_{i(j+n_1)}) \cap Z_\nu[j]\right)\right).
\end{aligned}$$

Based on Proposition 4.4 in Chapter 4, we have $Y^l_\nu[i] = \mathcal{BR}^\delta_l(Y^0_\nu[i])$, which is the l-step δ-robustly reachable set to any of the $\mathcal{S}$-domains of the accepting nodes, i.e., $\bigcup_{j=1}^{n_2} L^{-1}(m_{i(j+n_1)}) \cap Z_\nu[j]$. Hence, $Y_\nu[i] = \bigcup_{l=0}^\infty Y^l_\nu[i] \subseteq \mathcal{BR}^\delta_\infty(Y^0_\nu[i])$ for all $i = 1, \cdots, n_1$. Under condition (6.13), we can show the other direction, i.e., $Y^l_\nu[i] \supseteq \mathcal{BR}^\delta_\infty(Y^0_\nu[i])$. Suppose that $q_0, q' \in Q$ in the associated DBA $\mathcal{A}_\varphi$ correspond to the kth and k'th row of $\mathbf{M}_\varphi$, respectively, and $q' = r(q_0, L(x_0))$. For any $x_0 \notin Y_\nu[k]$, we have by (6.13)

$$x_0 \notin \mathrm{Pre}^\delta\left(\left(\bigcup_{j=1}^{n_1} L^{-1}(m_{kj}) \cap Y_\nu[j]\right) \cup \left(\bigcup_{j=1}^{n_2} L^{-1}(m_{k(j+n_1)}) \cap Z_\nu[j]\right)\right),$$

which means that for all $u_0 \in \mathbb{U}$ there exists $d_0 \in \mathbb{D}$ such that $x_1 = f(x_0, u_0) + d_0 \notin Y_\nu[k_1]$ or $Z_\nu[k_2]$ for all $k_1 = 1, \cdots, n_1$ and $k_2 = 1, \cdots, n_2$. Since $x_1 \notin Y_\nu[k']$ (the current DBA state is q_i), we can use the same argument as for x_0, i.e., for all $u_1 \in \mathbb{U}$ there exists $d_1 \in \mathbb{D}$ such that $x_2 = f(x_1, u_1) + d_1 \notin Y_\nu[k_1]$ or $Z_\nu[k_2]$ for all $k_1 = 1, \cdots, n_1$ and $k_2 = 1, \cdots, n_2$. In this way, we can construct a sequence of disturbances $\{d_t\}_{t=0}^{\infty}$ such that $x_t \notin \bigcup_{j=1}^{n_2} L^{-1}(m_{i(j+n_1)}) \cap Z_\nu[j]$ for all $t \in \mathbb{N}$ and $i = 1, \cdots, |Q|$. Hence, $x \notin \mathcal{BR}_\infty^\delta(Y_\nu^0[i])$ and the claim is proved.

We now prove the theorem by showing both $\mathbf{W}_\mathcal{S} \preceq W$ and $W \preceq \mathbf{W}_\mathcal{S}$.

To see $\mathbf{W}_\mathcal{S} \preceq W$, we only need to prove that $\mathbf{W}_\mathcal{S} \preceq W_\nu$ for an arbitrary $\nu \in \mathbb{N}$. As the initial condition, $Z_0[i] = \mathbb{X}$, and thus $\mathbf{W}_\mathcal{S}[n_1 + i] \preceq Z_0[i]$ for all $i = 1, \cdots, n_2$. As we have shown in the claim above, $Y_\nu[i]$ are the maximal δ-robustly backward reachable set to $\bigcup_{j=1}^{n_2} Z_\nu[j]$. And by Definition 6.3, for all $i = 1, \cdots, n_1$, $\mathbf{W}_\mathcal{S}[i] \subseteq \mathcal{BR}_\infty^\delta(\bigcup_{j=1+n_1}^{|Q|} \mathbf{W}_\mathcal{S}[j])$, we have $\mathbf{W}_\mathcal{S}[i] \subseteq \mathcal{BR}_\infty^\delta(\bigcup_{j=1+n_1}^{|Q|} Z_0[j]) = Y_0[i]$. Therefore, $\mathbf{W}_\mathcal{S} \preceq W_0$. Assume that $\mathbf{W}_\mathcal{S} \preceq W_\nu$ for some $\nu \in \mathbb{N}$. By Proposition 6.2 and 6.1 (i), we have $\mathbf{W}_\mathcal{S} = T^\delta(\mathbf{M}, \mathbf{W}_\mathcal{S}) \preceq T^\delta(\mathbf{M}, W_\nu)$. Then $\mathbf{W}_\mathcal{S}[n_1 + 1, \cdots, |Q|] \preceq T^\delta(\mathbf{M}_2, W_\nu) = Z_{\nu+1}$. Since the above derivation of $\mathbf{W}_\mathcal{S}[i] \subseteq \mathcal{BR}_\infty^\delta(\bigcup_{j=1+n_1}^{|Q|} Z_0[j]) = Y_0[i]$ holds for $\nu + 1$, we have $\mathbf{W}_\mathcal{S}[1, \cdots, n_1] \preceq Y_{\nu+1}$, and hence $\mathbf{W}_\mathcal{S} \preceq W_{\nu+1}$, which shows that $\mathbf{W}_\mathcal{S} \preceq W$.

To show $W \preceq \mathbf{W}_\mathcal{S}$, we aim to prove that $W[i] \subseteq \mathbf{W}_\mathcal{S}[i]$ for any $i = 1, \ldots, |Q|$. Suppose that the current DBA state is q_{i+n_1-1} ($i \in \{1, \cdots, n_2\}$) with which $Z[i]$ is associated. Given (6.14), i.e.,

$$Z[i] = \mathrm{Pre}^\delta\left(\left(\bigcup_{j=1}^{n_1} L^{-1}(m_{(i+n_1)j}) \cap Y[j]\right) \cup \left(\bigcup_{j=1}^{n_2} L^{-1}(m_{(i+n_1)(j+n_1)}) \cap Z[j]\right)\right)$$

for all $i = 1, \cdots, n_2$, for any state $x_0 \in Z[i]$ there exists $u_0 \in \mathbb{U}$ such that $x_1 = f(x_0, u_0) + d_0 \in Z[j]$ if $L(x_0) = m_{(i+n_1)(j+n_1)}$ or $x_1 \in Y[k]$ if $L(x_0) = m_{(i+n_1)k}$ under any possible disturbance $d_0 \in \mathbb{D}$. If $x_1 \in Z[j]$, then x_2 at $t = 2$ can be still kept inside $Z[j]$ or $Y[k]$. If $x_1 \in Y[k]$, then for all sequences of disturbances $\{d_t\}_{t=1}^{\infty}$ with $d_t \in \mathbb{D}$ there exists $t' \in \mathbb{Z}^+$ and a control signal $\{u_t\}_{t=1}^{t'}$ such that $x_{t'} \in Z[j]$ for some $j \in \{1, \cdots, n_2\}$, since $Y[k]$ is the maximal δ-robustly backward reachable set to $\bigcup_{j=1}^{n_2} Z[j]$ by (6.13). In this sense, for any sequence of disturbances $\{d_t\}_{t=0}^{\infty}$, we can always find a control signal $\{u_t\}_{t=0}^{\infty}$ such that the run of the automaton $\mathcal{A}_\varphi$ under the trace of resulting solution of system $\mathcal{S}$ with initial condition $x_0 \in \bigcup_{i=1}^{|Q|} W[i]$ satisfies the Büchi accepting condition. Therefore, $W \preceq \mathbf{W}_\mathcal{S}$ and the proof is complete. □

Theorem 6.1 essentially says that LTL control synthesis for general dynamical system can be sound and complete under conditions (6.13) and (6.14). It also implies that we only need finite memories to realize control synthesis with respect to the LTL formulas that can be translated into DBA. At each state of an LTL equivalent DBA $\mathcal{A}_\varphi$, a memoryless control strategy is sufficient to maintain the state of system $\mathcal{S}$ inside $\mathcal{S}$-domains of $\mathcal{A}_\varphi$, which by definition are subsets of the state space $\mathbb{X}$ that system $\mathcal{S}$ can be controlled to satisfy the Büchi accepting condition (6.1). The current DBA state needs to be recorded in a variable so that a proper memoryless control strategy κ from $\mathcal{K}$ can be chosen. Such a variable is considered as the memory of the control strategy and is updated according to the transitions of $\mathcal{A}_\varphi$. Detailed control strategy structure will be discussed in the next section.

6.2.3 Automata-Embedded Control Structure

As a result of Theorem 6.1, we can design a finite-memory control strategy that is embedded with the given DBA.

Definition 6.5. Let $\mathcal{A}_\varphi = (Q, \Sigma, r, q_0, F)$ be the equivalent DBA of an LTL formula φ. For system $\mathcal{S} = \langle \mathbb{X}, \mathbb{U}, \mathbb{D}, R, AP, L \rangle$, an *automaton-embedded control strategy* is defined as

$$\mathcal{C}_\varphi = \langle \mathbb{X}_c, \mathbb{U}_c, Q_c, \Sigma_c, r_c, q_0, H \rangle :$$

- $\mathbb{X}_c \subseteq \mathbb{X}$ is a set of inputs;
- $Q_c = Q$ is a finite set of states;
- $\Sigma_c = \Sigma = 2^{AP}$ is an alphabet;
- $r_c = r \subseteq Q_c \times \Sigma_c \times Q_c$ is a transition relation that updates the controller state;
- q_0 is the initial state;
- $\mathbb{U}_c \subseteq 2^{\mathbb{U}}$ is a set of outputs;
- $H : Q_c \times \mathbb{X}_c \to \mathbb{U}_c$ is an output function defined by

$$H(q, x) = \kappa_{Id(q)+1}(x),\ x \in \mathbb{X}_c, q \in Q_c,$$

  where $\kappa_{Id(q)}(x)$ belongs to the set of memoryless control strategies $\{\kappa_i\}_{i=1}^{|Q_c|}$ returned by (6.11) and $Id(q)$ is the index of the state q. The index $Id(q)$ determines which memoryless control mapping to be activated.

The components Q_c, Σ_c, r_c, q_0 originally given in $\mathcal{A}_\varphi$ are embedded into $\mathcal{C}_\varphi$. One can use a single variable that takes values in a subset of $\mathbb{N}$ to represent Q. Such a variable is called a *memory variable*. A memoryless control strategy κ from $\mathcal{K}$ is activated by the function H, which outputs the index of current state q of $\mathcal{A}_\varphi$ by the transition relation r of $\mathcal{A}_\varphi$ according to the previous automaton state and the labels $L(x)$ of the current system state x. Therefore, the embedded $\mathcal{A}_\varphi$ manages the control memory, and the structure in Definition 6.5 is visualized in Figure 6.6.

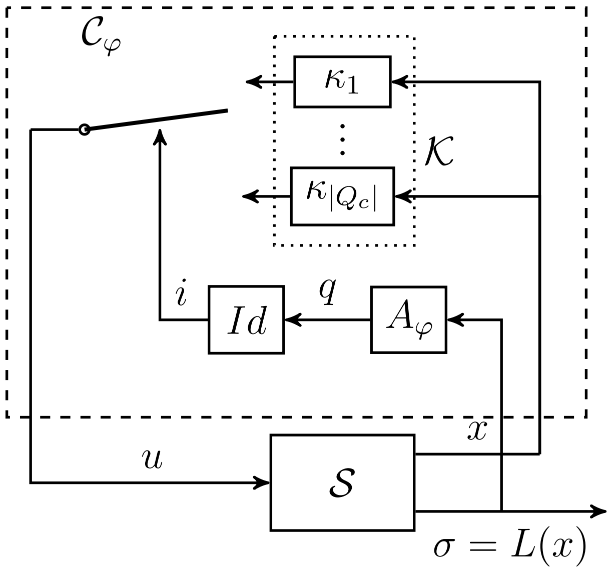


Figure 6.6: The structure of the automaton-embedded control strategy for $\mathcal{S}$ with respect to φ.

Corollary 6.1. Let $\mathcal{A}_\varphi$ be the equivalent DBA for an LTL specification φ for system $\mathcal{S}$. If φ is realizable for $\mathcal{S}$, then a finite-memory control strategy in Definition 6.5 can realize φ.

Proof. It is a direct result of Theorem 6.1. □

6.3 Robust Completeness of LTL Control Synthesis

The operator T defined in (6.8) is essentially a predecessor map of unions of sets, which is difficult to be exactly computed for nonlinear systems. Same as the control synthesis for basic LTL formulas such as invariance and reachability, we can also inner approximate $W' = T(\mathbf{M}, W)$

by using Algorithm 5.1, denoted by $W'^{\varepsilon} = [\underline{T}]^{\varepsilon}(\mathbf{M}, W)$, with

$$W'^{\varepsilon}[i] = [\underline{\text{Pre}}]^{\varepsilon}\left(\bigcup_{j=1}^{n_2} L^{-1}(m_{i,j}) \cap W[j] \middle| \mathbb{X}\right), \text{ or} \tag{6.15}$$

$$W'^{\varepsilon}[i] = [\underline{\text{Pre}_{\mu}}]^{\varepsilon}\left(\bigcup_{j=1}^{n_2} L^{-1}(m_{i,j}) \cap W[j] \middle| \mathbb{X}\right) \tag{6.16}$$

where $[\underline{\text{Pre}}]^{\delta}$ is given in (5.2) for system $\mathcal{S}$ with finite control values, and $[\underline{\text{Pre}_{\mu}}]^{\varepsilon}$ is the one for system $[\mathcal{S}]_{\mu}$ defined in (5.9).

Hence, the sequences $\{Y_{\nu}\}$ and $\{Z_{\nu}\}$ in algorithm (6.11) can be inner approximated by using $[\underline{T}]^{\varepsilon}$, which gives Algorithm 6.1 as the approximated version of algorithm (6.11).

In the following, we focus on the approximation (6.16) since (6.15) can be seen as a special case of (6.16).

Lemma 6.1. Consider system $[\mathcal{S}]_{\mu}$ where μ is a parameter given in (5.8). Let $\mathbf{M}$ be a matrix of symbols from Σ and W be a vector of subsets of $\mathbb{X}$, and $\mathbf{M}$ and W match in dimension. If Assumption 5.1 and 5.2 hold in $\mathbb{X}$, then

$$T^{(\rho_1\varepsilon+\rho_2\mu)}(\mathbf{M}, W) \preceq [\underline{T}]^{\varepsilon}(\mathbf{M}, W) \preceq T(\mathbf{M}, W). \tag{6.17}$$

Proof. Let $V = T(\mathbf{M}, W)$, $V' = [\underline{T}]^{\varepsilon}(\mathbf{M}, W)$ and $V'' = T^{(\rho_1\varepsilon+\rho_2\mu)}(\mathbf{M}, W)$. Assume that $\mathbf{M}$ is of size $n_1 \times n_2$ and W is of size $n_2 \times 1$. Then V, V' and V'' are $n_1 \times 1$.

Consider an arbitrary element V_i of V, $i = 1, \cdots, n_1$. Lemma 5.2 gives

$$\begin{aligned}
\text{Pre}^{(\rho_1\varepsilon+\rho_2\mu)}\left(\bigcup_{j=1}^{n_2} L^{-1}(m_{ij}) \cap W[j]\right) &= \text{Pre}\left(\bigcup_{j=1}^{n_2} \left(L^{-1}(m_{ij}) \cap W[j]\right) \ominus \mathcal{B}_{\rho_1\varepsilon+\rho_2\mu}\right) \\
&\subseteq [\underline{\text{Pre}_{\mu}}]^{\varepsilon}\left(\bigcup_{j=1}^{n_2} L^{-1}(m_{ij}) \cap W[j]\right) \\
&\subseteq \text{Pre}\left(\bigcup_{j=1}^{n_2} L^{-1}(m_{ij}) \cap W[j]\right).
\end{aligned}$$

Then by (6.6), we have $V_i'' \subseteq V_i' \subseteq V_i$, which shows that $V'' \preceq V' \preceq V$. □

Algorithm 6.1 $W^\varepsilon, \mathcal{K}^\varepsilon = \textsc{Sdom}(\mathbf{M}_\varphi, [\underline{T}]^\varepsilon)$

```
1: $n_1 = |Q| - |F|$, $n_2 = |F|$
2: $\mathbf{M}_\varphi = \begin{bmatrix}\mathbf{M}_1\\ \mathbf{M}_2\end{bmatrix}$, $\mathbf{M}_1, \mathbf{M}_2$ are of $n_1 \times (n_1+n_2)$, $n_2 \times (n_1+n_2)$, respectively.
3: $\widetilde{Z}[1,\cdots,n_2]$, $\widetilde{Z}[i] \leftarrow \mathbb{X}$, $i = 1,\ldots,n_2$
4: $Z[1,\cdots,n_2]$, $Z[i] \leftarrow \emptyset$
5: $\widetilde{Y}[1,\cdots,n_1]$, $Y[1,\cdots,n_1]$
6: $\mathcal{K}[1,\ldots,|Q|]$ is a vector of memoryless control strategies.                ▷ (2.8)
7: while $Z \neq \widetilde{Z}$ do                                                           ▷ (6.7)
8:     $Z \leftarrow \widetilde{Z}$
9:     $\widetilde{Y}[i] \leftarrow \bigcup_{j=1}^{n_2} L^{-1}(m_{i(j+n_1)}) \cap Z[j]$, $Y[i] \leftarrow \emptyset$, $i = 1,\ldots,n_1$
10:    while $Y \neq \widetilde{Y}$ do
11:        $Y \leftarrow \widetilde{Y}$
12:        $\widetilde{Y} \leftarrow Y + [\underline{T}]^\varepsilon(\mathbf{M}_1, \begin{bmatrix}Y\\ Z\end{bmatrix})$              ▷ (6.4)
13:        assign $\mathcal{K}^\varepsilon[i](x)$ by (6.10) for all $x \in \widetilde{Y}[i] \setminus Y[i]$ and $i \in \{1,\cdots,n_1\}$
14:    end while
15:    $\widetilde{Z} \leftarrow [\underline{T}]^\varepsilon(\mathbf{M}_2, \begin{bmatrix}Y\\ Z\end{bmatrix})$                      ▷ (6.4)
16:    assign $\mathcal{K}^\varepsilon[n_1+i](x)$ by (6.10) for all $x \in \widetilde{Z}[i]$ and $i \in \{1,\cdots,n_2\}$
17: end while
18: $W^\varepsilon \leftarrow \begin{bmatrix}Y\\ Z\end{bmatrix}$
```

Based on Lemma 6.1, we can show that by using Algorithm 6.1 control synthesis for system $\mathcal{S}$ with respect to DBA-recognizable LTL formulas can be made sound and robustly complete in the sense that a finite-memory control strategy defined in Definition 6.5 can be constructed whenever φ is realizable for system $\mathcal{S}$ under additional δ bounded disturbances.

Theorem 6.2 (Conditional Soundness and Robust Completeness)**.** Consider system $\mathcal{S}$ and a DBA $\mathcal{A}_\varphi$. Denote by $\mathbf{M}_\varphi$ the transition matrix of $\mathcal{A}_\varphi$. Let $W^\varepsilon, \mathcal{K}^\varepsilon = \textsc{Sdom}(\mathbf{M}_\varphi, [\underline{T}]^\varepsilon)$, where $[\underline{T}]^\varepsilon$ is an interval approximation of T defined in (6.16), and $W^\varepsilon(q_0)$ be the element of W^ε corresponding to the initial state q_0 of $\mathcal{A}_\varepsilon$. Suppose that Assumption 5.1 and 5.2 hold in $\mathbb{X}$. Then Algorithm 6.1 terminates in a finite number of iterations, and if $\rho_1\varepsilon + \rho_2\mu \leq \delta$, then

$$\text{Win}^\delta_{\mathcal{S}}(\varphi) \subseteq W^\varepsilon(q_0) \subseteq \text{Win}_{\mathcal{S}}(\varphi). \tag{6.18}$$

Proof. We first show the finite termination. As we have a lower bound, which is determined by ε, for the width of all the intervals that partition the state space $\mathbb{X}$, each element in vectors X and Y contains finitely many intervals. Algorithm 6.1 will terminate in a finite number of steps.

Based on Proposition 6.1 (iv), we have $T^\delta(\mathbf{M}, W) \preceq T^{\rho_1\varepsilon+\rho_2\mu)}(\mathbf{M}, W)$ for all proper $\mathbf{M}$ and W if $\rho(\varepsilon+\mu) \leq \delta$. Together with Lemma 6.1, we have $T^\delta(\mathbf{M}, W) \preceq [\underline{T}]^\varepsilon(\mathbf{M}, W) \preceq T(\mathbf{M}, W)$. Let $\{[Z]^\varepsilon_i\}$ and $\{[Y]^\varepsilon_i\}$ be the monotone sequences by using $[\underline{T}]^\varepsilon$ as the input of the procedure $\textsc{Sdom}$. The same sequences for the system with disturbances of magnitude δ are denoted by $\{Z^\delta_i\}$ and $\{Y^\delta_i\}$. Then the relationship $Z^\delta_i \preceq [Z]^\varepsilon_i \preceq Z_i$ and $Y^\delta_i \preceq [Y]^\varepsilon_i \preceq Y_i$ are maintained because of their monotonicity. Therefore, $W^\delta \preceq W^\varepsilon \preceq W$ which implies $\text{Win}^\delta_{\mathcal{S}}(\varphi) \subseteq W^\varepsilon(q_0) \subseteq \text{Win}_{\mathcal{S}}(\varphi)$. □

Remark 6.3. In Chapters 4 and 5, we have given sound and complete control synthesis algorithm (4.5) and (4.14) for invariance and reachability control objectives, respectively, and the robust completeness is guaranteed by using interval approximations $[\underline{\text{Pre}}]^\varepsilon$ or $[\underline{\text{Pre}}_\mu]^\varepsilon$ of Pre. As a matter of fact, invariance and reachability formulas are two of the simplest ones that can be translated into DBA (see Figure 6.3), and the associated transition matrices are

$$\mathbf{M}_{\varphi_s} = \begin{bmatrix} \top & e \\ \neg G & G \end{bmatrix}, \quad \mathbf{M}_{\varphi_r} = \begin{bmatrix} \neg G & G \\ e & \top \end{bmatrix}. \tag{6.19}$$

Hence, they are special cases of Algorithm 6.1, in which two nested while loops reduce to a single while loop: for invariance control, the inner loop (line 10-14) can be omitted, and Algorithm 6.1 reduces to the iteration

$$Z_{\nu+1} = T^\delta\left(\begin{bmatrix} \neg G & G \end{bmatrix}, \begin{bmatrix} \emptyset \\ Z_\nu \end{bmatrix}\right),$$

which matches (4.5); for reachability control, only the inner loop takes effect, and Algorithm 6.1 reduces to the iteration

$$Y_0^{l+1} = Y_0^l + T^\delta \left(\begin{bmatrix} \neg G & G \end{bmatrix}, \begin{bmatrix} Y_0^l \\ \mathbb{X} \end{bmatrix} \right),$$

which matches (4.14) as expected.

However, control synthesis with respect to a reach-and-stay LTL formula $\varphi_{\mathrm{rs}} = \Diamond\Box G$ can not be generalized by Algorithm 6.1 as φ_{rs} is not DBA-recognizable.

6.4 Control Synthesis with Pre-processing

Another problem about LTL control synthesis that we are concerned with is its computational complexity. In Algorithm 6.1, the vector Z, which approaches the vector of $\mathcal{S}$-domains of accepting nodes in $\mathcal{A}_\varphi$ as the iteration proceeds, updates only after the vector Y (the vector of subsets of $\mathbb{X}$ that approximates the $\mathcal{S}$-domains of the rest of the nodes in $\mathcal{A}_\varphi$) remains unchanged in the inner loop. In addition, at the beginning of each computation in the outer loop, the value of Y needs to be reinitialized since the value of Z is changed from the last iteration. In this sense, the use of nested loops for the computation of interdependent Z and Y increases the computational complexity.

Analyzing the transition matrices $\mathbf{M}_{\varphi_{\mathrm{s}}}$ and $\mathbf{M}_{\varphi_{\mathrm{r}}}$ in (6.19), it is not hard to notice their lower and upper triangular structures, which indicates that the dependency between the $\mathcal{S}$-domains of accepting and nonaccepting nodes is only in one direction: $\mathbf{W}_{\mathcal{S}}(q_1)$ is not dependent on $\mathbf{W}_{\mathcal{S}}(q_0)$ in both Figures 6.3a and 6.3b. This is what breaks the nested loops in invariance and reachability control synthesis, and we can get inspiration from this two special cases for reducing the complexity of Algorithm 6.1.

Suppose that the transition matrix $\mathbf{M}_\varphi$ of a DBA $\mathcal{A}_\varphi$ is an upper triangular block matrix based on the indexed set of states $Q = \left\{q_1, \cdots, q_{|Q|}\right\}$, i.e.,

$$\mathbf{M}_\varphi = \begin{bmatrix} \mathbf{M}_{UL} & \mathbf{M}_{UR} \\ e & \mathbf{M}_{LR} \end{bmatrix}_{|Q|\times|Q|}, \tag{6.20}$$

where $\mathbf{M}_{UL}$ and $\mathbf{M}_{LR}$ are n_L by n_L and n_R by n_R matrices, respectively, and $n_L + n_R = |Q|$, $n_L, n_R \in \mathbb{Z}^+$. Let Q_L be the set of states of $\mathcal{A}_\varphi$ with the first n_L indices and Q_R be the set of the rest of the states and

$$F_L = \{q \in Q : q \in F \wedge q \in Q_L\}, \quad F_R = \{q \in Q : q \in F \wedge q \in Q_R\}. \tag{6.21}$$

Denote $n_{L2} = |F_L|$, $n_{R2} = |F_R|$, $n_{L1} = n_L - n_{L2}$, and $n_{R1} = n_R - n_{R2}$. We also assume that the states in Q_L and Q_R are sorted so that the accepting states always rank after nonaccepting ones.

Let W_L and W_R be vectors of subsets of the state space $\mathbb{X}$ of length n_L and n_R, respectively. Then it is straightforward that

$$\begin{aligned}\widetilde{W}_R &= T^{\delta}\left(\begin{bmatrix} e & \mathbf{M}_{LR}\end{bmatrix}, \begin{bmatrix} W_L \\ W_R \end{bmatrix}\right) = T^{\delta}\left(\mathbf{M}_{LR}, W_R\right), \\ \widetilde{W}_L &= T^{\delta}\left(\begin{bmatrix} \mathbf{M}_{UL} & \mathbf{M}_{UR}\end{bmatrix}, \begin{bmatrix} W_L \\ W_R \end{bmatrix}\right),\end{aligned}$$

which shows that $\widetilde{W}_R$ does not rely on W_L, but $\widetilde{W}_L$ relies on both W_L and W_R.

If Q_R contains accepting nodes, then the block matrix $\mathbf{M}_{LR}$ can be treated as a sub-transition matrix based on which $\mathcal{S}$-domains of the corresponding states in Q_R, i.e., $\{\mathbf{W}_{\mathcal{S}}(q)\}_{q\in Q_R}$, can be approximated firstly by Algorithm 6.1, independent of other parts of $\mathcal{A}_{\varphi}$. On the other hand, if Q_R has no accepting nodes, then computing $\{\mathbf{W}_{\mathcal{S}}(q)\}_{q\in Q_R}$ is pointless because there is no transition from any $q \in Q_R$ to $q' \in Q_L$ and any run that contains q does not satisfy the Büchi accepting condition. The approximation of $\mathcal{S}$-domains of the states in Q_L, i.e., $\{\mathbf{W}_{\mathcal{S}}(q)\}_{q\in Q_L}$, starts after the computation with respect to $\mathbf{M}_{LR}$ completes. In this way, the repetitive initialization and computation of W_L caused by the updates in W_R when using the operator T^{δ} can be avoided. We can also use Algorithm 6.1 for the approximation of $\{\mathbf{W}_{\mathcal{S}}(q)\}_{q\in Q_L}$ with a slight modification, which is presented as Algorithm 6.2.

Therefore, if we can arrange the transition matrix $\mathbf{M}_{\varphi}$ into a triangular matrix or triangular block matrix without changing the original transition relations in $\mathcal{A}_{\varphi}$, then Algorithm 6.1 can reduce to a single loop or several smaller nested loops. We now compare the complexities of control synthesis with respect to an upper triangular block matrix in the form of (6.20) and a general transition matrix by Algorithm 6.1 or 6.2.

Suppose that the numbers of accepting and nonaccepting nodes in $\mathcal{A}_{\varphi}$ are n_2 and $n_1 = |Q| - n_2$, respectively, and the resulting numbers of outer-loop and inner-loop iterations by using Algorithm 6.1 directly are K_2 and K_1. Then the complexity is

$$\mathcal{O}(n_2 K_2 n_1 K_1)$$

for the control synthesis without using its triangular form. Let the numbers of outer and inner-loop iterations for block $\mathbf{M}_{LR}$ be K_{R2} and K_{R1}, respectively, and the ones for block $\begin{bmatrix} \mathbf{M}_{UL} & \mathbf{M}_{UR}\end{bmatrix}$ be K_{L2} and K_{L1}, respectively. If we perform control synthesis sequentially to

Algorithm 6.2 $W_L^\varepsilon, \mathcal{K}_L^\varepsilon = \textsc{SdomExtra}([\mathbf{M}_{UL} \quad \mathbf{M}_{UR}], [\underline{T}]^\varepsilon, W_R)$

```
1: $\begin{bmatrix}\mathbf{M}_{UL} & \mathbf{M}_{UR}\end{bmatrix} = \begin{bmatrix}\mathbf{M}_{UL1} & \mathbf{M}_{UR1} \\ \mathbf{M}_{UL2} & \mathbf{M}_{UR2}\end{bmatrix}$, where $\mathbf{M}_{UL2}$ corresponds to the $F_L$ defined in (6.21)
   and has $n_{L2}$ number of rows. The block $\mathbf{M}_{UL1}$ contains $n_{L1}$ rows. The elements of $\mathbf{M}_{UL}$
   and $\mathbf{M}_{UR}$ are denoted by $m_{ij}^L$ $(i, j = 1, \ldots, n_L)$ and $m_{ij}^R$ $(i = 1, \ldots, n_L,\ j = 1, \ldots, n_R)$,
   respectively.
2: $\widetilde{Z}[1, \cdots, n_{L2}]$, $\widetilde{Z}[i] \leftarrow \mathbb{X}$, $i = 1, \ldots, n_{L2}$
3: $Z[1, \cdots, n_{L2}]$, $Z[i] \leftarrow \emptyset$
4: $\widetilde{Y}[1, \cdots, n_{L1}]$, $Y_L[1, \cdots, n_{L1}]$
5: $\mathcal{K}_L[1, \ldots, n_{L1} + n_{L2}]$ is a vector of memoryless control strategies.      ▷ (2.8)
6: while $Z \neq \widetilde{Z}$ do                                                                ▷ (6.7)
7:     $Z \leftarrow \widetilde{Z}$
8:     $Y[i] \leftarrow \emptyset$, $i = 1, \ldots, n_{L1}$
9:     $\widetilde{Y}[i] \leftarrow \left(\bigcup_{j=1}^{n_{L2}} L^{-1}(m_{i(j+n_{L1})}^L) \cap Z[j]\right) \cup \left(\bigcup_{j=1}^{n_R} L^{-1}(m_{ij}^R) \cap W_R[j]\right)$, $i = 1, \ldots, n_{L1}$
10:    while $Y \neq \widetilde{Y}$ do
11:        $Y \leftarrow \widetilde{Y}$
12:        $\widetilde{Y} \leftarrow Y + [\underline{T}]^\varepsilon \left(\begin{bmatrix}\mathbf{M}_{UL1} & \mathbf{M}_{UR1}\end{bmatrix}, \begin{bmatrix}Y \\ Z \\ W_R\end{bmatrix}\right)$      ▷ (6.4)
13:        assign $\mathcal{K}_L^\varepsilon[i](x)$ by (6.10) for all $x \in \widetilde{Y}[i] \setminus Y[i]$ and $i \in \{1, \cdots, n_{L1}\}$
14:    end while
15:    $\widetilde{Z} \leftarrow [\underline{T}]^\varepsilon \left(\begin{bmatrix}\mathbf{M}_{UL2} & \mathbf{M}_{UR2}\end{bmatrix}, \begin{bmatrix}Y \\ Z \\ W_R\end{bmatrix}\right)$      ▷ (6.4)
16:    assign $\mathcal{K}_L^\varepsilon[n_{L1} + i](x)$ by (6.10) for all $x \in \widetilde{Z}[i]$ and $i \in \{1, \cdots, n_{L2}\}$
17: end while
18: $W_L^\varepsilon \leftarrow \begin{bmatrix}Y \\ Z\end{bmatrix}$
```

blocks $\mathbf{M}_{LR}$ and $[\mathbf{M}_{UL} \quad \mathbf{M}_{UR}]$, the complexity is

$$\mathcal{O}(n_{R2}K_{R2}n_{R1}K_{R1} + n_{L2}K_{L2}n_{L1}K_{L1}).$$

The numbers of outer and inner-loop iterations are determined by the row that converges the slowest, and hence we have $K_2 = \max\{K_{R2}, K_{L2}\}$ and $K_1 = \max\{K_{R1}, K_{L1}\}$. As defined in (6.21), $n_2 = n_{L2} + n_{R2}$ and $n_1 = n_{L1} + n_{R1}$. Then

$$\begin{aligned} n_{R2}K_{R2}n_{R1}K_{R1} + n_{L2}K_{L2}n_{L1}K_{L1} &\leq (n_{R2}n_{R1} + n_{L2}n_{L1})K_2K_1 \\ &< (n_{R2} + n_{L2})(n_{R1} + n_{L1})K_2K_1 = n_2n_1K_2K_1, \end{aligned}$$

which shows that we can gain computational efficiency by using an upper triangular block matrix.

So the question is how to pre-process $\mathbf{M}_\varphi$ so that $\mathbf{M}_\varphi$ is a triangular block matrix. We now propose the following procedure, called Preprocess, for this purpose:

i) Detect all SCCs in the graph representation of $\mathcal{A}_\varphi$. Then $\mathcal{A}_\varphi$ can be simplified to a DAG $\mathcal{G}_{dag} = (V, E)$ in which each node is either a single automaton state or an SCC. Hence, the number of nodes $|V|$ of $\mathcal{G}_{dag}$ is less than or equal to $|Q|$.

ii) Perform a *topological sort* on the DAG $\mathcal{G}_{dag}$, which determines a linear ordering of the nodes in $\mathcal{G}_{dag}$ such that q precedes q' for any edge $(q, \sigma, q') \in E$. Rather than being interdependent, computation of $\mathbf{W}_\mathcal{S}(q)$ only needs to be performed after $\mathbf{W}_\mathcal{S}(q')$ is obtained for any state q that comes before q'.

iii) Let $q_{i_1} \ldots q_{i_k} \ldots q_{i_l}$ $(0 < l \leq |Q|)$ be the resulting topological sort, where q_{i_k} is the last node of $\mathcal{G}_{dag}$ that is or contains an accepting state in $\mathcal{A}_\varphi$. List the states in Q in the order of $q_{i_1} \ldots q_{i_k}$, and no particular order of the states in the same SCC is required except that the accepting states rank after the nonaccepting ones.

The transition matrix $\mathbf{M}_\varphi$ based on the order of the automaton states obtained by Preprocess can be formulated as an upper triangular block matrix. Control synthesis, as a result, can be performed independently for the sub-matrices in the reversed order.

Example 6.3. Consider the LTL formula

$$\varphi = \Diamond(a_1 \wedge \Diamond(a_2 \wedge \Diamond(a_3 \wedge (\neg a_2)\mathbf{U}a_1))), \tag{6.22}$$

whose equivalent DBA is shown in Figure 6.7.

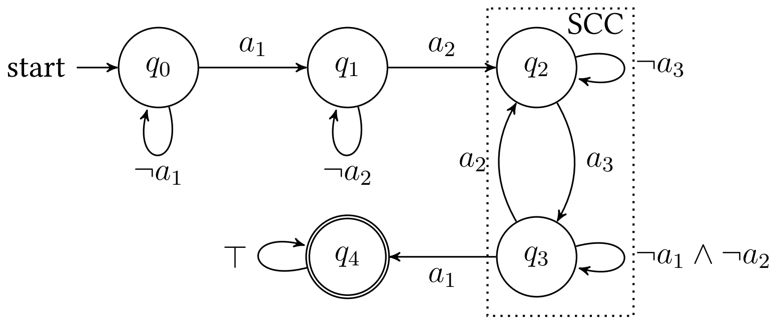


Figure 6.7: The translated DBA using Spot [37].

The states q_2 and q_3 constitute an SCC q_{23}, and the rest of the states are trivial SCCs. A topological sort of $\mathcal{A}_\varphi$ is $q_0q_1q_{23}q_4$, and q_4 is the unique accepting state. Then the Preprocess yields an order of the states in Q: $q_0q_1q_2q_3q_4$ (or $q_0q_1q_2q_3q_4$, because the order between q_2 and q_3 does not matter). Based on this order, the transition matrix is

$$\mathbf{M}_\varphi = \left[\begin{array}{c:c:cc:c} \neg a_1 & a_1 & e & e & e \\ \hdashline e & \neg a_2 & a_2 & e & e \\ \hdashline e & e & \neg a_3 & a_3 & e \\ e & e & a_2 & \neg a_1 \wedge \neg a_2 & a_1 \\ \hdashline e & e & e & e & \top \end{array}\right] = \begin{bmatrix} \mathbf{M}_1 & \mathbf{M}_{*1} & & \\ & \mathbf{M}_2 & \mathbf{M}_{*2} & \\ & & \mathbf{M}_3 & \mathbf{M}_{*3} \\ & & & \mathbf{M}_4 \end{bmatrix}$$

Since q_4 is the only accepting state and its corresponding block matrix $\mathbf{M}_4 = \top$, $\mathbf{W}_\mathcal{S}(q_4) = \mathbb{X}$ and $\kappa_4(x) = \mathbb{U}$ for all $x \in \mathbb{X}$. Algorithm 6.2 is then applied to $[\mathbf{M}_3 \quad \mathbf{M}_{*3}]$, $[\mathbf{M}_2 \quad \mathbf{M}_{*2}]$, and $[\mathbf{M}_1 \quad \mathbf{M}_{*1}]$ sequentially with only the inner loop. It returns $\mathbf{W}_\mathcal{S}(q_3)$, $\mathbf{W}_\mathcal{S}(q_2)$, and $\mathbf{W}_\mathcal{S}(q_1)$ along with the corresponding memoryless control strategies κ_3 κ_2, and κ_1.

As a summary of this section, we provide the following procedure of control synthesis for solving the LTL control problem:

S1 Translate φ into a DBA $\mathcal{A}_\varphi$, and trim $\mathcal{A}_\varphi$ by removing the invalid labeled transitions if necessary. Denote by $\tilde{\mathcal{A}}_\varphi$ the trimmed automaton.

S2 Perform Preprocess to $\tilde{\mathcal{A}}_\varphi$, which gives a sorted set of automaton states $\{q_{i_1}, \cdots, q_{i_l}\}$, where $i_j \in \{1, \cdots, |Q|\}$ for all $j = 1, \cdots, l$ denotes the index of the automaton state. Note that $1 \leq l \leq |Q|$. Then the corresponding upper triangular transition matrix $\mathbf{M}_\varphi$ is in the form

$$\begin{bmatrix} \mathbf{M}_1 & * & \\ & \ddots & * \\ & & \mathbf{M}_k \end{bmatrix}.$$

S3 Apply Algorithm 6.1 or 6.2 backwardly from $\mathbf{M}_k$ until $[\mathbf{M}_1 \quad *]$ to compute the $\mathcal{S}$-domain of $\{q_{i_1}, \cdots, q_{i_l}\}$. The corresponding memoryless control strategies will be generated at the same time. The $\mathcal{S}$-domain of any automaton state that is not in the sorted list is considered .

S4 Construct the automata-embedded control strategy according to Definition 6.5 based on the memoryless control strategies generated in S3.

6.5 Application to Motion Planning Problems

In most of the motion planning problems, control specifications are often given in the form of LTL formulas that are more complex than simple invariance and reachability objectives. In this section, we apply the proposed LTL control synthesis algorithm to solve motion planning problems.

The vehicle model we use for motion planning is (5.29), which is given in Section 5.5.2. We also adopt the same workspace and simulation parameter setting as in the *motion planning* example in Section 5.5.3, i.e., $\mathbb{X} = [0, 10] \times [0, 10] \times [-3.4, 3.4]$, the sampled control values are $\{\pm 0.9, \pm 0.6, \pm 0.3, 0\}$, sampling time $\tau_s = 0.3\text{s}$, and precision parameter $\varepsilon = 0.2$.

Example 6.4. We now study again the control specification in Example 6.3, where a_1, a_2 and a_3 are three atomic propositions assigned by a labeling function L to three isolated work areas Ω_1, Ω_2 and Ω_3. Using a_0 as the label of the rest of the workspace, then

$$L(x) = \begin{cases} a_1 & x \in \Omega_1 \\ a_2 & x \in \Omega_2 \\ a_3 & x \in \Omega_3 \\ a_0 & x \notin (\Omega_1 \cup \Omega_2 \cup \Omega_3) \end{cases}$$

Hence, we have $a_i \wedge a_j = \perp$ for all $i \neq j \in \{1, 2, 3\}$.

The workspace setup is shown in Figure 6.8. As the control objective, the order of the areas that the vehicle has to visit is: $\Omega_1 \to \Omega_2 \to \Omega_3 \to \neg\Omega_2 \to \Omega_1$.

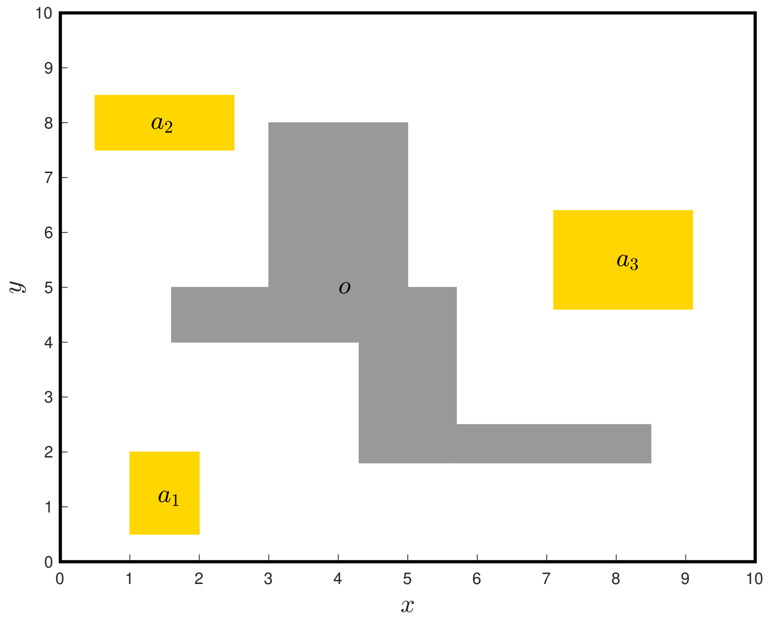


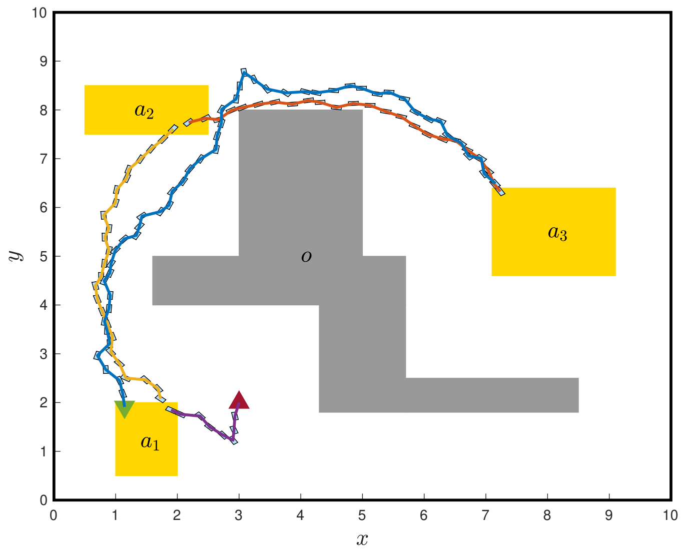
Figure 6.8: The top view of the motion planning workspace. The shaded area is marked as an obstacle. The target area $L^{-1}(a_1) = [1, 2] \times [0.5, 2] \times [-\pi, \pi]$, $L^{-1}(a_1) = [0.5, 2.5] \times [7.5, 8.5] \times [-\pi, \pi]$, and $L^{-1}(a_1) = [7.1, 9.1] \times [4.6, 6.4] \times [-\pi, \pi]$.

We obtain memoryless control strategies κ_q for each $q \in Q_c$ by performing Algorithm 6.1 with pre-processing, which is provided in ROCS. Since $\kappa_q(x)$ returns all valid control values given a memory value q and a state x of system $\mathcal{S}$ in the domain of κ_q, a random value u that confirms to $\kappa_q(x)$ is used.

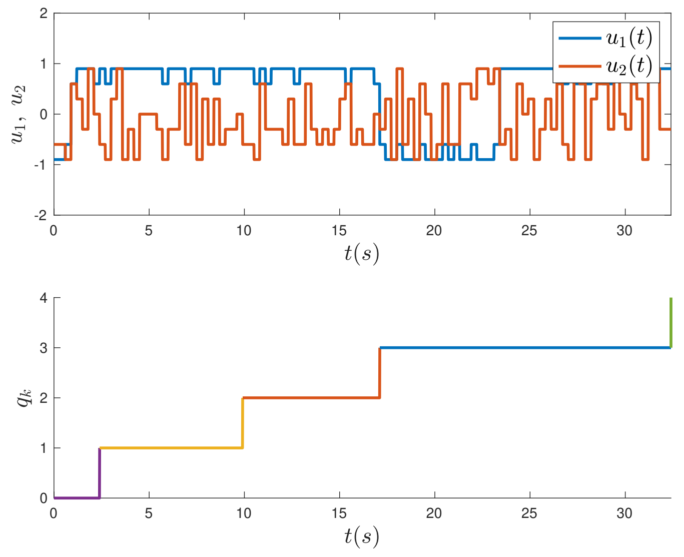
Figure 6.9 shows the simulation result from a initial condition $x_0 = (3, 2, 90°)$. It can be observed that the closed loop trajectory of the vehicle fulfills the expected visiting order specified by (6.22). Some more simulation results can be found in Figure 6.10 with different initial conditions.

To see how much the pre-processing procedure can help in saving computational time, we also perform the control synthesis algorithm without any pre-processing, i.e., Algorithm 6.1 takes in M_φ given in Example 6.3 directly. The time for control synthesis without pre-processing is 189.152s while it is 162.553s if the DBA is pre-processed.

Example 6.5 (Generalized Büchi Specification)**.** In this example, we use the same workspace setup as in Example 6.4, but the vehicle is expected to infinitely often visit Ω_1, Ω_2, and Ω_3, which

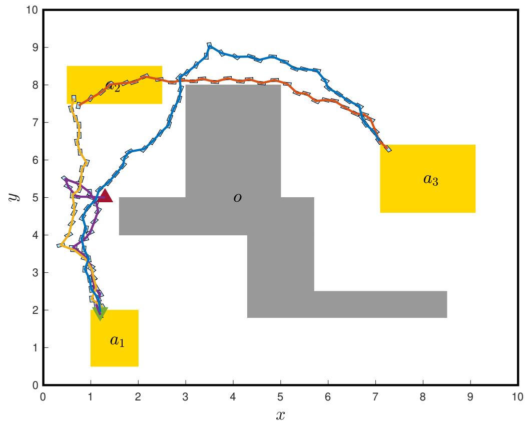


(a) The closed-loop 2-D trajectory of the vehicle starting from x_0, which is divided into 4 sections corresponding to 4 different control strategy memory values. The upward red and downward green triangles mark the initial and terminal states, respectively.

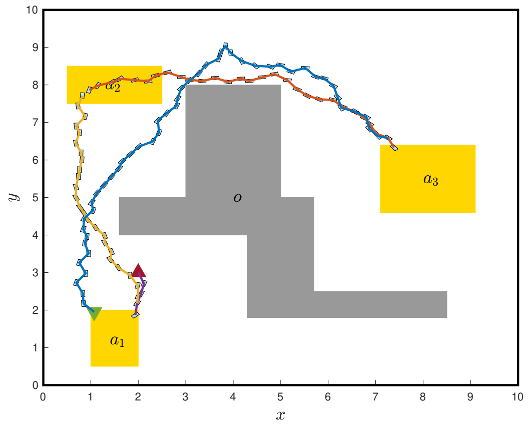


(b) The time histories of control values u_1, u_2 of system $\mathcal{S}$ and the memory value q, $q \in \{0, 1, 2, 3, 4\}$.

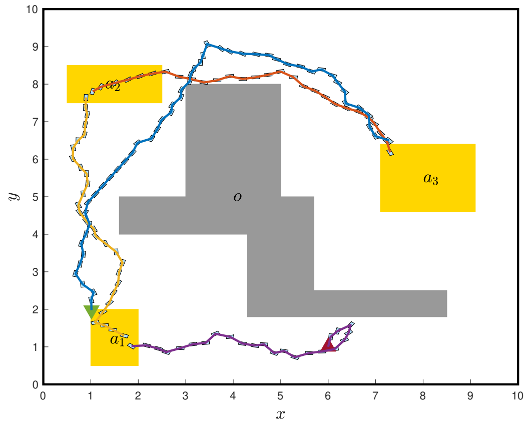
Figure 6.9: The simulation result for Example 6.4 with the initial condition $x_0 = (3, 2, 90°)$.



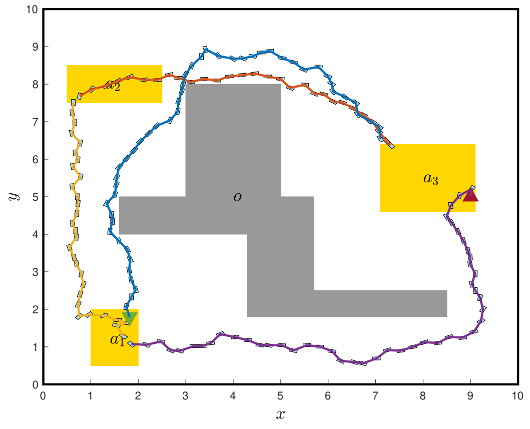
(a) $x_0 = (1.3, 5, 135°)$



(b) $x_0 = (2, 3, 90°)$



(c) $x_0 = (6, 1, 90°)$



(d) $x_0 = (9, 5, 45°)$

Figure 6.10: Closed-loop trajectories of the vehicle from 4 different initial conditions.

are labeled by a_1, a_2 and a_3, respectively. This requirement can be written as the following LTL formula:

$$\varphi = \bigwedge_{i=1}^{3} \Box\Diamond a_i. \tag{6.23}$$

The corresponding DBA of (6.23) is shown in the following Figure 6.11.

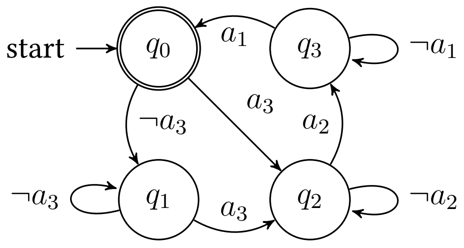


Figure 6.11: The DBA translated from (6.23) using Spot [37].

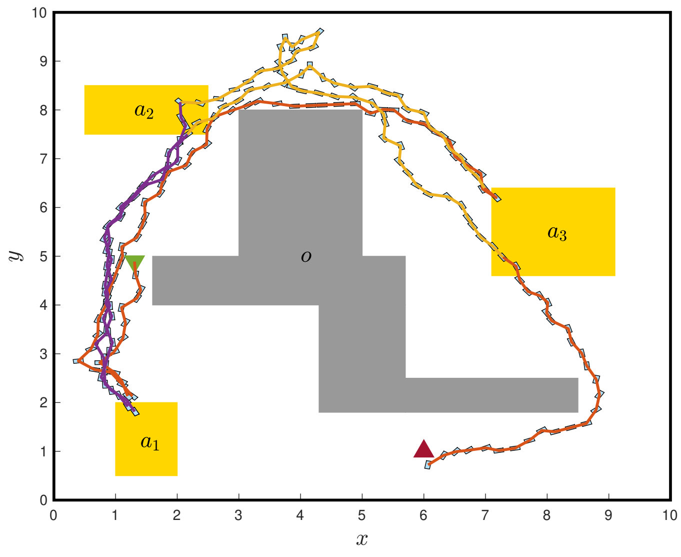
The DBA itself is an SCC, and thus no further pre-processing or topological sort is needed. Arranging the states in the order $q_3q_2q_1q_0$, the transition matrix is

$$\mathbf{M}_\varphi = \begin{bmatrix} \neg a_1 & e & e & a_1 \\ a_2 & \neg a_2 & e & e \\ e & a_3 & \neg a_3 & e \\ e & a_3 & \neg a_3 & e \end{bmatrix}.$$

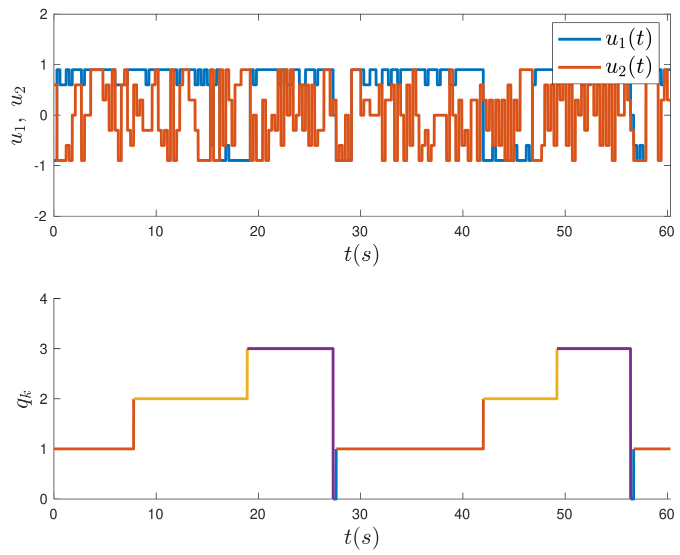
After overall 165 iterations in running Algorithm 6.1, we obtain the approximated $\mathcal{S}$-domains for automaton state q_0 to q_4 with their corresponding memoryless control strategies κ_1 κ_4. The time for control synthesis is 121.121s. The result of closed-loop control simulation with initial condition $x_0 = (6, 1, 90°)$ is shown in Figure 6.12. The automaton in Figure 6.11 always starts from the state q_0. Hence, the automaton state jumps to q_1 immediately since $x_0 \notin \Omega_3$ and the memoryless control strategy κ_1 is used until automaton state changes. As opposed to Example 6.4, the satisfaction of the generalized Büchi specification requires infinite time horizon. In our simulation, we test the controlled system for 60s, which shows the two whole periods of the update of the automaton states.

6.6 Summary

In this chapter, we considered a more general class of LTL specifications than invariance or reachability that are discussed in Chapter 4. This class of LTL formulas can be translated into



(a) The closed-loop 2-D trajectory within 60s that satisfies the specification (6.23). The sections of the trajectory related to different automaton states are marked in different colors: q_0-blue, q_1-orange, q_2-yellow, and q_3-purple.



(b) The time histories of control values u_1, u_2 of system $\mathcal{S}$ and the memory value q, $q \in \{0, 1, 2, 3\}$.

Figure 6.12: The simulation result for Example 6.5 with the initial condition $x_0 = (6, 1, 90°)$.

DBA. Control synthesis for system $\mathcal{S}$ with respect to such formulas is more difficult since the winning control strategies usually require memories, which leads to the questions raised at the beginning of this chapter.

To address the first question, we discussed the soundness and completeness of algorithm (6.11), which iteratively computes the exact winning set with respect to a DBA-recognizable LTL directly over the infinite state space of a dynamical system, instead of approximating the original dynamics by a finite abstraction as in [10]. This is because, without any stability assumptions, the completeness can not be guaranteed if the finite abstraction is constructed by over approximating system transitions for general nonlinear systems. To deal with general temporal properties, algorithm (6.11) applies a monotonic operator T^{δ}, which is based on the predecessor map, according to the transition matrix that reflects the graph structure of the DBA. We show that the soundness and completeness can be achieved under the assumption that the output of (6.11) is a fixed point. The construction of a finite abstraction as the first step is also avoided in [135, 7]). The algorithm in [135] is sound but not complete for the purpose of high computational efficiency, and [7] assume linear systems.

Close to our control synthesis setting is the discussion in [36] of symbolic algorithms for infinite-state games. The condition to make control synthesis algorithm finitely terminating is that the infinite state game structure should have equivalences with finite index [36]. This essentially means that there exists a finite abstraction that can represent the infinite-state dynamics. Since such a condition does not usually hold, we propose a finitely terminating algorithm (Algorithm 6.1) based on interval approximation of T^{δ}, which is proved to be sound and robustly complete.

As the answer to the second question about the structure of control strategies defined over the infinite state space to realize the given DBA-recognizable LTL formula, we proposed a structure that contains the DBA whose states represent control memories and the transition relation serves as a mechanism updating control memories. This structure is similar to the Last Visited Record (LVR) strategy for two-player games [89, 142] and the supervisor for discrete-event systems [109]. For finite transition systems, controller automata are also proposed in [71, 10]. Unlike these works, our control strategy is defined for dynamical systems, and an extension from finite-state systems to infinite-state systems is not straightforward.

A preprocessing procedure is also proposed to reduce computational cost. Before running algorithm (6.11), states of the given LTL-equivalent DBA are grouped together to produce a higher-level DAG. A topological sorting is then performed to determine the dependency among DAG nodes in terms of $\mathcal{S}$-domain computation. Control synthesis performed in this order can avoid unnecessary iterations. By complexity analysis in Section 6.4 as well as the empirical result in a motion planning scenario, we showed that the preprocessing is cost effective, because the size of the DBA is usually small compared to the discretized system, so that the preprocessing takes little time, but a little improvement can result in higher efficiency in control synthesis that involves nonlinear dynamics.

There is a connection between LTL control synthesis and reference tracking. In reference tracking, the dynamical system is controlled so that the state or output can track *a priori* reference signal:

$$\lim_{t\to\infty} \|x_t - r_t\|_2 = 0.$$

A controller that realizes the convergence to reference signal contains the reference signal model (called exosystem in [130]). The DBA-embedded control strategy structure and the traditional tracking controller are similar in the sense that following the same *internal model principle* [43].

Chapter 7

Control Synthesis for Sampled-Data Systems

Physical systems are often modeled by the ODEs:

$$\dot{x}(t) = f(x(t), u(t)) + d(t), \tag{7.1}$$

where $d(t)$ is a time-varying disturbance, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a smooth function. Similar to our discussion in Chapter 2, when $d(t) = 0$, system (7.1) reduces to the nominal form

$$\dot{x}(t) = f(x(t), u(t)). \tag{7.2}$$

Let I be an interval in $\mathbb{R}$. With a slight abuse of notation, we denote by $\mathbf{u} : I \to \mathbb{U}$ and $\mathbf{d} : I \to \mathbb{D}$ the continuous-time control and disturbance signal $u(t)$ and $d(t)$, respectively. We also denote by $\mathbb{U}^I$ and $\mathbb{D}^I$ the set of control and disturbance signals.

Definition 7.1. Given a control signal $\mathbf{u}$ and a disturbance signal $\mathbf{d}$, a *solution* of (7.1) from an initial state $x_0 \in \mathbb{X}$ over a time interval I within its maximum interval of existence is a function $\xi(t, x_0, \mathbf{u}, \mathbf{d})$ that satisfies

$$\frac{d\xi}{dt} = f(\xi(t), u(t)) + d(t), \quad \forall t \in I.$$

To apply either digital controller or control synthesis based on formal methods, system (7.1) is only measured, evaluated, and processed at discrete time instances. A Zero Order Hold (ZOH) is often used to hold sampled values during inter-sample periods. Such a controlled system with a continuous-time plant and discrete-time control components is typically called a *sampled-data system*.

Let $\tau > 0$ be a fixed sampling time for (7.1). The system state is only evaluated at discrete time instances $j\tau$ ($j \in \mathbb{N}$), and the control signal $u(t)$ is constant and takes values from a finite set $\mathbb{U}$ over $[0, \tau)$. Then the corresponding sampled-data system of (7.1) can also be translated into a transition system:

$$\mathcal{S}_\tau : \langle \mathbb{X},\ \mathbb{U},\ R_\tau,\ AP,\ L \rangle, \tag{7.3}$$

where the set of states $\mathbb{X}$, the set of inputs $\mathbb{U}$, the set of atomic propositions AP, and the labeling function L are defined as in (2.4). The transition relation $R_\tau : \mathbb{X} \times \mathbb{U} \to 2^{\mathbb{X}}$ is given by

$$R_\tau(x, u) \triangleq \{\xi(\tau, x, \mathbf{u}, \mathbf{d}) :\ u(t) \equiv u, d(t) \in \mathbb{D}, \forall t \in [0, \tau)\}\,.$$

In this sense, continuous-time system (7.1) is scaled over time and treated as a discrete-time system (7.3).

In this chapter, we aim to show how the proposed specification-guided method via interval computation can be applied to sampled-data systems.

7.1 Reachable Set Approximation Using Interval Analysis

The construction of inclusion functions for the sampled-data system of (7.1) is more difficult, because the post-transition states are not determined by a function explicitly, but related to reachable sets defined below.

Definition 7.2. The *reachable set* for system (7.1) after time τ from a set of initial states $X_0 \subseteq \mathbb{X}$ under a control signal $\mathbf{u} : [0, \tau) \to \mathbb{U}$ is defined by

$$\mathcal{R}_\tau(X_0, \mathbf{u}) = \{\xi(\tau, x_0, \mathbf{u}, \mathbf{d}) :\ \mathbf{d} \in \mathbb{D}^{[0,\tau)}, \|\mathbf{d}\|_\infty \le \delta, x_0 \in X_0\}. \tag{7.4}$$

To be more specific, the reachable set of (7.1) is denoted as $\mathcal{R}^*_\tau(X_0, \mathbf{u})$ if $d(t) \equiv 0$ and $\mathcal{R}^\delta_\tau(X_0, \mathbf{u})$ if $d(t)$ is bounded by $\delta > 0$, respectively.

We define a set of maps $\{\mathcal{R}_\tau(\cdot, u)\}_{u \in \mathbb{U}}$ by using constant control signals in (7.4). An over-approximation of the map $\mathcal{R}_\tau(\cdot, u)$ by definition serves as an inclusion function for the sampled-data system (2.2) of (7.1).

A standard algorithm for over-approximating the reachable set from an initial interval $[x_0]$ relies on the kth degree of Taylor expansion of the solution at time $t = 0$ [95]:

$$\mathcal{R}_\tau([x_0], u) \subseteq \sum_{i=0}^{k} f^{[i]}([x_0], u)\frac{\tau^i}{i!} + f^{[k+1]}(\widehat{[x_0]}, u)\frac{\tau^{k+1}}{(k+1)!}, \tag{7.5}$$

where $\widehat{[x_0]}$ is an *a priori* enclosure for the solution on $[0, \tau)$ and the sequence of functions $f_u^{[i]}(x)$ $(i \geq 0)$ are defined by

$$\begin{aligned} f^{[0]}(x, u) &= x, \\ f^{[i]}(x, u) &= \frac{\partial f^{[i-1]}(\cdot, u)}{\partial x} f(x, u),\ i \geq 1. \end{aligned}$$

We can over-approximate the function $f^{[i]}(\cdot, u)$ in (7.5) by using convergent inclusion functions $[f]^{[i]}(\cdot, u)$. Then

$$\begin{aligned} \widehat{\mathcal{R}}_\tau^k([x_0], u) &= \sum_{i=0}^{k} [f]^{[i]}([x_0], u)\frac{\tau^i}{i!} + [f]^{[k+1]}(\widehat{[x_0]}, u)\frac{\tau^{k+1}}{(k+1)!} \\ &\supseteq \mathcal{R}_\tau([x_0], u). \end{aligned} \tag{7.6}$$

Therefore, the computation of $[f]([x], u)$ can be replaced by $\widehat{\mathcal{R}}_\tau([x], u)$ in (7.6) for sampled-data systems. An interval $\widehat{[x_0]}$ can function as an *a priori* enclosure for $[x_0]$ if there exists some $\bar{k}$ that

$$[x_0] + \sum_{i=1}^{\bar{k}-1} [f]^{[i]}([x_0], u)\frac{[0, \tau^i]}{i!} + [f]^{[\bar{k}]}(\widehat{[x_0]}, u)\frac{[0, \tau^{\bar{k}}]}{\bar{k}!} \subseteq \widehat{[x_0]}. \tag{7.7}$$

We show that such an *a priori* enclosure can always be found under the following assumption.

Assumption 7.1. Let $\mathbb{X}, \mathbb{U}$ be compact and $[\mathbb{X}]$ be an interval containing $\mathbb{X}$. For a given order $k_{\max} \geq 1$, there exists a constant $K > 0$ and inclusion functions $[f]^{[i]}(\cdot, u)$ of $f^{[i]}(\cdot, u)$ such that for all $1 \leq i \leq k_{\max}$,

$$\mathrm{wid}([f]^{[i]}([x]), u) \leq K \mathrm{wid}([x]),\ \forall [x] \subseteq [\mathbb{X}], u \in \mathbb{U}.$$

Similar to Assumption 3.1 and 3.2, the above assumption can be guaranteed by $f(\cdot, u)$ being smooth, which implies the bounded partial derivative of $f^{[i]}(\cdot, u)$ on any compact set.

Lemma 7.1. Suppose that there exists an order $k_{\max} \geq 1$ for a sampled-data system (2.2) such that Assumption 7.1 holds on $\mathbb{X}$. Let

$$M_u = \sup_{1 \leq i \leq k_{\max}, x \in \mathbb{X}} \left\| f^{[i]}(x,u) \right\|_\infty, \quad W = \sup_{[x] \subseteq \mathbb{X}} \{\text{wid}([x])\}.$$

For any interval $[x_0] \subseteq \mathbb{X}$, if τ, $\epsilon \in (0,1)$ and the order $\bar{k} \in [1, k_{\max}]$ are chosen such that

$$[x_0] + [-1,1](M_u + K\text{wid}([x_0])\,(e^\tau - 1)\,\mathbf{1}^n + [-2\epsilon, 2\epsilon] \subseteq [\mathbb{X}],$$
$$\frac{\tau^i}{i!} < \frac{2\epsilon}{M_u + KW}, \quad \forall \bar{k} \leq i \leq k_{\max},$$

then

$$\widehat{[x_0]} \triangleq [x_0] + \sum_{i=1}^{\bar{k}-1} [f]^{[i]}([x_0], u) \frac{[0, \tau^i]}{i!} + [-2\epsilon, 2\epsilon] \tag{7.8}$$

is an *a priori* enclosure, i.e., $\widehat{[x_0]} \subseteq [\mathbb{X}]$.

Proof. For any $\epsilon > 0$, there exists $\bar{k} \in [1, k_{\max}]$ and $\tau > 0$ such that $\tau^i/i! < 2\epsilon/(M_u + KW)$ for all $\bar{k} \leq i \leq k_{\max}$. Under Assumption 7.1, we can construct a centered inclusion function $[f]^{[i]}([x], u) = f^{[i]}(\bar{x}, u) + K([x] - \bar{x})$ for $1 \leq i \leq k_{\max}$, where $\bar{x}$ is the center point of the interval $[x]$. Then for any interval $[x] \subseteq [\mathbb{X}]$,

$$\begin{aligned}
\text{wid}([f]^{[\bar{k}]}([x], u)) = \text{wid}(K([x] - \bar{x})) &\leq KW/2 \Rightarrow \\
[f]^{[\bar{k}]}([x], u) &\subseteq [-1,1](M_u + KW/2)\mathbf{1}^n \Rightarrow \\
[f]^{[\bar{k}]}([x], u)\frac{[0, \tau^{\bar{k}}]}{\bar{k}!} &\subseteq [-1,1](M_u + KW/2)\frac{\tau^{\bar{k}}}{\bar{k}!}\mathbf{1}^n \\
&\subseteq [-2\epsilon, 2\epsilon]\mathbf{1}^n.
\end{aligned}$$

Let x_0 be the center point of $[x_0]$. Similarly, we have

$$\begin{aligned}
&\sum_{i=1}^{\bar{k}-1} \left(f^{[i]}(x_0, u) + K([x_0] - x_0)\right) \frac{[0, \tau^i]}{i!} \\
&\subseteq [-1,1](M_u + K\text{wid}([x_0])\left(\sum_{i=1}^{\infty} \frac{\tau^i}{i!} - \sum_{i=\bar{k}}^{\infty} \frac{\tau^i}{i!}\right)\mathbf{1}^n \\
&\subseteq [-1,1](M_u + K\text{wid}([x_0])\,(e^\tau - 1)\,\mathbf{1}^n.
\end{aligned}$$

Hence, $[x_0] + [-1,1](M_u + K\mathrm{wid}([x_0]))\,(e^\tau - 1)\,\mathbf{1}^n + [-2\epsilon, 2\epsilon]\mathbf{1}^n \subseteq [\mathbb{X}]$ implies that $\widehat{[x_0]} \subseteq [\mathbb{X}]$. Furthermore,

$$[x_0] + \sum_{i=1}^{\bar{k}-1} [f]^{[i]}([x_0], u)\frac{[0, \tau^i]}{i!} + [f]^{[\bar{k}]}(\widehat{[x_0]}, u)\frac{[0, \tau^{\bar{k}}]}{\bar{k}!} \subseteq$$
$$[x_0] + \sum_{i=1}^{\bar{k}-1} [f]^{[i]}([x_0], u)\frac{[0, \tau^i]}{i!} + [-2\epsilon, 2\epsilon] = \widehat{[x_0]},$$

which means that the $\widehat{[x_0]}$ defined above satisfies (7.7). □

7.2 Robust Completeness

It remains to determine the order k for a sufficiently close approximation such that algorithm (4.27) is still guaranteed to be robustly complete for sampled-data systems. We additionally define a interval-valued system

$$[\mathcal{S}_\tau] : \langle \mathbb{X},\ \mathbb{U},\ [R_\tau],\ AP,\ L \rangle,$$

which differs from $\mathcal{S}_\tau$ by the transition relation $[R_\tau]$ is given by (7.6). In the following, we let $[\mathrm{Pre}]^\varepsilon$ be the set of intervals defined in (5.2) with $\underline{A}$ returned by $\textsc{Pre}([\mathcal{S}_\tau], B, A, \varepsilon)$.

Theorem 7.1 (Soundness and Robustly Completeness)**.** Consider system $\mathcal{S}_\tau$ and a DBA convertible LTL or reach-and-stay formula φ_{rs}. Suppose that Assumption 7.1 holds for a sampled-data system of (7.1). Let W be the output of Algorithm 6.1 (or algorithm (4.27)) by using $[\mathrm{Pre}]^\varepsilon$ to construct $[\underline{T}]^\varepsilon$ (or $\widehat{\mathrm{Pre}} = [\mathrm{Pre}]^\varepsilon$). Then

$$\mathrm{Win}^{\delta}_{\mathcal{S}_\tau}(\varphi) \subseteq W \subseteq \mathrm{Win}_{\mathcal{S}_\tau}(\varphi), \tag{7.9}$$

if the *a priori* enclosure $\widehat{[x_0]}$ and the corresponding order $\bar{k}$ are constructed by Lemma 7.1 for any interval $[x_0] \subseteq \mathbb{X}$ with $\mathrm{wid}([x_0]) < \varepsilon$, and additionally,

$$k \geq \max\left\{\bar{k} - 1, \left\lceil \frac{\log \frac{(1-\alpha)\delta}{K\bar{w}} + \log(\bar{k}+1)!}{\log \tau} \right\rceil \right\}, \tag{7.10}$$

$$\varepsilon \leq \frac{\alpha\tau}{2Ke^\tau}\delta, \tag{7.11}$$

where $\lceil\cdot\rceil$ is the ceiling function, $\alpha \in (0,1)$, $\bar{w} = \mathrm{wid}(\widehat{[x_0]})$.

The fraction α is used to distribute the error allowed in interval approximation for the first k terms and the remainder. The proof of Theorem 7.1 is based on Proposition 7.1 below.

Proposition 7.1. Let $D \subseteq \mathbb{X}$. Assume that $\|f(x,u) - f(y,u)\|_\infty \leq \rho_L \|x-y\|_\infty$ for all $x, y \in D$ and $u \in \mathbb{U}$. The reachable set of (7.1) at time τ from an initial set of states $X_0 \subseteq D$ under a control signal $\mathbf{u} : [0,\tau) \to \mathbb{U}$ satisfies

$$\mathcal{R}^*_\tau(X_0, \mathbf{u}) \oplus \mathcal{B}_{r_1} \subseteq \mathcal{R}^\delta_\tau(X_0, \mathbf{u}) \subseteq \mathcal{R}^*_\tau(X_0, \mathbf{u}) \oplus \mathcal{B}_{r_2}, \tag{7.12}$$

where $r_1 = \delta\tau$ and $r_2 = \delta\rho_L^{-1}(e^{\rho_L \tau} - 1)$.

Proof. Consider solutions $x(t)$ and $y(t)$ of $\dot{x}(t) = f(x(t), u(t)) + d(t)$ and $\dot{y}(t) = f(y(t), u(t))$ with $x(0) = y(0)$, respectively. Then

$$\begin{aligned}\|\dot{x}(t) - \dot{y}(t)\|_\infty &= \|f(x(t),u(t)) - f(y(t),u(t)) + d(t)\|_\infty \\ &\leq \rho_L \|x(t) - y(t)\|_\infty + \|d(t)\|_\infty .\end{aligned}$$

Letting $z(t) = \|x(t) - y(t)\|_\infty \geq 0$ gives $\dot{z}(t) \leq \rho_L z(t) + \delta$. By Gronwall's Lemma, we obtain that $\|z(t)\|_\infty \leq \delta\rho_L^{-1}(e^{\rho_L\tau} - 1)$, which proves the right part of (7.12).

To prove the left part, let

$$d(t) = \delta \frac{f(x(t),u(t)) - f(y(t),u(t))}{\|f(x(t),u(t)) - f(y(t),u(t))\|_\infty}.$$

It follows that

$$\dot{z}(t) = \delta + \|f(x(t),u(t)) - f(y(t),u(t))\|_\infty \geq \delta.$$

Hence $z(\tau) \geq \delta\tau$ and the left part is proved. □

Proof of Theorem 7.1. For any interval $[x_0] \subseteq \mathbb{X}$, by Lemma 7.1, there exists an order $\bar{k}$ and an *a priori* enclosure $\widehat{[x_0]}$ such that $\widehat{\mathcal{R}}_\tau([x_0], u)$ obtained by (7.6) is an over-approximation of the reachable set $\mathcal{R}_\tau([x_0], u)$.

We first derive a sufficient condition such that $\mathrm{Pre}^\delta(B|A) \subseteq [\underline{\mathrm{Pre}}]^\varepsilon(B|A) \subseteq \mathrm{Pre}(B|A)$, where $B \subseteq A \subseteq \mathbb{X}$. It is trivial that $[\underline{\mathrm{Pre}}]^\varepsilon(B|A) \subseteq \mathrm{Pre}(B|A)$ for all k and τ, so we only consider the conditions such that $\mathrm{Pre}^\delta(B|A) \subseteq [\underline{\mathrm{Pre}}]^\varepsilon(B|A)$ here. Let x_0 be the center point of an arbitrary interval $[x_0] \subseteq \mathbb{X}$ with $\mathrm{wid}([x_0]) \leq 2\varepsilon$. Under Assumption 7.1, we rewrite (7.6) in the following centered form

$$\widehat{\mathcal{R}}_\tau^k([x_0],u) = \sum_{i=0}^{k} f^{[i]}(x_0,u)\frac{\tau^i}{i!} + \underbrace{[f]^{[k+1]}(\widehat{[x_0]},u)\frac{\tau^{k+1}}{(k+1)!}}_{\text{truncation error}}$$

$$+ \underbrace{\sum_{i=0}^{k} K([x_0]-x_0)\frac{\tau^i}{i!}}_{\text{propagated enclosure}}.$$

For the propagated enclosure,

$$\begin{aligned}\mathrm{wid}\left(\sum_{i=0}^{k} K([x_0]-x_0)\frac{\tau^i}{i!}\right) &\le 2K\varepsilon \sum_{i=0}^{k}\frac{\tau^i}{i!}\\ &\le 2K\varepsilon\sum_{i=0}^{\infty}\frac{\tau^i}{i!} = 2K\varepsilon e^\tau.\end{aligned}$$

For the truncation error, we have

$$\mathrm{wid}([f]^{[k+1]}(\widehat{[x_0]},u)\frac{\tau^{k+1}}{(k+1)!}) \le K\bar{w}\frac{\tau^{k+1}}{(k+1)!}.$$

Let $\alpha \in (0,1)$, $k \ge \bar{k}$ and

$$K\bar{w}\frac{\tau^{k+1}}{(\bar{k}+1)!} \le (1-\alpha)\delta\tau, \tag{7.13}$$

$$2K\varepsilon e^\tau \le \alpha\delta\tau. \tag{7.14}$$

Then $w\left(\widehat{\mathcal{R}}_\tau^k([x_0],u)\right) \le (1-\alpha)\delta\tau + \alpha\delta\tau = \delta\tau$, which leads to $\widehat{\mathcal{R}}_\tau^k([x_0],u) \subseteq \mathcal{R}_\tau(x_0,u)\oplus\mathcal{B}_{\delta\tau}$. Solving for k and ε in (7.13) and (7.14) gives $k \ge \left\lceil \log(K\bar{w})^{-1}(1-\alpha)\delta + \log(\bar{k}+1)!/\log\tau\right\rceil$ and (7.11). We take the maximum of k and $\bar{k}-1$ to guarantee that $\widehat{[x_0]}$ is an *a priori* enclosure. Hence, we arrive at (7.10).

Suppose that $x_0 \in \mathrm{Pre}^\delta(B|A)$, i.e., $\mathcal{R}_\tau^\delta(x_0,u) \subseteq B$ for some $u \in \mathbb{U}$. Assumption 7.1 implies that $f(\cdot,u)$ is Lipschitz over $\mathbb{X}$ for all $u\in\mathbb{U}$. Then we have $\widehat{\mathcal{R}}_\tau^k([x_0],u) \subseteq \mathcal{R}_\tau^*(x_0,u)\oplus\mathcal{B}_{\tau\delta} \subseteq \mathcal{R}_\tau^\delta(x_0,u) \subseteq B$ by Proposition 7.1. It implies that $x \notin [x] \in A_c$, because $\widehat{\mathcal{R}}_\tau^k([x],u)\cap B \neq \emptyset$. Any interval $[x]\in\Delta A$ that contains x_0 satisfies $[x]\subseteq[x_0]$. It then follows that $\widehat{\mathcal{R}}_\tau^k([x],u)\subseteq B$, but $\widehat{\mathcal{R}}_\tau^k([x], u) \nsubseteq B$ by Algorithm 5.1. Hence, $x_0 \notin [x] \in \Delta A$ and thus it is only possible that $x_0 \in [x] \in \underline{A}$, which means $x_0 \in [\underline{\mathrm{Pre}}]^\varepsilon(B|A)$.

If φ is DBA convertible, the resulting interval $[\underline{T}]^\varepsilon$ operator satisfies (6.17) by Lemma 6.1. By Theorem 6.2, (7.9) is proved. If φ is a reach-and-stay formula, then (5.21) is satisfied, which shows (7.9). □

Remark 7.1. Evaluating the constant K over the entire state space $\mathbb{X}$ will make the choice of ε conservative. A remedy is to compute K locally based on a guess of the *a priori* enclosure $\widehat{[x_0]}$ and a given order threshold $k_{\max}$:

$$K = \max_{\substack{i=1,\cdots,k_{\max}\\ u\in\mathbb{U}}} \left\{ \frac{\mathrm{wid}([f]^{[i]}(\widehat{[x_0]}, u))}{\mathrm{wid}(\widehat{[x_0]})} \right\}.$$

The *a priori* enclosure $\widehat{[x_0]}$ is then updated by Lemma 7.1. If the updated enclosure is not contained in $\widehat{[x_0]}$, then $\widehat{[x_0]}$ needs to be enlarged. The size of $\widehat{[x_0]}$ is related to the initial interval $[x_0]$. In the modified algorithm, the size of $[x_0]$ is managed though subdivision. The coefficient K will then be updated to determine the local maximum size of the intervals. The defect of using such local evaluation, however, is that it will incur extra computational cost in a single loop.

Example 7.1. Consider a sampled-data system $\mathcal{S}$ with sampling time $\tau_s = 0.05$s and the reversed Van der Pol dynamics:

$$\begin{cases} \dot{x}_1 = -x_2, \\ \dot{x}_2 = x_1 + (x_1^2 - 1)x_2. \end{cases}$$

Suppose that $\mathcal{S}$ is subject to a uniformly distributed disturbance with bound $\delta = 10$. The following sampled position in the state space are analyzed: $p_1 = (0.5, 0.3)$, $p_2 = (-1.2, -0.6)$ and $p_3 = (-2.3, -1.7)$, where p_1 and p_2 are inside of the limit cycle while p_3 is outside.

Table 7.1: Local parameters for reachable set computation.

Samples	p_1	p_2	p_3
k	3	3	4
K	15.27	64.78	49137.10
ε	0.01557	0.00367	4.8×10^{-6}

Table 7.1 lists the parameters addressed in Theorem 7.1, which are computed based on local dynamics. It can be seen that K is large since the system is unstable around p_3, and thus we need to use a much smaller interval in order that the approximation error of the reachable set is no bigger than the one caused by disturbance.

7.3 Examples

In this section, we give two examples showing the effectiveness of the proposed interval-based method for control synthesis for sampled-data systems with respect to LTL specifications. One is an application to the estimation of regions of attraction for nonlinear systems and the other is the stabilization of inverted pendulum on cart. Both cases have been studied extensively in the literature.

7.3.1 Estimation of Regions-of-Attraction

A problem of interest in the study of dynamical systems is to determine the Region of Attraction (ROA) of an equilibrium point. This problem has important applications in safety-critical industries such as aviation and power systems, where determining the operating envelope of an aircraft or a power network is vital. In the literature, computational methods for determining the ROA for nonlinear systems have been developed by way of Lyapunov functions. The key aspect is to search Lyapunov functions that maximize the estimated ROA. For this purpose, linear matrix inequalities [29] and sum-of-square programming techniques [126, 127] are used for the construction of such Lyapunov functions for polynomial systems. Using Lyapunov functions with fixed forms, subsets of the ROAs can also be obtained by solving a constraint satisfaction problem [125]. How to choose the form of Lyapunov functions, however, remains a challenging problem.

Consider the continuous-time system

$$\dot{x}(t) = f(x(t)), \tag{7.15}$$

where $x \in \mathbb{R}^n$, f is continuously differentiable and the origin is a hyperbolic stable equilibrium point. Let $\xi(t, x_0)$ denote the solution of (7.15) with initial condition x_0. Its ROA is a subset of initial conditions from which the solution converges to the origin, i.e.,

$$\left\{ x_0 \in \mathbb{R}^n : \lim_{t \to \infty} \xi(t, x_0) = 0 \right\}.$$

System (7.15) is a special case of (7.1) with a single input value and zero disturbance. We show next that the ROA approximation problem for system (7.15) can be interpreted as a reach-and-stay control problem with the specification $\varphi(\Omega)$, where $\Omega \subseteq \mathbb{R}^n$ is a subset of the exact ROA of system (7.15) containing the origin.

A routine to determine the subset Ω is to use the linearization at the origin. Let A be the Jacobian matrix at the origin. Then a quadratic Lyapunov function $V(x) = x^T P x$ exists and can be constructed by solving $A^T P + PA = -Q$, where P, Q are positive definite matrices and P is symmetric [69, Theorem 4.7]. To estimate the neighborhood around the origin where the quadratic Lyapunov function $V(x)$ decreases along the system solution, we write $\dot{x} = f(x) = Ax + g(x)$, where $g(x)$ contains higher-order terms of x, i.e., $\lim_{\|x\|_2 \to 0} \|g(x)\|_2 / \|x\|_2 = 0$. Hence, by the definition of function limit, for any $r > 0$, there exits $e > 0$ such that

$$\|x\|_2 < e \implies \|g(x)\|_2 / \|x\|_2 < r \Leftrightarrow \|g(x)\|_2 < r \|x\|_2 .$$

Let $\lambda_{\text{min}}(Q)$ denotes the minimum eigenvalue of Q. Then

$$\begin{aligned}\dot{V}(x) &= x^T P f + f^T P x \\ &= x^T P(Ax + g(x)) + (x^T A^T + g^T(x))Px \\ &= x^T(PA + A^T P)x + 2x^T P g(x) \\ &= -x^T Q x + 2x^T P g(x) \\ &\leq (-\lambda_{\text{min}}(Q) + 2r \|P\|_2) \|x\|_2^2 .\end{aligned}$$

Given $r, c > 0$, let $S_r \triangleq \{x \in \mathbb{R}^n : \|g(x)\|_2 < r \|x\|_2\}$ and $\Omega_c \triangleq \left\{x \in \mathbb{R}^n : x^T P x \leq c\right\}$. We can first choose r to satisfy

$$-\lambda_{\text{min}}(Q) + 2r \|P\|_2 < 0 \tag{7.16}$$

and then determine c such that $\Omega_c \subseteq S_r$. This will guarantee that Ω_c is invariant and any solution staying inside Ω_c will converge to the origin. Consequently, any state in $\mathbb{R}^n$ that can reach Ω_c in a finite time horizon will also converge to the origin. In this case, the ROA is equivalent to the winning set of $\varphi(\Omega_c)$.

To demonstrate the correctness and effectiveness of such an interpretation, we consider a sampled-data system $\mathcal{S}$ with sampling time $\tau_s = 0.05$s and the reversed Van der Pol dynamics in Example 7.1. The state space is assumed to be $\mathbb{X} = [-4, 4] \times [-4, 4]$. Letting Q be the identity matrix gives

$$P = \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 1 \end{bmatrix}.$$

We choose $r = 0.2754$, $c = 1.43$ and $\Omega_c = \{x \in \mathbb{R}^n : x^T P x \leq c\}$.

We approximate the ROA of the Van der Pol equations using algorithm (4.27) with different precision control parameters and display the results together with the real limit cycle in Fig. 7.1. As observed, a higher precision yields a closer inner-approximation to the real ROA. By setting ε sufficiently small, the estimated boundary of ROA can be of arbitrarily close to the real limit cycle.

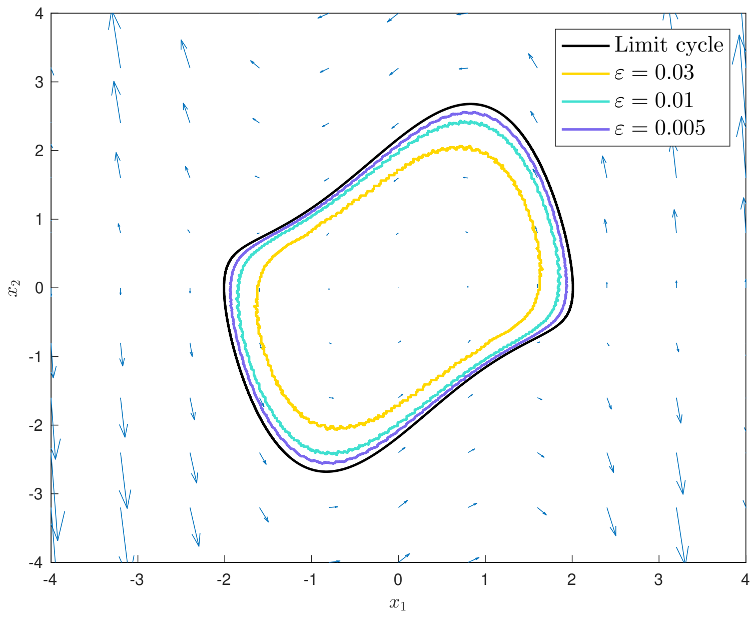


Figure 7.1: Comparison of inner-approximations of the ROA for reversed Van der Pol sampled-data system with three different precisions.

Formulating the problem of ROA approximation as a reachability or reach-and-stay control synthesis problem releases the burden of choosing proper Lyapunov functions. The required smoothness condition is less strict than being polynomial in many of the methods for ROA estimation.

7.3.2 Stabilization of Inverted Pendulum

In the example, we aim to stabilize the inverted pendulum, through which the scalability of the proposed method can be demonstrated by using adaptive precisions.

Consider an inverted pendulum on a cart (see Figure 7.2) modeled by the continuous-time ODEs:

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = \frac{mgl}{J_t}\sin x_1 - \frac{b}{J_t}x_2 + \frac{l}{J_t}\cos x_1 u, \end{cases} \tag{7.17}$$

where $x_1 = \theta$ (rad) is the angle of the pendulum to the upper vertical line, x_2 is the angle change rate $\dot{\theta}$ (rad/s), and u is the force applied to the cart; $J_t = J + ml^2$, $m = 0.2$kg, $g = 9.8\text{m/s}^2$, $l = 0.3$m, $J = 0.006\text{kgm}^2$, $b = 0.1$N/m/s.

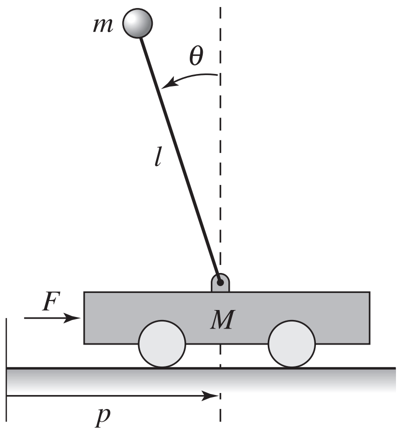


Figure 7.2: Inverted pendulum on cart [90].

We aim to control the pendulum to the upright position. This specification can be written as the LTL formula $\varphi = \Diamond\Box G$, where $\Omega = L^{-1}(G) = [-0.05, 0.05] \times [-0.01, 0.01]$. Let the state space $\mathbb{X} = [-2, 2] \times [-3.2, 3.2]$. The sampled-data system of (7.17) with the sampling time $\tau_s = 0.01$s is used, and the control input u is chosen from the finite set $\mathbb{U} = \{-0.5, -0.45, \cdots, 0.45, 0.5\}$ obtained by a sampling granularity $\mu = 0.05$.

The modified Algorithm 4.27 with (7.6) is used to perform reach-and-stay control synthesis, as opposed to using the local growth bound [113]:

$$\beta(\eta, u) = e^{L(u)\tau_s}\eta, \ L(u) = \begin{bmatrix} 0 & 1 \\ \sqrt{24.5^2 + 12.5^2u^2} & -4.17 \end{bmatrix},$$

where $\eta = [\eta_1, \eta_2]$ is the grid width.

In this case, the target stabilization area G is tiny compared to the entire state space X. In order to maintain the pendulum angle and angle change rate in the region G, the value of the precision control parameter ε has to be determined according to the size of G. Thus, we use a precision $\varepsilon = 0.001$ for $[\underline{\mathrm{Pre}}_{\mu}]^{\varepsilon}(X_{i+1}^{j}|W_i^j)$ as an implementation of $\widehat{\mathrm{Pre}}(\widehat{X}_{i+1}^{j}|W_i^j)$ in (4.27). Since the state space $\mathbb{X}$ is nearly 40 times the size of G, the partition of $\mathbb{X}$ will contain a huge number of cells if a uniform precision $\varepsilon = 0.001$ is used in Algorithm 4.27. To obtain an acceptable computational complexity, we use a relative precision, which is determined with respect to the size of the winning set throughout iterations, for the computation of $Z_i = [\underline{\mathrm{Pre}}]^{\varepsilon}(\widehat{X}_{i+1}^{\infty}|V_i)$. The inner loop precision reflects the bound of the perturbation that can be tolerated by the resulting switching strategy.

For an initial condition $(\theta_0, \dot{\theta}_0) = (1, 1)$, the closed-loop simulation result (see Figure 7.3) shows that, applying the extracted switching strategy, the angle of pendulum is stabilized to zero with a steady-state error of 0.05 within 0.5s.

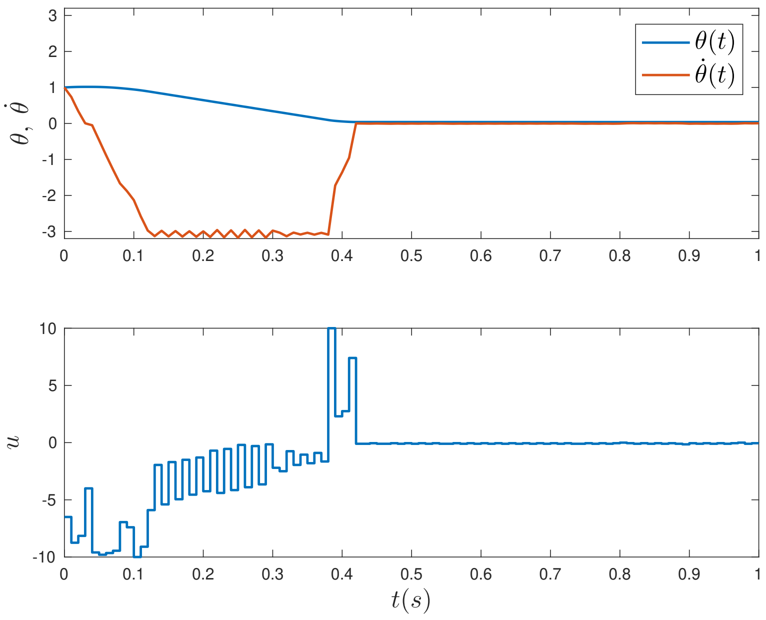


Figure 7.3: Closed-loop simulation with the initial condition $(\theta_0, \dot{\theta}_0) = (1, 1)$ for system (7.17).

This system is neither globally asymptotically stable nor incrementally asymptotically stable around the upright position. Hence, abstraction-based methods using bisimulation relations [53] do not apply while over-approximations [139, 86, 84, 113] based on uniform grids can be used. To achieve equivalent stabilization precisions, the grid size needs to be at least 0.001 according to the size of the stabilization area. Using uniform partitions as in abstraction-based methods, the entire safe region $[\![a_s]\!]$ is discretized to overall 2.56×10^7 cells with grid points of width 0.001. Using SCOTS [118], computation of the abstraction lasts for more than 12 hours without returning any result. In contrast, our algorithm generates a winning set covering most of the state space in around 480 seconds with 26340 partitions.

7.4 Summary

This chapter extended the previous results on sound and robustly complete LTL control synthesis to sampled-data systems. The results in this chapter can also be found in [82].

For sampled-data systems, the problem of applying the proposed interval-based control synthesis algorithms is that predecessors can not be approximated by Algorithm 5.1 using a convergent interval function $[f]$ of f. This is because system state at the next time step is not determined by the function f in (7.1) as in the cases for discrete-time systems. We resolved this problem by using higher-order Taylor model for the validated (over) approximation of the reachable set $\mathcal{R}_\tau(X_0, u)$ from an initial set of states $X_0 \subseteq \mathbb{X}$, which essentially plays the role of $[R](X_0, u)$ in Algorithm 5.1. The Taylor model has been used for computing validated solutions of initial value problems [95, 94] and reachability analysis for nonlinear and hybrid systems [27, 26]. Unlike the application of Taylor model in the literature, where the approximation precision is adjusted by the small time steps in a fixed horizon, we showed in this chapter that the approximation can be arbitrarily precise by controlling the order of the Taylor model. This is usually what we hope for the control synthesis for sampled-data systems, in which the system information is updated every sampling time.

In addition to the consistency with the proposed interval-based control synthesis scheme, using interval in reachable set approximation also shows that the proposed control synthesis algorithms can still be made sound and robustly complete by choosing a sufficiently high order in the computation of the Taylor model.

Chapter 8

Application to Reactive Locomotion Planning in Constrained Environments

In the field of robotics, planning and control of bipedal locomotion has been one of the attractive topics. Due to the complexity of whole-body dynamic locomotion (WBDL) behaviors and requirements of being reactive to the dynamic environment, planning and control often live on a hierarchical structure [14]. In this way, the complicated control problem is decomposed into simpler problems which are solved at different levels. The WBDL model, therefore, is simplified accordingly at different levels to serve different control purposes.

This chapter illustrates an application of the proposed control synthesis method in previous chapters to a reactive locomotion planning problem, in which the bipedal robot is expected to behave in respond to the changes of the environment. Recently, formal methods have gained increasing attention for solving such problems because of the correctness guarantee and LTL formulas are favored for specifying temporal and reactivity properties [9, 103, 58, 132, 136].

Contact-based decision and planning method, which operates over a set of robotic maneuvers determined by contact points [128], is applied in the design of control hierarchy. At the high level, a motion plan, which is sequence of locomotion modes (chosen from a library of simplified locomotion models) and corresponding setpoints, are generated by solving a two-player game between the planner and the dynamic environment with the constraints expressed in LTL formulas. At the low level, the bipedal robot is controlled so that the robot behaviors can be classified into different locomotion modes. To guarantee that such a plan can be realized by actual system dynamics, transitions between two locomotion modes need to be verified and mode-switching control strategies need to be constructed.

In the literature, formal methods are often used at the planner level, where the underlying system is finite, to reason about reactive planning strategies. In this chapter, we will demonstrate how the proposed control synthesis method can be applied to a middle level in which locomotion switching strategies are designed.

8.1 Reactive Locomotion Planning Problem

First of all, we introduce the reactive locomotion planning problem in this section, which is formalized as a switching control problem between abstracted multi-contact locomotion models.

8.1.1 Hybrid System Model of Bipedal Locomotion

In general, dynamics of mechanical systems are described by their rate of linear and angular momenta, which are usually affected by external force and/or torque:

$$\dot{\boldsymbol{l}} = m\ddot{\boldsymbol{p}}_{\text{com}} = \sum_{i}^{N_c} \boldsymbol{f}_i + m\boldsymbol{g}, \tag{8.1}$$

$$\dot{\boldsymbol{k}} = \sum_{i}^{N_c} (\boldsymbol{p}_i - \boldsymbol{p}_{\text{com}}) \times \boldsymbol{f}_i + \boldsymbol{\tau}_i, \tag{8.2}$$

where N_c is the number of limb contacts, $\boldsymbol{l} \in \mathbb{R}^3$ and $\boldsymbol{k} \in \mathbb{R}^3$ represent the centroidal linear and angular momenta, respectively, $\boldsymbol{f}_i \in \mathbb{R}^3$ is the ith ground reaction force, m is the total mass of the robot, $\boldsymbol{\tau}_i \in \mathbb{R}^3$ is the contact torque of the ith limb, the variables

$$\boldsymbol{g} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}, \quad \boldsymbol{p}_i = \begin{bmatrix} p_{i,x} \\ p_{i,y} \\ p_{i,z} \end{bmatrix}, \quad \boldsymbol{p}_{\text{com}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

correspond to the gravity field, the position of the ith limb contact position, and the center of mass (CoM) position respectively.

The above general model can be simplified based on different contact modes under the assumptions that are commonly imposed to make the problem tractable [6]. In this WBDL control problem, six locomotion modes are considered to produce various behaviors [140], which are also pictured in Figure 8.1.

The prismatic inverted pendulum mode (PIPM). In a normal environment, the bipedal robot exhibits a normal walking gait: there is a single foot contact with the floor in each walking

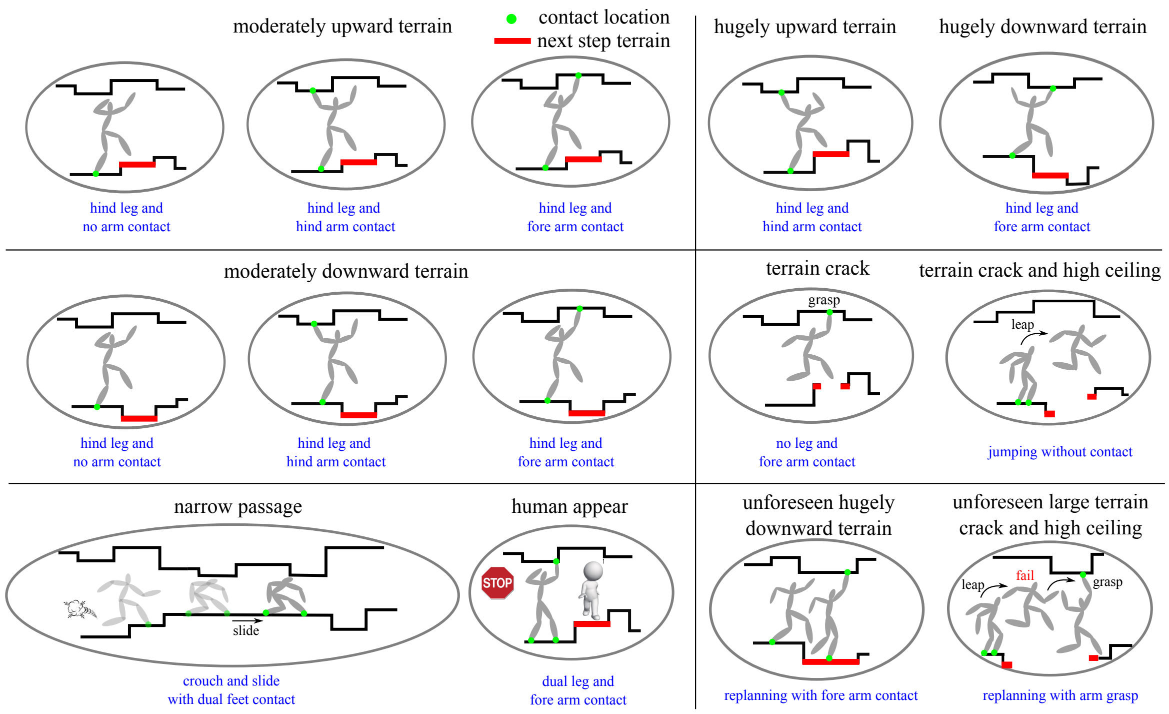

moderately upward terrain
● contact location
next step terrain
hugely upward terrain
hugely downward terrain
hind leg and no arm contact
hind leg and hind arm contact
hind leg and fore arm contact
hind leg and hind arm contact
hind leg and fore arm contact
moderately downward terrain
terrain crack
terrain crack and high ceiling
grasp
leap
hind leg and no arm contact
hind leg and hind arm contact
hind leg and fore arm contact
no leg and fore arm contact
jumping without contact
narrow passage
human appear
unforeseen hugely downward terrain
unforeseen large terrain crack and high ceiling
STOP
slide
leap
fail
grasp
crouch and slide with dual feet contact
dual leg and fore arm contact
replanning with fore arm contact
replanning with arm grasp

Figure 8.1: Contact-based planning strategies for locomotion in rough terrains [141]. Events motivated by ordinary accidents in human daily lives, such as a crack on the terrain and the sudden appearance of a human, are treated as emergency events, and incorporated into the allowable environment.

period. Such walking dynamics can be considered as a inverted pendulum model. Since $N_c = 1$ in this mode, we can simplify (8.2) to

$$(\boldsymbol{p}_{\text{com}} - \boldsymbol{p}_{\text{foot}}) \times (\boldsymbol{f}_{\text{com}} + m\,\boldsymbol{g}) = -\boldsymbol{\tau}_{\text{com}},$$

where $\boldsymbol{f}_{\text{foot}}$ is the force imposed at the contact foot point, and $\boldsymbol{f}_{\text{com}}$ is the vector of CoM inertial forces:

$$\boldsymbol{f}_{\text{com}} = m\ddot{\boldsymbol{p}}_{\text{com}} = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix},$$

Assume that the bipedal locomotion follows a piece-wise linear CoM path surface

$$\psi_{\text{com}}(x, y, z) = z - ax - by - c = 0, \tag{8.3}$$

where a, b and c are the coefficients of the surface. Thus, the dynamics in the vertical direction are represented by $\ddot{z} = a\ddot{x} + b\ddot{y}$ and not explicitly shown here.

Hence, the mathematical model for this mode is

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \omega_{\text{PIPM}}^2 \begin{bmatrix} x - x_{\text{foot}} - \frac{\tau_y}{mg} \\ y - y_{\text{foot}} - \frac{\tau_x}{mg} \end{bmatrix}, \tag{8.4}$$

where $\ddot{x}$ and $\ddot{y}$ are CoM accelerations aligned with sagittal and lateral directions, and

$$\omega_{\text{PIPM}} = \sqrt{\frac{g}{z_{\text{PIPM}}^{\text{apex}}}}, \quad z_{\text{PIPM}}^{\text{apex}} = (a \cdot x_{\text{foot}} + b \cdot x_{\text{foot}} + c - z_{\text{foot}})$$

is the PIPM phase-space asymptotic slope [140]. The control input is

$$\boldsymbol{u} = \begin{bmatrix} x_{\text{foot}} \\ y_{\text{foot}} \\ \omega_{\text{PIPM}} \\ \tau_x \\ \tau_y \end{bmatrix}.$$

The prismatic pendulum mode (PPM).. When the terrain is cracked, the robot has to grasp the overhead support to swing over an unsafe region using brachiation. The system dynamics can be approximated as a pendulum model. For a single hand contact, we have

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = -\omega_{\text{PPM}}^2 \begin{bmatrix} x - x_{\text{hand}} - \frac{\tau_y}{mg} \\ y - y_{\text{hand}} - \frac{\tau_x}{mg} \end{bmatrix}, \tag{8.5}$$

where similarly

$$\omega_{\text{PPM}} = \sqrt{\frac{g}{z_{\text{PPM}}^{\text{apex}}}}, \quad z_{\text{PPM}}^{\text{apex}} = (z_{\text{hand}} - a \cdot x_{\text{hand}} - b \cdot x_{\text{hand}} - c)$$

given the same surface given in (8.3). Similarly, vertical direction dynamics are represented by $\ddot{z} = a\ddot{x} + b\ddot{y}$. A difference between PIPM and PPM lies in that PPM dynamics is inherently stable since the CoM is always attracted to move towards the apex position while the PIPM dynamic is not. This study assumes the robot can firmly grasp the overhead support once receiving the upper limb contact command.

The stop-launch mode (SLM). When a human appears, the robot has to come to a stop, wait until human disappears, and start to move forward. The task in this mode consists on decelerating the CoM motion to zero and accelerating it from zero again. We name this model as a SLM with a constant CoM sagittal accelerations:

$$\dot{l}_x = ma_x, \ \dot{l}_y = ma_y, \ \dot{l}_z = ma_z,$$

where a_x, a_y, a_z are the control inputs. The resulting phase-space trajectory is a parabolic manifold.

The multi-contact mode (MCM). When the robot maneuvers through unstructured rough terrains, arms and legs in contact can accelerate and decelerate the CoM according to terrain height variations. To make the dynamics tractable, we assume a known constant vertical acceleration a_z in each step and neglect of the angular momentum k_z around the z-axis [6], which leads to

$$\sum_i^{N_c} f_{i,z} = m(\ddot{z} - g).$$

With multiple point contacts, we let $\boldsymbol{\tau}_i = 0$ for all $i \leq N_c$ in (8.2), and the dynamics can be simplified to

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\varphi} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \sum_i^{N_c} f_{i,x}/m \\ \sum_i^{N_c} f_{i,y}/m \\ -(\ddot{z} - g) \cdot y + z \cdot \sum_i^{N_c} f_{i,y}/m - \sum_i^{N_c} p_{i,z} \cdot f_{i,x}/m + \sum_i^{N_c} p_{i,z} \cdot f_{i,z}/m \\ (\ddot{z} - g) \cdot x - z \cdot \sum_i^{N_c} f_{i,x}/m + \sum_i^{N_c} p_{i,z} \cdot f_{i,x}/m - \sum_i^{N_c} p_{i,y} \cdot f_{i,y}/m \end{bmatrix},$$

where φ and θ are torso roll and pitch angles aligned with the CoM sagittal and lateral directions as derived from (8.2). The external force vector $(f_{i,x}, f_{i,y}, f_{i,z})$ represents the ith contact force. The vertical position z is a function of x and y defined *a priori*.

The hopping mode (HM). This model applies when the locomotion model needs to jump over an unsafe region. In this case, the CoM dynamics follow a free-falling ballistic trajectory. We have

$$\ddot{x} = \ddot{y} = 0, \ddot{z} = -g.$$

The trajectory is fully controlled by the initial condition, where a discontinuous jump in the CoM state can occur and be used to generate a desired linear momentum. For instance, when the robot jumps over a cracked terrain, it needs to push the ground as the foot lifts to generate a sufficiently large sagittal linear acceleration.

The sliding mode (SM). This model applies when the robot needs to slide through a constrained region. The CoM dynamics are subject to a constant friction force. Thus, $\ddot{x}$ is a constant negative value, and we assume $\ddot{y} = 0, \ddot{z} = 0$. The sagittal linear velocity decays at a constant rate.

In this research, the considered locomotion modes are selected from the set

$$\mathcal{M} \triangleq \{p_{\text{PIPM}}, p_{\text{MCM}}, p_{\text{PPM}}, p_{\text{SLM}}, p_{\text{HM}}, p_{\text{SM}}\}.$$

The switched system representation. Given the continuous locomotion modes above, we formulate the WBDL as a switched system:

$$\dot{\boldsymbol{\xi}}(\zeta) = f_{p(\zeta)}\big(\boldsymbol{\xi}(\zeta), \boldsymbol{u}(\zeta), \boldsymbol{d}(\zeta)\big),\ p(\zeta) \in \mathcal{M}, \tag{8.6}$$

where $\boldsymbol{\xi}(\zeta) \in \Xi \subseteq \mathbb{R}^{12}$ denotes the 12 dimensional CoM position and angular state vector of the robot at $\zeta \geq 0$ on the manifolds of the dynamics (cf. (8.17)), the phase progression variable ζ, analogous to time, represents the current phase progression on a locomotion trajectory, the functions $p(\cdot) : \mathbb{R} \to \mathcal{M}$ and $\boldsymbol{d}(\cdot) \in \mathcal{D} \subseteq \mathbb{R}^d$ $(0 \leq d \leq 12)$ are the switching signal and external disturbance, respectively, and f_p denotes the dynamics under mode $p \in \mathcal{M}$. The control input is denoted by

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{p}_{\text{contact}} \\ \omega \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \in \mathbb{U} \subseteq \mathbb{R}^7,$$

where $\boldsymbol{p}_{\text{contact}}$ represents a set of contact position vectors in which each contact position vector is three-dimensional, ω represents the slope of the phase-space asymptote dependent on specific locomotion modes as defined in the above modes, and τ_x, τ_y, and τ_z represent the torso torques along x, y, and z axis, respectively.

The sampled-data system of (8.6) with a constant sampling time $\Delta\zeta \geq 0$ can be written in the form of a transition system (see Definition 2.1)

$$\mathcal{S}_L = (\Xi, \mathcal{M} \times \mathbb{U}, R_L, AP, L), \tag{8.7}$$

where $R_L : \Xi \times \mathbb{U} \times \mathcal{M} \to \Xi$ is determined by

$$R_L(\boldsymbol{\xi}, \boldsymbol{u}, p) \triangleq \Big\{\boldsymbol{\xi}(\Delta\zeta) \in \Xi :\ \dot{\boldsymbol{\xi}}(\zeta) = f_p\big(\boldsymbol{\xi}(\zeta), \boldsymbol{u}, \boldsymbol{d}(\zeta)\big), \boldsymbol{\xi}(0) = \boldsymbol{\xi}, \forall \boldsymbol{d}(\zeta) \in \mathcal{D}, \forall \zeta \in [0, \Delta\zeta]\Big\}.$$

8.1.2 Reactive Locomotion Planning via Hierarchical Strategy

In order to be responsive to any changes in the environment, it is often necessary to predict different scenarios that could happen in the environment and include it in the design of overall motion planning strategy.

Definition 8.1 (Environment System). The environment can be modeled as a finite transition system:

$$\mathcal{S}_e \triangleq (\mathcal{E}, \mathcal{I}_e, R_e, AP_e, L_e), \tag{8.8}$$

where $\mathcal{E}$ is a finite set of environmental states, $R_e : \mathcal{E} \to \mathcal{E}$ is a transition relation, $\mathcal{I}_e = \mathcal{E}_0 \subseteq \mathcal{E}$ is a set of initial states, AP_e is a set of atomic propositions, $L_e : \mathcal{E} \to 2^{AP_e}$ is a labeling function mapping the state to an atomic proposition.

The following definition of *product system* incorporates the external environment $\mathcal{S}_e$ as the model that generates uncontrollable exogenous inputs.

Definition 8.2 (Product System). The product system of system $\mathcal{S}_L$ and $\mathcal{S}_e$ is a tuple:

$$\mathcal{S}_{\text{prod}} \triangleq (\Xi, \mathcal{M} \times \mathbb{U}, \mathcal{E}, R_{\text{prod}}, \widetilde{AP}, \widetilde{L}), \tag{8.9}$$

where Ξ, $\mathcal{M}$, $\mathbb{U}$ are defined in (8.7), and $\mathcal{E}$ is a finite set of uncontrollable environmental actions, which is defined in (8.8) as the set of environmental states, $R_{\text{prod}} : \Xi \times \mathcal{M} \times \mathbb{U} \times \mathcal{E} \to 2^{\Xi}$ is the transition relation, $\widetilde{AP}$ is a set of atomic propositions, $\widetilde{L} : \Xi \to 2^{\widetilde{AP}}$ is a labeling function mapping the state to an atomic proposition.

Definition 8.3 (Execution of A Product System). An *execution* $\boldsymbol{\pi}$ of system $\mathcal{S}_{\text{prod}}$ is an infinite sequence $\boldsymbol{\pi} = (\boldsymbol{\xi}_0, \boldsymbol{p}_0, \boldsymbol{u}_0, \boldsymbol{e}_0)(\boldsymbol{\xi}_1, \boldsymbol{p}_1, \boldsymbol{u}_1, \boldsymbol{e}_1)(\boldsymbol{\xi}_2, \boldsymbol{p}_2, \boldsymbol{u}_2, \boldsymbol{e}_2)\cdots$, where $\boldsymbol{\xi}_i \in \Xi$, $p_i \in \mathcal{M}$, $\boldsymbol{u}_i \in \mathbb{U}$, and $\boldsymbol{e}_i \in \mathcal{E}$ for all $i \in \mathbb{N}$. The word generated from $\boldsymbol{\pi}$ is $\mathrm{w}_{\pi} = \widetilde{L}(\boldsymbol{\xi}_0)\widetilde{L}(\boldsymbol{\xi}_1)\widetilde{L}(\boldsymbol{\xi}_2)\cdots$.

The execution $\boldsymbol{\pi}$ is said to satisfy an LTL formula φ, if and only if the word w_γ satisfies φ. If all executions of $\mathcal{S}_{\text{prod}}$ satisfy φ, we say that $\mathcal{S}_{\text{prod}}$ satisfies φ, i.e., $\mathcal{S}_{\text{prod}} \models \varphi$.

Planning and control of a complex robotic system as (8.6), which is high dimensional, contains multiple control inputs, and is subject to environmental constraints, is often achieved via hierarchical design [141]:

- The high-level planner works on an abstracted state space called *keyframe state space* $\mathcal{Q}$. A *keyframe state* $q = (\boldsymbol{p}_{\text{contact}}, \dot{x}_{\text{apex}}) \in \mathcal{Q}$ of a locomotion system is in general a pair of contact location $\boldsymbol{p}_{\text{contact}}$ and the apex state $\dot{x}_{\text{apex}}$ when the CoM velocity reaches the local minimal or maximal value. The planner determines the sequence of non-periodic keyframe states and locomotion modes by the planning strategy

$$\kappa_h : \mathcal{M} \times \mathcal{Q} \times \mathcal{E} \rightarrow \mathcal{M} \times \mathcal{Q}, \tag{8.10}$$

- The low-level controller within each mode directly control system dynamics in local region in the state space so that the assumptions for the modeling of different modes can be satisfied. A controller at this level is usually pre-designed and not considered in the planning problem.
- The middle-level mode-transition controller guarantees the feasibility of mode switching required by the planner. It also generate a control strategy κ_l that takes in the command from the planner:

$$\kappa_l : \mathcal{M} \times \mathcal{Q} \times \Xi \rightarrow 2^{\mathbb{U}}. \tag{8.11}$$

In this way, the overall control strategy defined as (2.7) for the locomotion planning problem can be decomposed to a planning strategy (8.10) and a mode-transition control strategy (8.11). Figure 8.2 shows the hierarchical framework of locomotion planning described above.

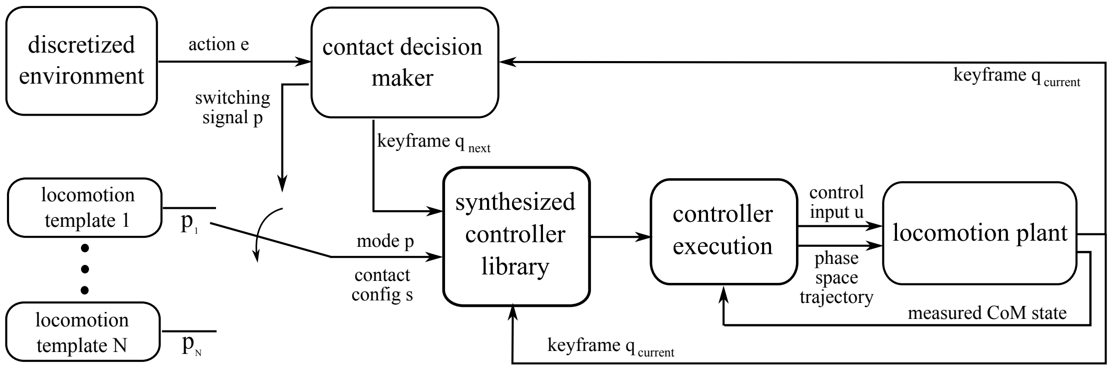


Figure 8.2: Hierarchical locomotion planner structure [141].

The specifications for the product system that cover the reactivity property is often given in the assume-guarantee form [19]:

$$\varphi = \big(\varphi_e \Rightarrow (\varphi_q \wedge \varphi_s)\big), \tag{8.12}$$

where φ_e and φ_q, φ_s are propositions for the admissible environment actions, the keyframe states, and the correct overall system behavior, respectively. In particular, φ_s specifies the conditions of mode switching.

The formula φ_v ($v \in \{e, q, s\}$) in (8.14) is expressed in the form

$$\varphi_v = \varphi^v_{\text{init}} \bigwedge_{i \in I_{\text{safety}}} \Box \varphi^v_{\text{trans},i} \bigwedge_{i \in I_{\text{goal}}} \Box \Diamond \varphi^v_{\text{goal},i}, \tag{8.13}$$

where φ^v_{init}, $\varphi^v_{\text{trans},i}$, and $\varphi^v_{\text{goal},i}$ are propositional formulas that pose constraints to the initial conditions, transitions, and goals, respectively.

Let the set of states $\mathcal{E}$ for the environment $\mathcal{S}_e$ be

$$\mathcal{E} \triangleq \mathcal{E}_{\text{terrain}} \cup \mathcal{E}_{\text{emergency}} = \{e_{\text{md}}, e_{\text{hd}}, e_{\text{mu}}, e_{\text{hu}}\} \cup \{e_{\text{tc-nc}}, e_{\text{tc-hc}}, e_{\text{ha}}, e_{\text{np}}\}, \tag{8.14}$$

where the elements in $\mathcal{E}_{\text{terrain}}$ denote different height terrain actions, as illustrated in Figure 8.1. For instance, e_{md} denotes moderatelyDownward terrain. The actions in $\mathcal{E}_{\text{emergency}}$ represent sudden events, i.e. terrainCrack-normalCeiling, terrainCrack-highCeiling, humanAppear, and narrowPassage.

Example 8.1 (Examples of formulas in φ_e)**.** An example of the initial specification of the environment is

$$\varphi^e_{\text{init}} = \neg e_{\text{tc-nc}} \wedge \neg e_{\text{tc-hc}} \wedge \neg e_{\text{ha}} \wedge \neg e_{\text{np}},$$

which means the initial environment should not be tough situations as terrain crack in normal or high ceiling, human appear, and narrow passages. The safety specifications will be given such as "if the current environment action is terrainCrack-highCeiling, then the next environmental action can not be terrainCrack-highCeiling, humanAppear, nor narrowPassage" with the equivalent LTL form:

$$\Box\Big(e_{\text{sc-hc}} \Rightarrow \neg(e_{\text{sc-hc}} \wedge e_{\text{ha}} \wedge e_{\text{np}})\Big)$$

To determine the sequence of locomotion modes, the set of robot actions corresponding to different modes is defined as follows:

$$\mathcal{F} \triangleq \{s_{\text{li-aj}}, : \forall (i, j) \in \mathcal{I}_{\text{index}}\}, \tag{8.15}$$

where l and a are short for leg and arm, respectively, the set of contact limb relative positions is $\mathcal{I}_{\text{index}} = \{(\text{h}, \text{n}), (\text{h}, \text{h}), (\text{h}, \text{f}), (\text{d}, \text{h}), (\text{d}, \text{f}), (\text{d}, \text{d}), (\text{d}, \text{n}), (\text{n}, \text{f}), (\text{n}, \text{n})\}$ with h, f, d and n represent hind, fore, dual and no contacts, respectively.

Example 8.2 (Example of φ_s)**.** An example for the term φ^s_{trans} of φ_s in response to varying-height terrain $\mathcal{E}_{\text{terrain}}$ is specified as

$$\square\Big((e_{\text{md}} \vee e_{\text{mu}}) \Rightarrow (p_{\text{PIPM}} \wedge s_{\text{lh-an}}) \vee \big(p_{\text{MCM}} \wedge (s_{\text{lh-ah}} \vee s_{\text{lh-af}})\big)\Big)$$
$$\bigwedge \square(e_{\text{hu}} \Rightarrow p_{\text{MCM}} \wedge s_{\text{lh-ah}}) \bigwedge \square(e_{\text{hd}} \Rightarrow p_{\text{MCM}} \wedge s_{\text{lh-af}}),$$

where $s_{\text{lh-af}}$, for example, means the legHindArmFore contact configuration in the sense that the robot's hind leg and the fore arm are in contact for that action while the other two limbs are not in contact.

The keyframe states consist of ordinary and special types:

$$\begin{aligned}\mathcal{Q} \coloneqq \mathcal{Q}_{\text{ordinary}} \cup \mathcal{Q}_{\text{special}} =& \{q_{i\text{-}j\text{-}k},\ i \in \mathcal{I}_{\text{ordinary-behavior}},\ \forall (j,k) \in \mathcal{I}_{\text{level}} \times \mathcal{I}_{\text{level}}\} \\ & \cup \{q_{i\text{-}j},\ i \in \mathcal{I}_{\text{special-behavior}},\ \forall j \in \mathcal{I}_{\text{level}}\}\end{aligned} \tag{8.16}$$

where $\mathcal{I}_{\text{ordinary-behavior}} = \{\text{walk, brachiation}\}$ and $\mathcal{I}_{\text{special-behavior}} = \{\text{stop, hop, slide}\}$ are ordinary and special behaviors, respectively. An apex velocity index j and a step length index k refer to the set $\mathcal{Q}_{\text{level}} = \{s, m, l\}$ whose elements are three different keyframe levels: s (Small), m (Medium) and l (Large). For instance, $q_{\text{walk-s-l}}$ represents walkSmallVelocityLargeStep, a walking keyframe with a small apex velocity, and a large step length.

Example 8.3 (Example for φ_q)**.** One of the formulas for $\varphi^q_{\text{trans},i}$ in φ_q is:

$$\square\big(\bigcirc e_{\text{np}} \Rightarrow \bigcirc(q_{\text{slide-s}} \vee q_{\text{slide-m}} \vee q_{\text{slide-l}})\big),$$

which means that if there is a narrow passage, i.e., e_{np}, then the next key frame state is q_{slide} relying on a specific apex velocity, regardless of the current q.

Details on the full set of specifications that consist the LTL specification (8.12) can be found in [141, Section 4].

Based on the above definitions, the locomotion planning problem can be described as:

Problem 8.1 (Contact-Based Reactive WBDL Planning)**.** Given bipedal robot $\mathcal{S}_L$ in (8.7) with a set of initial condition $\Xi_0 \subseteq \Xi$, environmental system $\mathcal{S}_e$ in (8.8), and an LTL specification φ in the form of (8.12), synthesize a planning strategy (8.10) and a mode-transition control strategy (8.11) such that the resulting execution π defined in Definition 8.3 satisfies φ in the sense that $\pi \models \varphi$ for all initial conditions in Ξ_0.

A two-player game problem can be formulated and analyzed to synthesize a planning strategy over a high-level finite abstract state space $\mathcal{M} \times \mathcal{Q} \times \mathcal{E}$ as illustrated in [141, Section 4]. The rest of the chapter will focus on the synthesis of the mode-transition control strategy for the middle layer.

8.2 Robust Switching Between Locomotion Modes

Uncertainty is ubiquitous in the modeling of the WBDL and environment, e.g. sensor noise, model inaccuracy, external disturbance, sudden environmental changes, contact surface geometry uncertainty. As a result, commands from the symbolic task planner are possible unrealizable for the low-level dynamics. Mismatches in real-time plan execution are not desired, and hence the task of the middle layer is to verify if the transitions between two modes at certain keyframe states can be achieved and construct a mode-transition strategy if possible.

The synthesis of the mode-transition strategy for every single walking step is performed on the robust abstractions for the dynamics of two successive modes with respect to reachability control specifications, which is given by the high-level planner. The finite abstractions as well as the *robustness margin sets* are constructed over the phase-space manifolds of the locomotion for the sake of consistency with the dynamics.

Assuming the x and y axes can be decoupled, we focus on the the dynamics along x axis (the dynamics along y axis is similar) and define a mapping between the Euclidean and Riemmanian:

$$\begin{bmatrix} \zeta \\ \sigma \end{bmatrix} = \mathcal{Z}_p(\boldsymbol{\xi}) = \begin{bmatrix} \mathcal{Z}_{p,\zeta}(x, \dot{x}) \\ \mathcal{Z}_{p,\sigma}(x, \dot{x}) \end{bmatrix} \tag{8.17}$$

where ζ is phase progression variable, σ is the tangent manifold, which can be used to measure deviations from the nominal locomotion trajectory in the phase-space, and $\mathcal{Z}_p(\boldsymbol{\xi})$ is a nonlinear mapping of the CoM state $(x, \dot{x})$ to the Riemannian space states for locomotion mode p. The inverse mapping $\mathcal{Z}_p$ is denoted by $\mathcal{Z}_p^{-1}$.

The specific mapping for each of the 6 locomotion modes are given in Appendix B.

8.2.1 Robustness Margin Sets in One Walking Step

Mode transitions usually take place in one walking step. A *one walking step (OWS)* is then composed of two consecutive semi-step phase-space trajectories. The first semi-step trajectory starts at the first keyframe state q_1 and ends at the contact switch, which will be determined by the mode-transition control strategy, and the second semi-step trajectory starts at the contact switch and ends at the second keyframe state q_2.

To guarantee that the motion planner yields plans that are robust to disturbances, we introduce ϵ_1 and ϵ_2 as initial and final robustness margins in the one walking step, respectively so that the neighborhood of nominal initial and final keyframe states q_1 and q_2 can also be considered for mode transition. The formal definition of robustness margin sets is provided below.

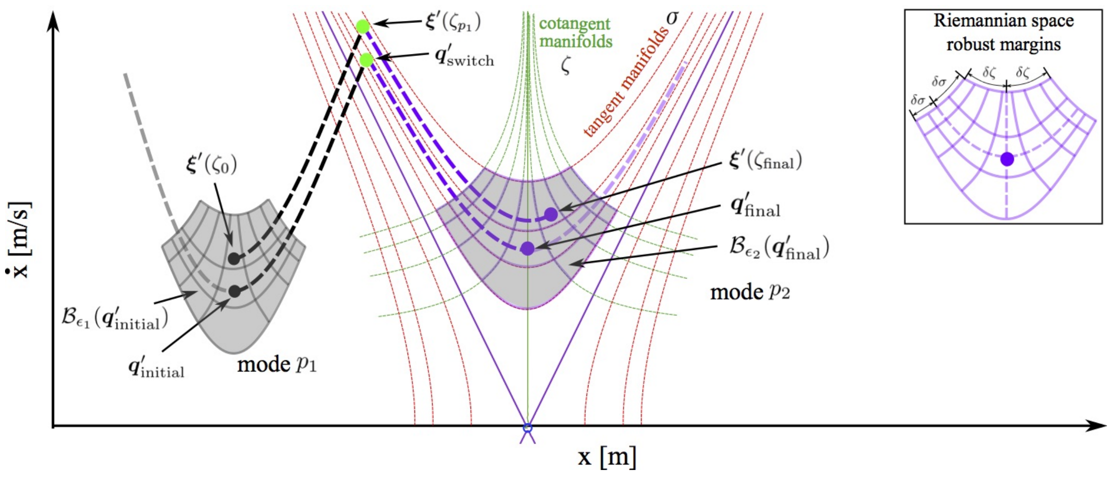
Definition 8.4 (Robustness Margin Sets). Given initial and final keyframe states q_1 and q_2, let $\zeta_0 = \mathcal{Z}_{p,\zeta}(q_1)$, $0 = \mathcal{Z}_{p,\sigma}(q_1)$, $\zeta_f = \mathcal{Z}_{p,\zeta}(q_2)$, and $0 = \mathcal{Z}_{p,\sigma}(q_2)$, where $\mathcal{Z}_{p,\zeta}(\cdot)$ and $\mathcal{Z}_{p,\sigma}(\cdot)$ are given in (8.17). Also let $\epsilon_1 = [\delta\zeta_{\epsilon_1}, \delta\sigma_{\epsilon_1}]$ and $\epsilon_2 = [\delta\zeta_{\epsilon_2}, \delta\sigma_{\epsilon_2}]$. The robustness margin set of q_1 and q_2 are

$$\mathcal{B}_{\epsilon_1}(q_1) \triangleq \left\{\mathcal{Z}_p^{-1}(\zeta,\sigma) \mid \zeta \in [\zeta_0 - \delta\zeta_{\epsilon_1}, \zeta_0 + \delta\zeta_{\epsilon_1}], \sigma \in [-\delta\sigma_{\epsilon_1}, \delta\sigma_{\epsilon_1}]\right\}, \tag{8.18}$$

$$\mathcal{B}_{\epsilon_2}(q_2) \triangleq \left\{\mathcal{Z}_p^{-1}(\zeta,\sigma) \mid \zeta \in [\zeta_f - \delta\zeta_{\epsilon_2}, \zeta_f + \delta\zeta_{\epsilon_2}], \sigma \in [-\delta\sigma_{\epsilon_2}, \delta\sigma_{\epsilon_2}]\right\}, \tag{8.19}$$

Remark 8.1. The keyframe states q_1 and q_2 and their robustness margin sets $\mathcal{B}_{\epsilon_1}(q_1)$ and $\mathcal{B}_{\epsilon_2}(q_2)$ are defined in the Euclidean space while the margins ϵ_1 and ϵ_2 are in the phase space.

Figure 8.3 gives an intuition of how the robustness margin sets defined in a walking step.



$\boldsymbol{\xi}'(\zeta_{p_1})$
$\boldsymbol{q}'_{\text{switch}}$
cotangent manifolds
ζ
σ
tangent manifolds
Riemannian space robust margins
$\delta\sigma$
$\delta\sigma$
$\delta\zeta$
$\delta\zeta$
$\boldsymbol{\xi}'(\zeta_0)$
$\boldsymbol{\xi}'(\zeta_{\text{final}})$
$\boldsymbol{q}'_{\text{final}}$
$\mathcal{B}_{\epsilon_2}(\boldsymbol{q}'_{\text{final}})$
mode p_2
$\dot{\mathbf{x}}$ [m/s]
$\mathcal{B}_{\epsilon_1}(\boldsymbol{q}'_{\text{initial}})$
$\boldsymbol{q}'_{\text{initial}}$
mode p_1
x [m]

Figure 8.3: One walking step with robustness margin sets. The horse shoe shape of the robustness margin sets is the result of mapping from phase space to Euclidean space. The robustness margins are shown in the upper right box. The green dot in a state trajectory is the point where mode switching takes place.

Recall that the high-level planner chooses keyframe states from the set $\mathcal{Q}$ defined in (8.16). These keyframe states represent the cells that are obtained by partitioning the robustness margin sets defined in Definition 8.4. In our case, the ordinary locomotion behaviors (i.e., walk and brachiation) comprise 9 keyframe states, respectively while the special locomotion behaviors (i.e., stop, hop and slide) comprise 3 keyframe states, respectively. The goal of mode-transition control synthesis is to determine the possible transitions between these keyframe states. The construction of the set of possible transitions is shown in Figure 8.4.

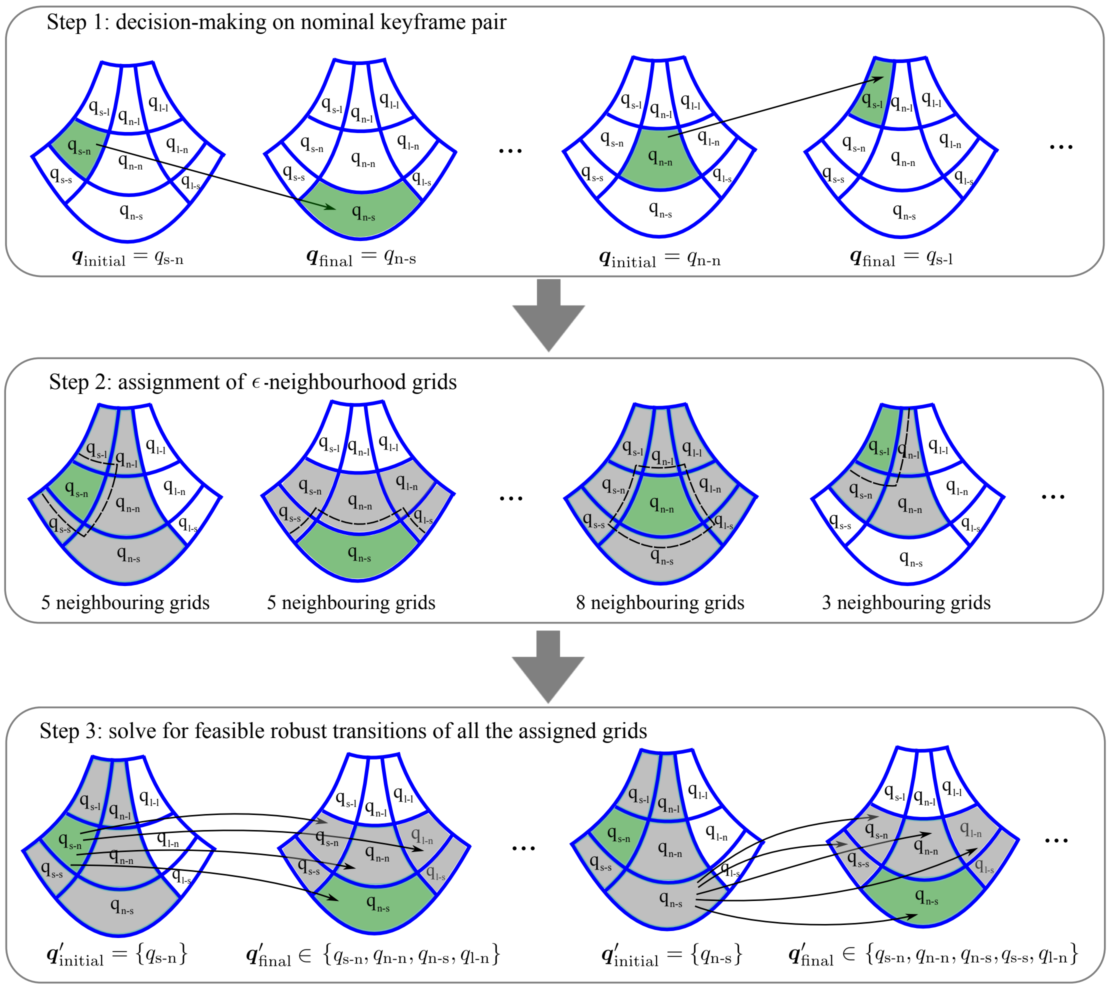

Step 1: decision-making on nominal keyframe pair
$q_{\text{initial}} = q_{\text{s-n}}$
$q_{\text{final}} = q_{\text{n-s}}$
$q_{\text{initial}} = q_{\text{n-n}}$
$q_{\text{final}} = q_{\text{s-l}}$
Step 2: assignment of ϵ-neighbourhood grids
5 neighbouring grids
5 neighbouring grids
8 neighbouring grids
3 neighbouring grids
Step 3: solve for feasible robust transitions of all the assigned grids
$q'_{\text{initial}} = \{q_{\text{s-n}}\}$
$q'_{\text{final}} \in \{q_{\text{s-n}}, q_{\text{n-n}}, q_{\text{n-s}}, q_{\text{l-n}}\}$
$q'_{\text{initial}} = \{q_{\text{n-s}}\}$
$q'_{\text{final}} \in \{q_{\text{s-n}}, q_{\text{n-n}}, q_{\text{n-s}}, q_{\text{s-s}}, q_{\text{l-n}}\}$

Figure 8.4: Construction of a library of possible robust keyframe transitions. The set $\mathcal{Q}$ of keyframe states is obtained by discritizing some neighborhood in the state space Ξ around nominal setpoints. The mode-transition control synthesis verifies the possibility of the transitions between these keyframe states and generate corresponding mode-transition control strategy.

8.2.2 Mode-Transition Control Synthesis for One Walking Step

The goal of mode-transition control synthesis for one walking step is to solve the closed-loop phase-space trajectories starting from the initial robustness margin set $\mathcal{B}_{\epsilon_1}(q_1)$ and reaching the final robustness margin set $\mathcal{B}_{\epsilon_2}(q_2)$ as defined in Definition 8.4 while switch from a locomotion mode p_1 to mode p_2. It is fair to consider one walking step as two sequential semisteps with the first semistep in mode p_1 and second in mode p_2.

To complete the switching between two locomotion modes in one walking step, as shown in Figure 8.3, the region Ξ_{inter} where the switching happens has to be determined. For the first semistep, the region Ξ_{inter} can be treated as the target set that robot state is expected to reach in finite time, and the final robustness margin set $\mathcal{B}_{\epsilon_2}(q_2)$ is the target set to reach in the second semistep. Hence, region Ξ_{inter} must lie in the overlap of the winning sets for both semisteps with respect to reachability specifications.

Additionally, we assume that the function f_p (for all $p \in \mathcal{M}$) in (8.6) is of a particular form:

$$f_p(\boldsymbol{\xi}, \boldsymbol{u}, \boldsymbol{d}) = g_p(\boldsymbol{\xi}) + h_p(\boldsymbol{\xi})\boldsymbol{u} + \boldsymbol{d},\ \forall p \in \mathcal{M}, \tag{8.20}$$

where g_p is Lipschitz continuous and h_p is bounded on Ξ for all $p \in \mathcal{M}$.

Consider one walking step transiting from keyframe state q_1 to q_2 with robustness margins ϵ_1 and ϵ_2, respectively. The locomotion mode has to switch from p_1 to p_2. Suppose that $\Xi_1 \subseteq \Xi$ and $\Xi_2 \subseteq \Xi$ are two local regions where the first and second semistep takes place, respectively. Based on the above discussion, we propose the following *two-semistep reachability control synthesis* for a mode transition:

(i) Perform reachability control synthesis with precision ε_2 (see algorithm (4.19) with (5.2) as the implementation of $\widehat{\text{Pre}}$) for the second semistep (under mode p_2) in the state space Ξ_2. The reachability formula is $\varphi_{r2} = \Diamond G_2$, where $L^{-1}(G) = \mathcal{B}_{\epsilon_2}(q_2)$ is the target set.

(ii) Determine the intermediate region Ξ_{inter} by

$$\Xi_{\text{inter}} = \left\{\boldsymbol{\xi} : \boldsymbol{\xi} \in \text{Win}_{f_{p_2}}(\varphi_{r2}) \wedge \|\mathcal{Z}_{p_1,\sigma}(\boldsymbol{\xi}) - \mathcal{Z}_{p_1,\sigma}(q_1)\|_{\infty} \leq \delta\sigma_{\epsilon_1}\right\} \tag{8.21}$$

(iii) Perform reachability control synthesis with precision ε_1 for the first semistep (under mode p_1) in the state space Ξ_2. The reachability specification is $\varphi_{r1} = \Diamond G_1$ with $L^{-1}(G_1) = \Xi_{\text{inter}}$.

Since the winning set $\text{Win}_{f_{p_1}}(\varphi_{r1})$ of the first semistep is unknown before the determination of Ξ_{inter} and by definition $\Xi_{\text{inter}} \subseteq \text{Win}_{f_{p_1}}(\varphi_{r1})$, the set Ξ_{inter} can be defined as the intersection of the winning set $\mathrm{Win}_{f_{p_2}}(\varphi_{r2})$ and the tube centered at nominal state trajectory from the initial keyframe state q_1 bounded by the robustness margin $\delta\sigma_{\epsilon_1}$, i.e., (8.21). To make sure the intersection is not always empty we need to choose Ξ_1 and Ξ_2 such that $\Xi_1 \cap \Xi_2 \neq \emptyset$. We write the intermediate set as $\Xi_{\text{inter}}((p_1, p_2), (q_1, q_2))$ because Ξ_{inter} is dependent on the given keyframe states q_1, q_2 and locomotion modes p_1, p_2.

The control strategy generated from the reachability control synthesis for two semisteps are $\kappa_1 : \Xi_1 \to 2^{\mathbb{U}}$ and $\kappa_2 : \Xi_2 \to 2^{\mathbb{U}}$. The control strategy for one walking step can be constructed as

$$\kappa((p_1, p_2), (q_1, q_2), \boldsymbol{\xi}) = \begin{cases} \kappa_{p_1}(\boldsymbol{\xi}) & \boldsymbol{\xi} \in \Xi_1 \setminus \Xi_{\text{inter}}((p_1, p_2), (q_1, q_2)), \\ \kappa_{p_2}(\boldsymbol{\xi}) & \boldsymbol{\xi} \in \Xi_{\text{inter}}((p_1, p_2), (q_1, q_2)). \end{cases}$$

8.3 Simulation Results

8.3.1 Evaluation of Mode-Transition Control Strategy

Case I Let us first consider the transition between two PIPM modes, i.e., $p_1 = p_2 = \text{PIPM}$. For the sake of simplicity, the PIPM dynamics in (8.4) is reformulated as

$$\begin{bmatrix} \dot{x}(\zeta) \\ \dot{v}_x(\zeta) \end{bmatrix} = \begin{bmatrix} v_x(\zeta) \\ \omega^2_{\text{PIPM}}(x(\zeta) - x_{\text{foot}}) \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{8.22}$$

by assuming $(\tau_x, \tau_y) = \mathbf{0}$ and x_{foot} is a predefined constant. The continuous control input $\omega_{\text{PIPM}} \in [\bar{\omega} - \delta\omega, \bar{\omega} + \delta\omega]$, where $\bar{\omega}$ is the nominal control input and $\delta\omega$ is a predefined bound. The disturbance $\boldsymbol{d} = (d_1, d_2)$ satisfies $d_{1,2} \in D_r$, where $D_r \subseteq \mathbb{R}^2$ is a bounded set. Hence (8.22) satisfies (8.20).

Suppose that the high-level planner generates the parameters $x_{\text{foot},1}, x_{\text{foot},2}$ and $\bar{\omega}$ and two nominal keyframe states. Let $\delta\zeta = 2\text{ms}$ be the sampling time. The setting of the mode transition problem for the considered one walking step is given in Table 8.1.

In this example, we use a precision $\varepsilon = (0.005\text{m}, 0.005\text{m/s})$ and sample the control space with a granularity $\mu = 0.02\text{rad/s}$. Given the setting above, we perform the two-semistep reachability control synthesis. The computed winning sets are shown in Figure 8.5a. As the result shows, the one-walking step reachability is realizable as long as the winning set overlaps (at least partially) the initial and final robustness margin sets. Five simulated trajectories under randomly-sampled bounded disturbances are shown as the black lines. The blue trajectory represents a trial suffering a large disturbance, i.e., a velocity jump in the phase-space. Figure 8.5b

Table 8.1: Parameters of the PIPM-PIPM mode transition. q_1 and q_2 are the initial and final keyframe states, respectively.

Parameters	**Values**	**Parameters**	**Values**
q_1	$(0\text{m}, 0.5\text{m/s})$	q_2	$(0.5\text{m}, 0.6\text{m/s})$
$\delta\zeta_{\epsilon_1}$	0.05	$\delta\sigma_{\epsilon_1}$	0.002
$\delta\zeta_{\epsilon_1}$	0.05	$\delta\sigma_{\epsilon_2}$	0.006
modes	PIPM $\rightarrow$ PIPM	D_r	$(0.05\text{m}, 0.1\text{m/s})$
Ξ_p	$[-0.1\text{m}, 0.7\text{m}] \times [0.1\text{m/s}, 1.2\text{m/s}]$	$\mathbb{U}_{\text{ows}}$	$[2\text{rad/s}, 4\text{rad/s}]$

shows the change of the winning set under different levels of the disturbance. The winning set shrinks as the disturbance set increases because the synthesized controller needs to reach the goal robust set against a larger set of disturbances.

Case II Consider another locomotion mode transition from the PIPM to PPM. Similarly, we can simplify (8.5) to

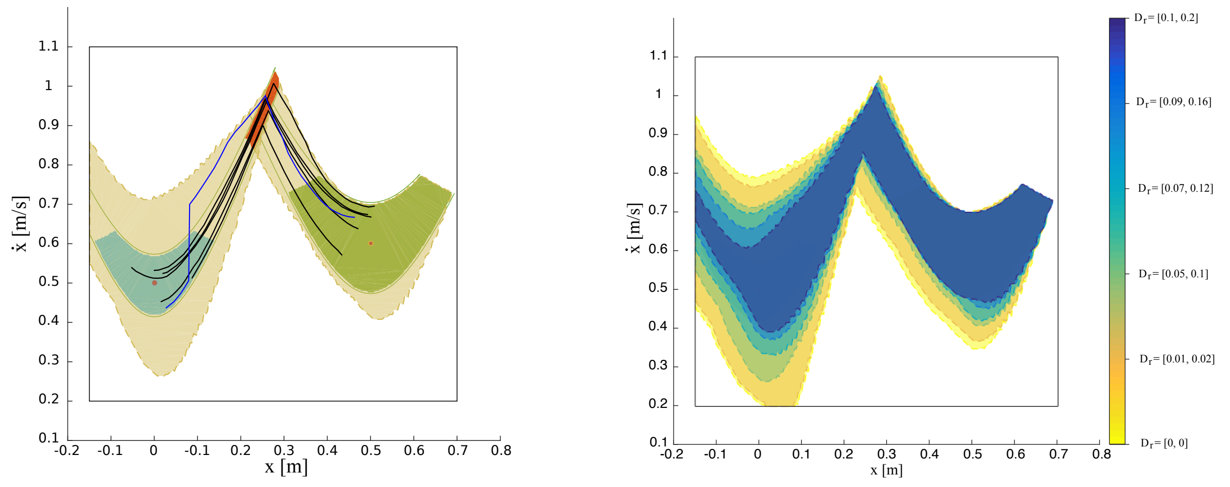
$$\begin{bmatrix} \dot{x}(\zeta) \\ \dot{v}_x(\zeta) \end{bmatrix} = \begin{bmatrix} v_x(\zeta) \\ -\omega_{\text{PPM}}^2(x(\zeta) - x_{\text{hand}}) \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{8.23}$$

with the assumption of $\tau_x = \tau_y = 0$ and a predefined hand contact position x_{hand}. Other parameters are defined in Table 8.2.

Table 8.2: Parameters of the PIPM-PPM mode transition. q_1 and q_2 are the initial and final keyframe states, respectively.

Parameters	**Values**	**Parameters**	**Values**
q_1	$(0\text{m}, 0.5\text{m/s})$	q_2	$(0.6\text{m}, 1.7\text{m/s})$
$\delta\sigma_{\epsilon_1}$	0.002	$\delta\zeta_{\epsilon_1}$	0.05
$\delta\sigma_{\epsilon_2}$	0.06	$\delta\zeta_{\epsilon_2}$	0.005
modes	PIPM $\rightarrow$ PPM	D_r	$(0.15\text{m}, 0.3\text{m/s})$
Ξ_p	$[-0.1\text{m}, 0.7\text{m}] \times [0.1\text{m/s}, 1.8\text{m/s}]$	$\mathbb{U}_{\text{ows}}$	$[2\text{rad/s}, 4\text{rad/s}]$

To evaluate the performance of the control strategy generated by two-semistep reachability control synthesis, we examine the success rate of reaching the goal robustness margin set



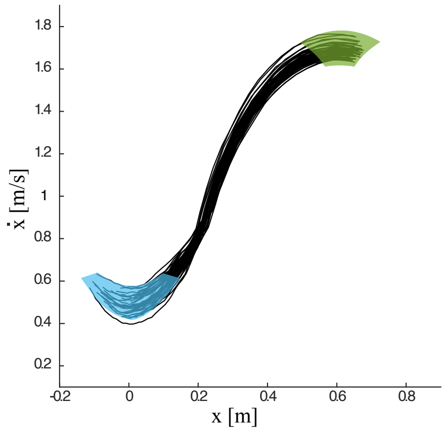
(a) Controlled trajectories.

(b) Winning sets under disturbances.

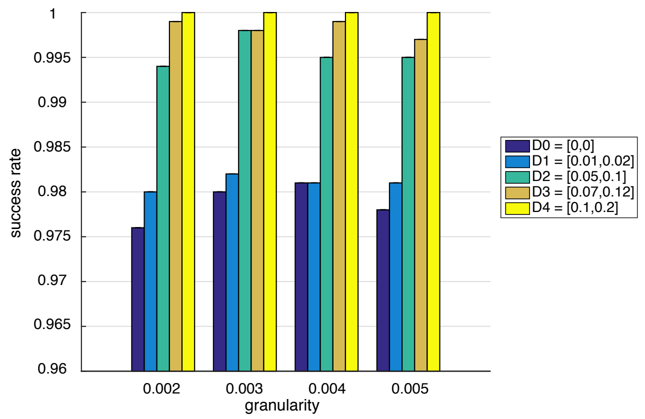
Figure 8.5: Control synthesis results for the walking step from PIPM to PPM. (a) The shaded yellow region represents the winning set of this walking step and the orange region is the intermediate robustness margin set. The black trajectories are 5 simulated closed-loop trajectories. (b) Comparison of winning sets under different levels of disturbance.

through 50 simulation tests under different granularities and bounded disturbances. In Figure 8.6a, each trial is run for the one walking step with PIPM to PPM mode transition. The exerted disturbance in the simulation is the same as the one used in the controller synthesis process. As shown in Figure 8.6a, all the trials reach the final robustness margin successfully.

We evaluate the effect of the control synthesis precision and the magnitude of disturbances used in the controller synthesis process as shown in Figure 8.6b. Figure 8.6b shows 4 sets of simulation results for different control precisions ranging from 0.002 to 0.005. For each set of simulations, the success rate increases as the modeled disturbance in the controller synthesis increases, and it reaches 100% when the modeled disturbance matches the actual disturbance D_r used in the simulation. If we compare the results for different control synthesis precisions under a same disturbance D_i ($i = 0, 1, 2, 3, 4$), the success rate almost remains the same. This is because we use the same disturbance for simulation and control analysis. In addition, it can be observed that the success rates for all the synthesized controllers are greater than 97%, even in the case no disturbance is considered in the controller synthesis. Moreover, under the same disturbance D_r, the nominal phase-space planner with a fixed open-loop control input only achieves a success rate of 29%. This huge discrepancy in success rate clearly shows the advantage of using the proposed method in the middle-layer of control synthesis within the planning framework.



(a) 50 simulated trajectories.



(b) Histogram of success rates.

Figure 8.6: Performance evaluation result for PIPM to PPM walking step. (a) All the 50 simulation trails can reach the goal robustness margin set successfully. (b) 1000 trials are run for each case with a specific precision and a bounded disturbance $D_r = (0.1\text{m}, 0.2\text{m/s})$.

8.3.2 Multi-Step Locomotion Transition

Now we simulate the closed-loop multi-step given the mode switching sequence generated by the high-level planner:

$$\text{PIPM} \rightarrow \text{PIPM} \rightarrow \text{PPM} \rightarrow \text{PIPM} \rightarrow \text{MCM} \rightarrow \text{PIPM} \rightarrow \text{PIPM}$$

To enable the initial and final keyframe robustness margin sets to cover a sufficiently larger phase space, we extend the default 3×3 keyframe grid to a 5×5 keyframe grid for each mode. This allows the mode-transition control strategy to be applicable to a larger set of keyframe states. For each locomotion mode transition, we synthesize all the possible control strategies that reach the final keyframe robustness margin set under a bounded disturbance. We enumerate all the combinations of the allowable locomotion mode pairs and generate all the reachability control policies offline. These controllers are saved as a control library and are executed at runtime according to the high-level decision and measured states under bounded disturbances.

Parameters used in this simulation as follows. The controller synthesis and execution process use the same disturbance bound $D_r = (0.05\text{m}, 0.1\text{m/s})$. The full state space is $\Xi_{\text{full}} = [-0.2\text{m}, 3.8\text{m}] \times [0.2\text{m/s}, 1.9\text{m/s}]$. The local state space of each walking step is chosen so that it is sufficiently large to cover the space around the two keyframe states. A time step $\delta\zeta = 0.02\text{ms}$ is used for the abstraction construction of each walking step. The control inputs for PIPM, PPM and MCM satisfy $\omega_{\text{PIPM}} \in [2, 4]$, $\omega_{\text{PPM}} \in [2, 4]$ and $\omega_{\text{MCM}} \in [1, 3]$. We obtain the sets of sampled control values by a granularity of 0.02. The robustness margins of the phase space manifolds are $\delta\sigma_{\text{PIPM}} = 0.002$, $\delta\zeta_{\text{PIPM}} = 0.002$; $\delta\sigma_{\text{PPM}} = 0.04$, $\delta\zeta_{\text{PPM}} = 0.003$; $\delta\sigma_{\text{MCM}} = 0.15$, $\delta\zeta_{\text{MCM}} = 0.9 \times 10^{-5}$.

We perform the two-semistep reachability control synthesis for each walking step with the precision $(0.003\text{m}, 0.003\text{m/s})$. The computational time is around 30s by average for synthesizing a reachability controller corresponding to each keyframe pair. Since we run 625 (i.e., 25×25) times of such reachability control synthesis for each walking step, the time of generating all the controller policies is approximately 90 mins for each walking step. In the simulation of these six consecutive walking steps, all the local reachability control strategies are patched together to cover the overall state space. The time for simulating a single closed-loop walking trajectory is around 2s. As the results show in Figure 8.7, we simulate six different trials with different initial conditions, i.e., starting from different initial robustness margin sets. Each locomotion trajectory is guaranteed to reach one of the robustness margin sets at the next walking step via using the reachability controller from the control library.

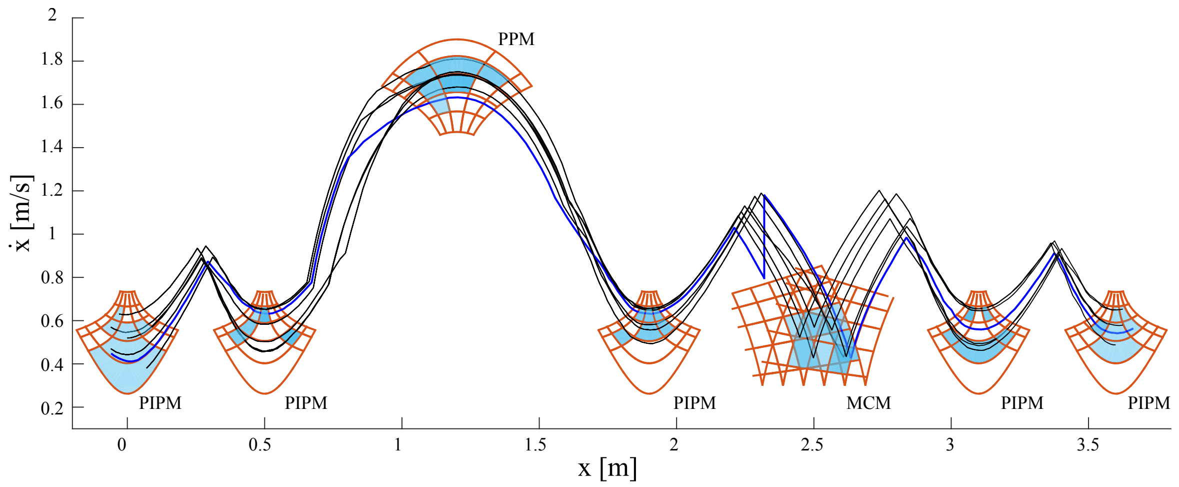


Figure 8.7: The state trajectories of multi-step mode transition under bounded disturbances.

Chapter 9

Conclusions and Future Work

Within the scope of control synthesis for nonlinear systems with temporal logic specifications, the purpose of this research is to understand and bridge (if possible) the gap between two ways of control synthesis from state-space point of view: the one based on analytical analysis of system dynamics on a continuous state space and the one performed by designing computer algorithms that operate on a discretized state space. To this end, we explored in this thesis the possibility of being sound and complete in LTL control synthesis and introduced a concept of *robust completeness* to capture the property of a control system that can tolerate numerical errors, which is inevitable in set computation of continuous-state systems.

In this chapter, we summarize the main contributions in three aspects and also bring up some related open questions worthwhile for future research.

Theoretical Results for General Dynamics

We showed that LTL control synthesis for general nonlinear dynamics can be made sound and complete by using fixed-point algorithms without assuming any stability properties. Specifically, the main theoretical contributions are:

(i) Based on the assumption that the system dynamics is determined by a continuous function in the system and control input spaces, we proved that the predecessor map Pre is both open and closed. A property that is crucial for proving fixed-point characterizations of winning sets for invariance and reachability specifications is also shown: countable set intersections and unions of decreasing sequence of compact and open sets are distributive for Pre, respectively.

(ii) For basic LTL formulas such as invariance, reachability and reach-and-stay that can formulate regulation problems, we characterized the corresponding winning sets by fixed points (sets) of iterative algorithms, which are based on the computation of predecessors, directly over the continuous state space of the original nonlinear systems. Memoryless control strategies can be constructed during fixed-point algorithms for the computation of winning sets. Such fixed-point characterizations for infinite-state systems are not as straightforward as for finite-state systems, which is commonly used in abstraction-based methods. This is our soundness and completeness result.

(iii) Similar to the basic formulas, we provide a sound and complete algorithm for computing the winning set with respect to a DBA-recognizable LTL formula for nonlinear systems over the infinite state space. As opposed to the control synthesis with respect to basic LTL formulas, the resulting control strategies need finite memories.

(iv) Approximation of predecessors is usually required for a concrete implementation of the proposed algorithms, because the exact computation of predecessors are nontrivial under nonlinear dynamics. We provided sufficient conditions for the approximation of predecessors such that the control synthesis algorithms are sound and robustly complete in the sense that control strategies can be found whenever the specifications can be realized for the system with additional disturbance.

However, there are still some questions have not been answered:

- *For reach-and-stay and DBA-recognizable LTL specifications, the robust completeness is valid based on the assumption that the iterative control synthesis algorithms generate their own fixed points. Can we derive a condition that is easier to check?*
- *Can we extend the method to any LTL formula that can only to translated into NBA? The difficulty lies in the nondeterminism of the NBA. A possible solution is to convert the NBA into a deterministic Rabin automaton (DRA). But how difficult it is to extend the similar idea to solving a Rabin game, which is more complex than a Büchi game?*

Implementation and Efficiency

Practical implementation of the proposed conceptual algorithms is one of the major problems that this research is concerned with. In this aspect, the main contributions are highlighted as follows:

(i) To deal with general nonlinearity, we proposed an interval implementation of the proposed control synthesis algorithms, in which predecessors are approximated by unions of intervals. The approximation procedure is carried out by integrating interval arithmetic in a bisection scheme. Under this scheme, set approximation is refined according to both specifications and system dynamics so that discretization is only performed on the region where necessary. For any given precision, such interval-based algorithms are guaranteed to be finitely terminating.

(ii) We establish the criteria of choosing the precision control parameter in the interval approximation of predecessors so as to satisfy the conditions proposed for basic and general LTL control problems. This shows that the LTL control synthesis can be made sound and robustly complete in practice.

(iii) We extended the sound and robustly complete algorithms for solving LTL control synthesis problems for nonlinear discrete-time systems to sampled-data systems. For this purpose, we rely on bounded approximation of the reachable set of a given initial set after one sampling step. This is achieved by computing Taylor expansion of the system solution over one sampling period based on interval arithmetic. We derived the condition for choosing the order of Taylor expansion in the interval approximation of reachable sets such that the proposed algorithm is sound and robustly complete.

(iv) For the control synthesis with respect to DBA-recognizable LTL formulas, we proposed a pre-processing procedure to reduce the computational complexity. The pre-processing is performed on the graph form of the deterministic Büchi automaton in prior to control synthesis.

(v) To show the effectiveness and efficiency of our method compared with abstraction-based methods in the literature, we analyze the complexities and the performances on benchmarking examples for both approaches. The worst case complexity of our proposed method is similar to the one of abstraction-based methods, but the experimental results show that our method enjoys higher computational efficiency.

Even though we have shown that, by using interval computation, the approximation of winning sets can be lower bounded within an arbitrary precision, the question is still open regarding to the convergence of the proposed method:

Can the inner approximations of the winning sets with respect to LTL specifications, such as invariance, reachability, reach-and-stay, that are commonly used for the control of dynamical systems converge to the real winning sets?

Applications

The third aspect we consider is how well the proposed method applies to solving real-world control problems. Therefore, in this thesis, we studied examples drawn from different practical applications including the voltage regulation problem of boost DC-DC converters, the stabilization problem of an inverted pendulum, the ROA estimation problem, the parallel parking problem, and motion planning problems with respect to different LTL specifications. In particular, we showed that the proposed formal control synthesis algorithms can be used to generate a middle-layer control strategy that synergizes high-level plan and low-level control in solving the reactive locomotion planning problem.

Bibliography

[1] R. Alur., C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229. 1993.

[2] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In *Proceedings of International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 322–335. 1990.

[3] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017.

[4] David Angeli. A lyapunov approach to incremental stability properties. *IEEE Transactions on Automatic Control*, 47(3):410–421, 2002.

[5] Karl Johan Astrom and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, 2008.

[6] Hervé Audren, Joris Vaillant, Abderrahmane Kheddar, Adrien Escande, Kunihiko Kaneko, and Erika Yoshida. Model preview control in multi-contact motion-application to a humanoid robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4030–4035, 2014.

[7] Ebru Aydin Gol, Mircea Lazar, and Calin Belta. Language-guided controller synthesis for linear systems. *IEEE Transactions on Automatic Control*, 59(5):1163–1176, 2014.

[8] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen. *Principles of Model Checking*. MIT press, 2008.

[9] Calin Belta, Antonio Bicchi, Magnus Egerstedt, Emilio Frazzoli, Eric Klavins, and George Pappas. Symbolic planning and control of robot motion. *IEEE Robotics & Automation Magazine*, 14(1):61–70, 2007.

[10] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Formal Methods for Discrete-Time Dynamical Systems*, volume 89. Springer International Publishing, 2017.

[11] Dimitri P. Bertsekas. Infinite-time reachability of state-space regions by using feedback control. *IEEE Transactions on Automatic Control*, 17(5):604–613, 1972.

[12] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume II. 2017.

[13] D.P. Bertsekas and I.B. Rhodes. On the minimax reachability of target sets and target tubes. *Automatica*, 7(2):233–247, 1971.

[14] Amit Bhatia, Matthew R. Maly, Lydia E. Kavraki, and Moshe Y. Vardi. Motion planning with complex goals. *IEEE Robotics & Automation Magazine*, 18(3):55–64, 2011.

[15] A. Bicchi, A. Marigo, and B. Piccoli. On the reachability of quantized control systems. *IEEE Transactions on Automatic Control*, 47(4):546–563, 2002.

[16] F. Blanchini. Minimum-time control for uncertain discrete-time linear systems. In *Proceedings of the 31st IEEE Conference on Decision and Control (CDC)*, pages 2629–2634, 1992.

[17] F. Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[18] Franco Blanchini. Ultimate boundedness control for uncertain discrete-time systems via set-induced lyapunov functions. *IEEE Transactions on Automatic Control*, 39(2):428–433, 1994.

[19] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'Ar. Synthesis of reactive(1) designs. *Journal of Computer and System Sciences*, 78(3):911–938, 2012.

[20] Mireille Broucke. A geometric approach to bisimulation and verification of hybrid systems. In *Proceedings of the 2nd International Workshop on Hybrid Systems: Computation and Control, HSCC'99*, pages 61–75, 1999.

[21] Mireille E. Broucke. Reach control on simplices by continuous state feedback. *SIAM Journal on Control and Optimization (SICON)*, 48(5):3482–3500, 2010.

[22] Mireille E. Broucke and Marcus Ganness. Reach control on simplices by piecewise affine feedback. *SIAM Journal on Control and Optimization (SICON)*, 52(5):3261–3286, 2014.

[23] J. Richard Buchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295, 1969.

[24] P.E. Caines and Y.J. Wei. The hierarchical lattices of a finite machine. *Systems & Control Letters*, 25(4):257–263, 1995.

[25] P.E. Caines and Yuan-Jun Wei. Hierarchical hybrid control systems: a lattice theoretic formulation. *IEEE Transactions on Automatic Control*, 43(4):501–508, 1998.

[26] Xin Chen. *Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models*. PhD thesis, RWTH Aachen University, 2015.

[27] Xin Chen, Erika Abraham, and Sriram Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *Proceedings of Real-Time Systems Symposium*, pages 183–192, 2012.

[28] Xin Chen, Sriram Sankaranarayanan, and Erika Ábrahám. Under-approximate flowpipes for non-linear continuous systems. In *Proceedings of 2014 Formal Methods in Computer-Aided Design (FMCAD)*, pages 59–66, 2014.

[29] Graziano Chesi. Estimating the domain of attraction for non-polynomial systems via LMI optimizations. *Automatica*, 45(6):1536–1541, 2009.

[30] Edmund Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.

[31] Pieter Collins. Optimal semicomputable approximations to reachable and invariant sets. *Theory of Computing Systems*, 41(1):33–48, 2007.

[32] Pieter Collins and Alexandre Goldsztejn. The reach-and-evolve algorithm for reachability analysis of nonlinear dynamical systems. *Theoretical Computer Science*, 223:87–102, 2008.

[33] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[34] Sumanth Dathathri and Richard M Murray. Decomposing GR(1) games with singleton liveness guarantees for efficient synthesis. In *Proceedings of the 56th IEEE Conference on Decision and Control (CDC)*, pages 911–917, 2017.

[35] L. de Alfaro, T.A. Henzinger, and R. Majumdar. From verification to control: dynamic programs for omega-regular objectives. In *Proceedings of 16th Annual IEEE Symposium on Logic in Computer Science*, pages 279–290, 2001.

[36] Luca de Alfaro, Thomas A. Henzinger, and Rupak Majumdar. Symbolic algorithms for infinite-state games. In Kim G. Larsen and Mogens Nielsen, editors, *CONCUR 2001 – Concurrency Theory*, pages 536–550. Springer Berlin Heidelberg, 2001.

[37] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Étienne Renault, and Laurent Xu. Spot 2.0 — a framework for ltl and ω-automata manipulation. In *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16)*, pages 122–129. Springer, 2016.

[38] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st international conference on Software engineering - ICSE '99*, pages 411–420, 1999.

[39] E. Allen Emerson and Chin-Laung Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Proceedings of the 1st Annual IEEE Symposium on Logic in Computer Science*, pages 267–278, 1986.

[40] E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of 32nd Annual Symposium of Foundations of Computer Science*, pages 368–377, 1991.

[41] Georgios E Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

[42] T. Faulwasser, B. Kern, and R. Findeisen. Model predictive path-following for constrained nonlinear systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 28th Chinese Control Conference (CCC)*, pages 8642–8647, 2009.

[43] B.A. Francis and W.M. Wonham. The internal model principle of control theory. *Automatica*, 12(5):457–465, 1976.

[44] Emilio Frazzoli, Munther A. Dahleh, Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, 2005.

[45] Laurent Fribourg and Romain Soulat. *Control of Switching Systems by Invariance Analysis: Application to Power Electronics*. Wiley-ISTE, 2013.

[46] Sicun Gao, Soonho Kong, Wei Chen, and Edmund M. Clarke. Delta-complete analysis for bounded reachability of hybrid systems. *The Computing Research Repository (CoRR)*, abs/1404.7171, 2014.

[47] Paul Gastin and Denis Oddoux. Fast ltl to büchi automata translation. In *Proceedings of the 13th International Conference on Computer-Aided Verification (CAV)*, volume 52, pages 53–65, 2001.

[48] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. *Simple On-the-fly Automatic Verification of Linear Temporal Logic*, pages 3–18. Springer US, Boston, MA, 1996.

[49] Antoine Girard, Gregor Gossler, and Sebti Mouelhi. Safety controller synthesis for incrementally stable switched systems using multiscale symbolic models. *IEEE Transactions on Automatic Control*, 61(6):1537–1549, 2016.

[50] Antoine Girard and George J. Pappas. Approximate bisimulations for constrained linear systems. In *Proceedings of the 44th IEEE Conference on Decision and Control (CDC) and European Control Conference (ECC)*, pages 4700–4705, 2005.

[51] Antoine Girard and George J. Pappas. Approximate bisimulations for nonlinear dynamical systems. In *Proceedings of the 44th IEEE Conference on Decision and Control (CDC) and European Control Conference (ECC)*, pages 684–689, 2005.

[52] Antoine Girard and George J. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5):782–798, 2007.

[53] Antoine Girard, Giordano Pola, and Paulo Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Transactions on Automatic Control*, 55(1):116–126, 2010.

[54] Laurent Granvilliers. On the combination of interval constraint solvers. *Reliable Computing*, 7(6):467–483, 2001.

[55] Per-Olof Gutman and Michael Cwikel. Admissible sets and feedback control for discrete-time linear dynamical systems with bounded controls and states. *IEEE Transactions on Automatic Control*, 31(4):373–376, 1986.

[56] L.C.G.J.M. Habets, P.J. Collins, and J.H. Van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Transactions on Automatic Control*, 51(6):938–948, 2006.

[57] Esfandiar Haghverdi, Paulo Tabuada, and George J. Pappas. Bisimulation relations for dynamical, control, and hybrid systems. *Theoretical Computer Science*, 342(2):229–261, 2005.

[58] Keliang He, Morteza Lahijanian, Lydia E Kavraki, and Moshe Y Vardi. Towards manipulation planning with temporal logic specifications. In *Proceedings of 2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 346–352, 2015.

[59] Mohamed K. Helwa and Peter E. Caines. In-block controllability of affine systems on polytopes. *IEEE Transactions on Automatic Control*, 62(6):2950–2957, 2017.

[60] Thomas A. Henzinger. Hybrid automata with finite bisimulations. In *Proceedings of International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 324–335. 1995.

[61] Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. Multi-layered abstraction-based controller synthesis for continuous-time systems. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control, Part of CPS Week, HSCC'18*, pages 120–129, 2018.

[62] Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. Multi-layered abstraction-based controller synthesis for continuous-time systems. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control, Part of CPS Week, HSCC'18*, pages 120–129, 2018.

[63] Bronislaw Jakubczyk and Eduardo D. Sontag. Controllability of nonlinear discrete-time systems: A lie-algebraic approach. *SIAM Journal on Control and Optimization (SICON)*, 28(1):1–33, 1990.

[64] Zachary Jarvis-Wloszek, Ryan Feeley, Weehong Tan, Kunpeng Sun, and Andrew Packard. Control applications of sum of squares programming. In *Proceedings of the 42nd IEEE Conference on Decision and Control (CDC)*, volume 5, pages 4676–4681, 2003.

[65] Luc Jaulin. *Applied Interval Analysis: with Examples in Parameter and State Estimation, Robust Control and Robotics*, volume 1. Springer Science & Business Media, 2001.

[66] Manuel Mazo Jr., Anna Davitian, and Paulo Tabuada. Pessoa: a tool for embedded controller synthesis. In *Proceedings of the 22nd International Conference on Computer-Aided Verification (CAV)*, pages 566–569, 2010.

[67] Eric C Kerrigan. *Robust Constraint Satisfaction: Invariant Sets and Predictive Control.* PhD thesis, Department of Engineering, University of Cambridge, 2000.

[68] Mahmoud Khaled, Eric S Kim, Murat Arcak, and Majid Zamani. Synthesis of symbolic controllers: A parallelized and sparsity-aware approach. In *Proceedings of the 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2019.

[69] Hassan K Khalil. *Nonlinear Systems.* Prentice Hall, 2002.

[70] Eric S. Kim, Murat Arcak, and Majid Zamani. Constructing control system abstractions from modular components. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control, Part of CPS Week, HSCC'18*, pages 137–146, 2018.

[71] Marius Kloetzer and Calin Belta. Dealing with nondeterminism in symbolic control. In *Proceedings of the 11th International Conference on Hybrid Systems: Computation and Control, Part of CPS Week, HSCC'08*, pages 287–300. 2008.

[72] Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[73] Ilya Kolmanovsky and Elmer G. Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Engineering*, 4(4):317–367, 1998.

[74] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. dreach: δ-reachability analysis for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 200–205, 2015.

[75] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.

[76] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

[77] Shengbo Li, Keqiang Li, Rajesh Rajamani, and Jianqiang Wang. Model predictive multi-objective vehicular adaptive cruise control. *IEEE Transactions on Control Systems Technology*, 19(3):556–566, 2011.

[78] Yinan Li and Jun Liu. Computing maximal invariant sets for switched nonlinear systems. In *Proceedings of 2016 IEEE Conference on Computer-Aided Control System Design (CACSD)*, pages 862–867, 2016.

[79] Yinan Li and Jun Liu. An interval analysis approach to invariance control synthesis for discrete-time switched systems. In *Proceedings of the 55th IEEE Conference on Decision and Control (CDC)*, pages 6388–6394, 2016.

[80] Yinan Li and Jun Liu. Invariance control synthesis for switched nonlinear systems: An interval analysis approach. *IEEE Transactions on Automatic Control*, 63(7):2206–2211, 2018.

[81] Yinan Li and Jun Liu. Robustly complete reach-and-stay control synthesis for switched systems via interval analysis. In *Proceedings of 2018 American Control Conference (ACC'18)*, pages 2350–2355, 2018.

[82] Yinan Li and Jun Liu. Robustly complete synthesis of memoryless controllers for nonlinear systems with reach-and-stay specifications. *arXiv:1802.09082*, 2018.

[83] Yinan Li and Jun Liu. ROCS: A robustly complete control synthesis tool for nonlinear dynamical systems. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control, Part of CPS Week, HSCC'18*, pages 130–135, 2018.

[84] Yinan Li, Jun Liu, and Necmiye Ozay. Computing finite abstractions with robustness margins via local reachable set over-approximation. *IFAC-PapersOnLine*, 48(27):1–6, 2015.

[85] Jun Liu. Robust abstractions for control synthesis: Completeness via robustness for linear-time properties. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, Part of CPS Week, HSCC'17*, pages 101–110, 2017.

[86] Jun Liu, N. Ozay, U. Topcu, and R.M. Murray. Synthesis of reactive switching protocols from temporal logic specifications. *IEEE Transactions on Automatic Control*, 58(7):1771–1785, 2013.

[87] Jun Liu and Necmiye Ozay. Finite abstractions with robustness margins for temporal logic-based control synthesis. *Nonlinear Analysis: Hybrid Systems*, 22:1–15, 2016.

[88] David Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

[89] Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure Applied Logic*, 65(2):149–184, 1993.

[90] Bill Messner and Dawn Tilbury. Inverted pendulum: System modeling. http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling, 2014.

[91] Pierre-Jean Meyer and Dimos V. Dimarogonas. Hierarchical decomposition of ltl synthesis problem for nonlinear control systems. *IEEE Transactions on Automatic Control*, pages 1–1, 2019.

[92] Ramon E Moore. *Interval Analysis*. Prentice-Hall, 1966.

[93] Sebti Mouelhi, Antoine Girard, and Gregor Gössler. CoSyMA: a tool for controller synthesis using multi-scale abstractions. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control, Part of CPS Week, HSCC'13*, pages 83–88, 2013.

[94] Nedialko S. Nedialkov, Kenneth R. Jackson, and John D. Pryce. An effective high-order interval method for validating existence and uniqueness of the solution of an ivp for an ode. *Reliable Computing*, 7(6):449–465, 2001.

[95] N.S. Nedialkov, K.R. Jackson, and G.F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999.

[96] Petter Nilsson, Omar Hussien, Ayca Balkan, Yuxiao Chen, Aaron D Ames, Jessy W Grizzle, Necmiye Ozay, Huei Peng, and Paulo Tabuada. Correct-by-construction adaptive cruise control: Two approaches. *IEEE Transactions on Control System Technology*, 24(4):1294–1307, 2016.

[97] Petter Nilsson, Omar Hussien, Yuxiao Chen, and et al. Preliminary results on correct-by-construction control software synthesis for adaptive cruise control. In *Proceedings of the 53rd IEEE Conference on Decision and Control (CDC)*, 2014.

[98] Petter Nilsson, Necmiye Ozay, and Jun Liu. Augmented finite transition systems as abstractions for control synthesis. *Discrete Event Dynamic Systems*, 27(2):301–340, 2017.

[99] Melkior Ornik, Miad Moarref, and Mireille E. Broucke. An automated parallel parking strategy using reach control theory. *IFAC-PapersOnLine*, 50(1):9089–9094, 2017.

[100] George J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, 2003.

[101] George J. Pappas and Shankar Sastry. Towards continuous abstractions of dynamical and control systems. In Panos J. Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems IV*, pages 329–341. Springer Berlin Heidelberg, 1997.

[102] Gilberto Pin and Thomas Parisini. On the robustness of nominal nonlinear minimum-time control and extension to non-robustly controllable target sets. *IEEE Transactions on Automatic Control*, 59(4):863–875, 2014.

[103] Erion Plaku and Sertac Karaman. Motion planning with temporal-logic specifications: Progress and challenges. *AI Communications*, 29(1):151–162, 2015.

[104] Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In *Automata, Languages and Programming. ICALP 1989. Lecture Notes in Computer Science*, volume 372, pages 652–671. 1989.

[105] Giordano Pola, Antoine Girard, and Paulo Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[106] Giordano Pola and Paulo Tabuada. Symbolic models for nonlinear control systems: Alternating approximate bisimulations. *SIAM Journal on Control and Optimization (SICON)*, 48(2):719–733, 2009.

[107] S.Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[108] S.V. Rakovic, E.C. Kerrigan, D.Q. Mayne, and J. Lygeros. Reachability analysis of discrete-time systems with disturbances. *IEEE Transactions on Automatic Control*, 51(4):546–561, 2006.

[109] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization (SICON)*, 25(1):206–230, 1987.

[110] Nacim Ramdani and Nedialko S. Nedialkov. Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint-propagation techniques. *Nonlinear Analysis: Hybrid Systems*, 5(2):149–162, 2011.

[111] Stefan Ratschan. Safety verification of non-linear hybrid systems is quasi-decidable. *Formal Methods in System Design*, 44(1):71–90, 2014.

[112] Stefan Ratschan and Zhikun She. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM Transactions on Embedded Computing Systems*, 6(1):8, 2007.

[113] Gunther Reissig, Alexander Weber, and Matthias Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62(4):1781 – 1796, 2017.

[114] Michael Rinehart, Munther A. Dahleh, Dennis Reed, and Ilya Kolmanovsky. Suboptimal control of switched systems with an application to the disc engine. *IEEE Transactions on Control System Technology*, 16(2):189–201, 2008.

[115] R Tyrrell Rockafellar and Roger J-B Wets. *Variational Analysis*. Springer, 2009.

[116] Matthias Rungger, Manuel Mazo Jr., and Paulo Tabuada. Specification-guided controller synthesis for linear systems and safe linear-time temporal logic. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control, Part of CPS Week, HSCC'13*, pages 333–342, 2013.

[117] Matthias Rungger and Paulo Tabuada. Computing robust controlled invariant sets of linear systems. *IEEE Transactions on Automatic Control*, 62(7):3665–3670, 2017.

[118] Matthias Rungger and Majid Zamani. SCOTS: a tool for the synthesis of symbolic controllers. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, Part of CPS Week, HSCC'16*, pages 99–104, 2016.

[119] Paulo Tabuada. An approximate simulation approach to symbolic control. *IEEE Transactions on Automatic Control*, 53(6):1406–1418, 2008.

[120] Paulo Tabuada. Controller synthesis for bisimulation equivalence. *System & Control Letters*, 57(6):443–452, 2008.

[121] Paulo Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer Science & Business Media, 2009.

[122] Paulo Tabuada and George J. Pappas. Model checking LTL over controllable linear systems is decidable. In *Proceedings of the 6th International Conference on Hybrid Systems: Computation and Control, Part of CPS Week, HSCC'03*, pages 498–513, 2003.

[123] Paulo Tabuada and George J Pappas. Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 51(12):1862–1877, 2006.

[124] Wolfgang Thomas. Infinite games and verification. In *Proceedings of the 14th International Conference on Computer-Aided Verification (CAV)*, Lecture Notes in Computer Science, pages 58–65. Springer Berlin Heidelberg, 2002.

[125] B. Tibken and O. Hachicho. Estimation of the domain of attraction for polynomial systems using multidimensional grids. In *Proceedings of the 39th IEEE Conference on Decision and Control (CDC)*, volume 4, pages 3870–3874, 2000.

[126] Ufuk Topcu, Andrew Packard, and Peter Seiler. Local stability analysis using simulations and sum-of-squares programming. *Automatica*, 44(10):2669–2675, 2008.

[127] Ufuk Topcu, Andrew K. Packard, Peter Seiler, and Gary J. Balas. Robust region of attraction estimation. *IEEE Transactions on Automatic Control*, 55(1):137–142, 2010.

[128] Marc Toussaint, Kelsey R. Allen, Kevin A. Smith, and Joshua B. Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning - extended abtract. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 6231–6235, 2019.

[129] Harry Trentelman, Anton A. Soorvogel, and Malo Hautus. *Control Theory for Linear Systems*. Springer-Verlag London, 2001.

[130] Harry L. Trentelman, Anton A. Stoorvogel, and Malo Hautus. *Tracking and regulation*, pages 195–209. Springer London, 2001.

[131] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency. Lecture Notes in Computer Science*, volume 1043, pages 238–266. Springer Berlin Heidelberg, 1996.

[132] Cristian Ioan Vasile and Calin Belta. Reactive sampling-based temporal logic path planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 53, pages 4310–4315, 2014.

[133] Jian Wan, Josep Vehí, Ningsu Luo, and Pau Herrero. Control of constrained nonlinear uncertain discrete-time systems via robust controllable sets: a modal interval analysis approach. *ESAIM: Control, Optimisation and Calculus of Variations (COCV)*, 15(1):189–204, 2009.

[134] Peter Wieland and Frank Allgöwer. Constructive safety using control barrier functions. *IFAC Proceeding*, 40(12):462–467, 2007.

[135] Eric M. Wolff, Ufuk Topcu, and Richard M. Murray. Automaton-guided controller synthesis for nonlinear systems with temporal logic. In *Proceedings of 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4332–4339, 2013.

[136] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M. Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, 2012.

[137] Tichakorn Wongpiromsarn, Ufuk Topcu, Necmiye Ozay, Huan Xu, and Richard M. Murray. TuLiP: a software toolbox for receding horizon temporal logic planning. In *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control, Part of CPS Week, HSCC'11*, pages 313–314, 2011.

[138] Bai Xue, Zhikun She, and Arvind Easwaran. Under-approximating backward reachable sets by polytopes. In *Proceedings of the 28th International Conference on Computer-Aided Verification (CAV)*, pages 457–476, 2016.

[139] Majid Zamani, Giordano Pola, Manuel Mazo Jr., and Paulo Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, 57(7):1804–1809, 2012.

[140] Ye Zhao, Benito R Fernandez, and Luis Sentis. Robust optimal planning and control of non-periodic bipedal locomotion with a centroidal momentum model. *The International Journal of Robotics Research*, 36(11):1211–1242, 2017.

[141] Ye Zhao, Yinan Li, Luis Sentis, Ufuk Topcu, and Jun Liu. Reactive task and motion planning for robust whole-body dynamic locomotion in constrained environments. *arXiv:1811.04333*, 2018.

[142] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.

APPENDICES

Appendix A

ROCS: A Tool for Robustly Complete Control Synthesis

This appendix presents ROCS, which is an algorithmic control synthesis tool for general discrete-time or sampled-data systems. It is based on the theoretical results in this thesis. At the core of ROCS is the interval branch-and-bound scheme with a precision control parameter that reflects the robustness of the realizability of the specification.

As opposed to other formal control synthesis tools [118, 93, 66, 137] the distinct features of ROCS include:

- Synthesis is performed *directly on the continuous state space*, without having to abstract the system into a finite-state model.
- Synthesis algorithms are *sound and robustly complete* in the sense that control strategies can be found whenever the given specification is robustly realizable [80]. This is similar to what dReach [74] offers for bounded reachability analysis, but in the context of control synthesis.
- Parameter setting is *simple and flexible.* ROCS generates partition-based control strategies, where the partitions are adaptively refined with respect to both the dynamics and given specifications. The precision of a partition is controlled by a single parameter, which can be easily configured by the user. By setting different values of this parameter, ROCS can be used for robustness analysis. Furthermore, ROCS allows one to use multiple and variable precisions to expedite computation.
- It currently supports a wider class of LTL specifications for the control synthesis purpose.

The tool is implemented as a C++ library providing algorithms, as well as their interface to matlab, for the proposed specification-guided control synthesis via interval computation. The source code and examples can be downloaded from: https://git.uwaterloo.ca/hybrid-systems-lab/rocs.

The description of the initial version of ROCS can be found in [83], and here we present ROCS in its current format.

A.1 Design and Structure

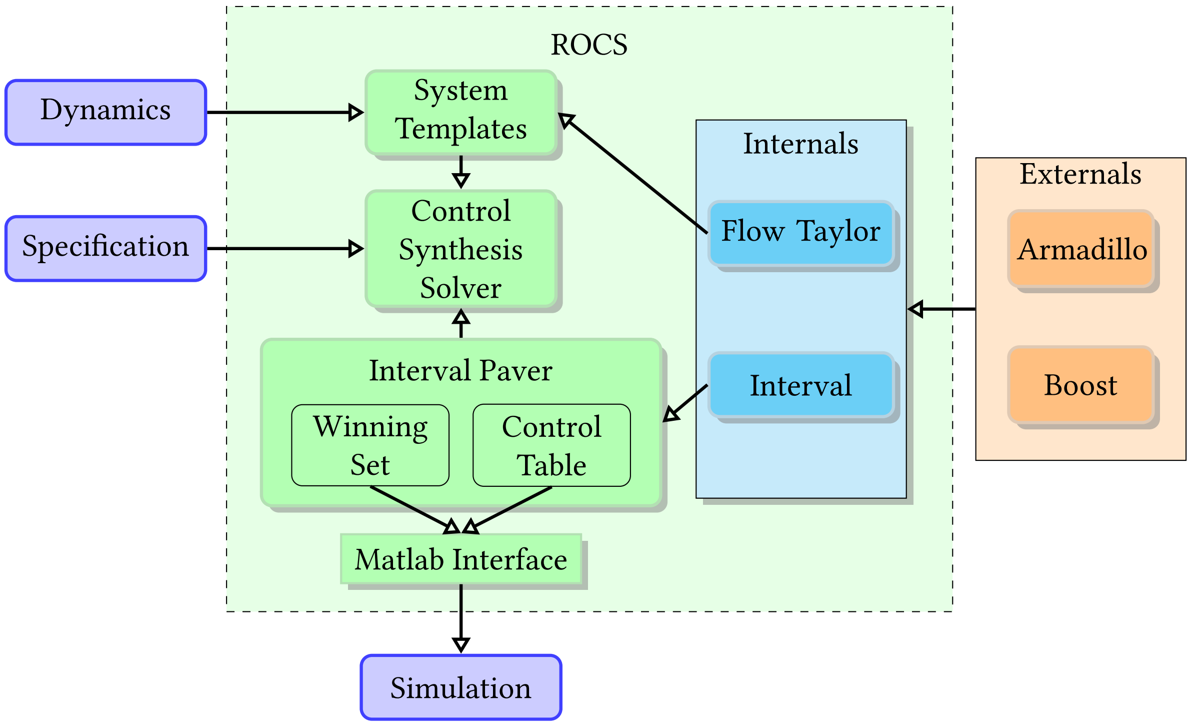


Figure A.1: The architecture of ROCS.

Figure A.1 shows the current architecture of ROCS. The user input includes *system dynamics* and *specifications* of a specific control synthesis problem. ROCS is composed of 6 core modules:

- `System Templates` defines different system types including discrete-time systems (2.2), discrete-time switched systems (3.6), and continuous-time systems (7.1). It mounts the user input `Dynamics` to the corresponding system template.

- `Flow Taylor` works as a wrapper of the continuous-time dynamics (i.e., ODE) and produces reachable sets of the ODE in one sampling time.

- `Control Synthesis Solver` is the core module that integrates different control synthesis algorithms, which operate on `Interval Paver`. It accesses to system dynamics and `Specification` specified by the user.

- `Interval Paver` is a binary tree data structure that represent the interval partition of the system state space or $\mathcal{S}$-domains in Chapter 6. The information of `Winning Set` and `Control Table`, which is the form of control strategy, is also contained in this structure.

- `Matlab Interface` is designed to convert the data representing the winning set and control strategy to Matlab data format.

- `Interval` is the basic data structure that computation relies on.

Two external libraries `Armadillo`[1] and `Boost`[2] are used in the design of the internal modules `Flow Taylor` and `Interval` for handling linear operations and boolean valued vectors, respectively.

A.2 Usage

To solve a control synthesis problem using ROCS, the user needs to provide:

- an interval inclusion function of the discrete-time system or the ODEs of the continuous-time system to be controlled, and

- a main program that defines the control problem and executes control synthesis.

To manage a control synthesis process, the user has to write a `main` function for each control problem. Figure A.2 is a sample `main` function coded with the invariance control synthesis workflow of a boost DC-DC converter in Section 5.5.1.

First, in the `main` function, the state and input spaces are specified by their lower and upper bounds. Next, after loading the customized dynamics, a control problem will be instantiated as

[1] http://arma.sourceforge.net/
[2] https://www.boost.org/

,

```
#include "dcdc.hpp"
  int main()
{
    /* set the state space */
    double xlb[] = {-2, 0.70};
    double xub[] = {2, 1.50};

    /* define the control system */
    rocs::DTSwSys<dcde> dcdcInv("dcdc", tau, dcde::n, dcde::m);
    dcdcInv.init_workspace(xlb, xub);

    /* set the specifications */
    double glb[] = {1.15, 1.09};
    double gub[] = {1.55, 1.17};

    /* solve the problem */
    rocs::CSolver solver(&dcdcInv);
    solver.init(rocs::GOAL, glb, gub);
    solver.init_goal_area();

    solver.invariance_control(&dcdcInv, 0.001, rocs::RELMAXG);
    solver.print_controller_info();

    /* save the problem data and the solution */
    rocs::matWriter wtr("data_dcdcInv.mat");
    wtr.open();
    wtr.write_problem_setting(dcdcInv, solver);
    wtr.write_sptree_controller(solver);
    wtr.close();

    return 0;
}
```

Figure A.2: A sample `main` function for the invariance control synthesis of a boost DC-DC converter. A partition precision of 0.001 and the relative bisection type `RELMAXG` are used when calling `invariance_control`, which is a member function of CSolver.

specified by the user. For example, a switched system template `DTSwSys`, which is defined in the file `system.hpp`, is used for the boost DC-DC converter to create the control problem `dcdcInv` in Figure A.2. The dynamics of the converter is provided in another file `dcdc.hpp` (shown in Figure A.3). After the target sets are defined, a solver (a `CSolver` object) is created to attach to the problem and gradually refines the the partition (an `interval_paver` object) of the system state space under the corresponding control synthesis algorithm. Finally when the iteration terminates, in order to test and visualize the control performance, the user can write the entire case information, including system and specification setups, and control strategy to `.mat` files. Utility functions for Matlab display are provided under the `matlab` folder of the ROCS package.

To perform control synthesis, the user chooses an algorithm provided by the `CSolver` class according to the control objective. For example, in Figure A.2, the `invariance_control` algorithm is used. Other available algorithms include `reachability_control`, `buchi`, `cobuchi`, and Algorithm 6.1 and 6.2.

These algorithms take three types of arguments:

- a precision parameter for control synthesis,
- a bisection type (choosing from `RELMAXW`, `RELMAXG` or `ABSMAX`), and
- a boolean indicating variable or fixed precision.

The precision control parameter determines the precision of the resulting partition and is related to the robustness margin of the specification (see [80, 81]). The bisection type indicates whether to subdivide an interval along the dimension of the greatest absolute or relative width to the state space/target area.

For specifications related to reachability, it is usually more efficient to use a variable precision (by setting the boolean argument to be `true`). For detailed descriptions and usage of the parameters of each algorithm, the user may refer to the documentation of the `CSolver` class.

In the package, we provide complete sets of examples under the subfolder *example* of the repository, including interval inclusion functions, main program files, and files for Matlab simulation, to show how to use ROCS for control synthesis. These examples have been illustrated in Chapter 5, 6, and 8:

- *dcdc*: invariance and reach-and-stay control of a boost DC-DC converter.
- *car*: motion planning problems considered in Section 5.5.3 and Examples 6.4 and 6.5 in Chapter 6, and the parallel parking problem in Section 5.5.2.

```
/* Parameters of the model */
const double tau = 0.5;
const double xc = 70.0;
const double xl = 3.0;
const double rc = 0.005;
const double rl = 0.05;
const double r0 = 1.0;
const double vs = 1.0;

arma::mat I = arma::eye<arma::mat>(2, 2);
arma::vec b = {vs/xl, 0};
arma::mat A1 = {{-rl/xl, 0}, {0, -1/(xc*(rc+r0))}};
arma::mat F1 = arma::expmat(A1 * tau);
arma::vec g1 = arma::inv(A1) * (F1 - I) * b;
arma::mat A2 = { {(-1/xl)*(rl+r0*rc/(r0+rc)),(-1/xl)*(r0/(r0+rc))}, {(1/xc)
    *(r0/(r0+rc)),(-1/xc)*(1/(r0+rc))} };
arma::mat F2 = arma::expmat(A2 * tau);
arma::vec g2 = arma::inv(A2) * (F2 - I) * b;

/* Discrete-time dynamics of the boost DCDC converter */
struct dcde {
    static const int n = 2;  // state dimension
    static const int m = 2;  // number of modes

    /**
     * Constructors:
     * real-valued (arma::vec) and interval-valued (rocs::ivec)
     * @param[out] y the next state after the sampling time.
     * @param[in] x the current state.
     * @param[in] m the mode.
     */
    dcde(rocs::ivec &y, const rocs::ivec &x, const int m) {
    switch (m) {
    case 1:
        y = linmap(F1, g1, x);
        break;
    case 2:
        y = linmap(F2, g2, x);
        break;
    default:
        break;
    }
    }
};
```

Figure A.3: The header file `dcdc.hpp` containing the dynamics of a boost DC-DC converter.

- *ipdl*: the problem of regulating an inverted pendulum to the upright position (Section 7.3.2).
- *temp*: control the room temperature (4-mode system) to a desired temperature (a setpoint) and keep the temperature around the setpoint.
- *vdp*: estimation of the ROA for Van der Pol equations (Section 7.3.1).
- *locomotion*: the simulations given in Chapter 8.

The future development of ROCS will focus on:

- Implement the interface between user input and the actual control synthesis so that the user does not have to write compatible C++ files to perform control synthesis.
- Improve the computational efficiency by designing a more proper data structure or searching algorithm.

Appendix B

Euclidean-to-Riemmannian Mapping for Locomotion Modes

Closed-form solutions of the phase-space manifolds are required to define the robustness margin sets in Definition 8.4. The followings are the closed-form solutions the locomotion modes presented in Section 8.1.1. A detailed derivation can be found in [140].

Proposition B.1 (PIPM phase-space tangent manifold). Given the PIPM mode defined in (8.4) with initial conditions $(x_0, \dot{x}_0) = (x_{\text{foot}}, \dot{x}_{\text{apex}})$ and known foot placement x_{foot}, the phase-space tangent manifold is characterized by the states $(x, \dot{x}, x_{\text{foot}}, \dot{x}_{\text{apex}})$ such that

$$\sigma(x, \dot{x}, x_{\text{foot}}, \dot{x}_{\text{apex}}) = \frac{\dot{x}_{\text{apex}}^2}{\omega_{\text{PIPM}}^2}\big(\dot{x}^2 - \dot{x}_{\text{apex}}^2 - \omega_{\text{PIPM}}^2(x - x_{\text{foot}})^2\big), \tag{B.1}$$

where σ denotes the Riemannian distance to the nominal phase-space manifold i.e., $\sigma = 0$).

Proposition B.2 (PIPM phase-space cotangent manifold). Let ζ_0 be a nonnegative scaling value representing the initial phase of a cotangent manifold. Given the PIPM in (8.4) and a specific initial state $(x_0, \dot{x}_0)$ different from the keyframe $(x_{\text{foot}}, \dot{x}_{\text{apex}})$, the cotangent manifold is characterized by the states $(x, \dot{x}, x_0, \dot{x}_0)$ such that

$$\zeta(x, \dot{x}, x_0, \dot{x}_0) = \zeta_0(\frac{\dot{x}}{\dot{x}_0})^{\omega_{\text{PIPM}}^2}\frac{x - x_{\text{foot}}}{x_0 - x_{\text{foot}}}, \tag{B.2}$$

where ζ_0 is chosen as the phase progression value at the keyframe state in this study.

This cotangent manifold represents the arc length along the tangent manifold σ in Eq. (B.1). We use this cotangent manifold to quantify the length of a phase-space robustness margin.

Detailed derivations of these two closed-form solutions above, i.e., $\sigma(x, \dot{x}, x_{\text{foot}}, \dot{x}_{\text{apex}}) = 0$ and $\zeta(x, \dot{x}, x_0, \dot{x}_0) = 0$, are provided in [140].

Proposition B.3 (PPM phase-space tangent manifold). Given the PPM in (8.23) with initial conditions $(x_0, \dot{x}_0) = (x_{\text{foot}}, \dot{x}_{\text{apex}})$ and known arm placement x_{foot}, the PPM phase-space tangent manifold is defined as

$$\sigma(x, \dot{x}, \dot{x}_{\text{apex}}, x_{\text{foot}}) = \frac{\dot{x}^2_{\text{apex}}}{-\omega^2_{\text{PPM}}}\big(\dot{x}^2 - \dot{x}^2_{\text{apex}} + \omega^2_{\text{PPM}}(x - x_{\text{foot}})^2\big), \tag{B.3}$$

Compared to the PIPM tangent manifold in Proposition B.1, the PPM tangent manifold has a negative asymptote slope square, i.e., $-\omega^2_{\text{PPM}}$. Thus, the tangent manifold with $\sigma > 0$ locates beneath the nominal $\sigma = 0$ tangent manifold. This property is in contrast to that of the PIPM tangent manifold.

Proposition B.4 (PPM phase-space cotangent manifold). Given the PPM in (8.23), the PPM cotangent manifold is

$$\zeta = \zeta_0(\frac{\dot{x}}{\dot{x}_0})^{-\omega^2_{\text{PPM}}} \frac{x - x_{\text{foot}}}{x_0 - x_{\text{foot}}}, \tag{B.4}$$

Proposition B.5 (MCM phase-space tangent manifold). Given the MCM with a constant acceleration ω_{MCM} (i.e., the control input), an initial condition $(x_0, \dot{x}_0) = (x_{\text{foot}}, \dot{x}_{\text{apex}})$, and a known foot placement x_{foot}, the MCM phase-space tangent manifold is

$$\sigma(x, \dot{x}, x_{\text{foot}}, \dot{x}_{\text{apex}}) = 2\omega_{\text{MCM}}(x - x_{\text{apex}}) - (\dot{x}^2 - \dot{x}^2_{\text{apex}}), \tag{B.5}$$

where $\sigma = 0$ represents the nominal phase-space tangent manifold.

Proposition B.6 (MCM phase-space cotangent manifold). Given the MCM with a constant acceleration and initial conditions $(x_0, \dot{x}_0) = (x_{\text{foot}}, \dot{x}_{\text{apex}})$ and known foot placement x_{foot}, the phase-space cotangent manifold is

$$\zeta(x, \dot{x}, x_{\text{foot}}, \dot{x}_{\text{apex}}) = \omega_{\text{MCM}} \cdot \ln(\frac{\dot{x}}{\dot{x}_{\text{apex}}}) - (x - x_{\text{foot}}), \tag{B.6}$$

The phase-space manifolds of the hopping model are trivial since its tangent phase-space manifold is a horizontal line. The stop-launch model and sliding model have similar phase-space manifolds (i.e., parabolic trajectories) as those of the multi-contact model since all of them has a constant sagittal acceleration. Their derivations are omitted for brevity.