Doctoral Thesis in Electrical Engineering

Safe Autonomy under Uncertainty: Computation, Control, and Application

YULONG GAO

Stockholm, Sweden 2020

NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE

Safe Autonomy under Uncertainty: Computation, Control, and Application

YULONG GAO

Academic Dissertation which, with due permission of the KTH Royal Institue of Technology and the Nanyang Technological University, is submied for public defence for the Degree of Doctor of Philosophy on Friday the 4th of December 2020, at 09:00 a.m. in F3, Lindstedtsvägen 26, KTH Campus, Stockholm.

Doctoral Thesis in Electrical Engineering
KTH Royal Institute of Technology
Stockholm, Sweden 2020

ISBN 978-91-7873-691-1
TRITA-EECS-AVL-2020:58

Printed by: Universitetsservice US-AB, Sweden 2020

Abstract

Safety is a primary requirement for many autonomous systems, such as automated vehicles and mobile robots. An open problem is how to assure safety, in the sense of avoiding unsafe subsets of the state space, for uncertain systems under complex tasks. In this thesis, we solve this problem for certain system classes and uncertainty descriptions by developing computational tools, designing verification and control synthesis algorithms, and evaluating them on two applications.

As our first contribution, we consider how to compute probabilistic controlled invariant sets, which are sets the controller is able to keep the system state within with a certain probability. By using stochastic backward reachability, we design algorithms to compute these sets. We prove that the algorithms are computationally tractable and converge in a finite number of iterations. We further consider how to compute invariant covers, which are covers of sets that can be enforced to be invariant by a finite number of control inputs despite disturbances. A necessary and sufficient condition on the existence of an invariant cover is derived. Based on this result, an efficient computational algorithm is designed.

The second contribution is to develop algorithms for model checking and control synthesis. We consider discrete-time uncertain systems under linear temporal logic (LTL) specifications. We propose the new notion of temporal logic trees (TLT) and show how to construct TLT from LTL formulae via reachability analysis for both autonomous and controlled transition systems. We prove approximation relations between TLT and LTL formulae. Two sufficient conditions are given to verify whether a transition system satisfies an LTL formula. An online control synthesis algorithm, under which a set of feasible control inputs can be generated at each time step, is designed, and it is proven to be recursively feasible.

As our third contribution, we study two important vehicular applications on shared-autonomy systems, which are systems with a mix of human and automated decisions. For the first application, we consider a car parking problem, where a remote human operator is guided to drive a vehicle to an empty parking spot. An automated controller is designed to guarantee safety and mission completion despite unpredictable human actions. For the second application, we consider a car overtaking problem, where an automated vehicle overtakes a human-driven vehicle with uncertain motion. We design a risk-aware optimal overtaking algorithm with guaranteed levels of safety.

Sammanfattning

Säkerhet är ett primärt krav för många autonoma system, såsom automatiserade fordon och robotar. En öppen frågeställning är hur man kan säkerställa säkerhet, det vill säga undvika farliga delmängder av tillståndsrummet, för osäkra system under komplexa specifikationer. I denna avhandling löser vi detta problem för vissa klasser av system och osäkerhetsformaliseringar genom att utveckla beräkningsverktyg, utforma verifierings- och styrsyntesalgoritmer och utvärderar dem i två tillämpningar.

Som första bidrag studerar vi hur man beräknar probabilistiskt styrda invarianta mängder, vilka är mängder regulatorn kan hålla systemtillståndet inom med en viss sannolikhet. Genom att använda stokastisk bakåtriktad uppnåelighet utformar vi algoritmer för att beräkna dessa mängder. Vi bevisar att dessa algoritmer är beräkningsmässigt hanterbara och konvergerar inom ett finit antal iterationer. Vi studerar också hur man beräknar invarianta övertäckningar, som är övertäckningar av mängder vilka kan tvingas vara invarianta med ett finit antal styrinsignaler trots störningar. Vi härleder ett nödvändigt och tillräckligt villkor för existensen av en invariant övertäckning. Med detta existensvillkor designas en effektiv beräkningsalgoritm.

Vårt andra bidrag utvecklar algoritmer för modellkontroll och styrsyntes. Vi betraktar osäkra system i diskret tid under specifikationer givna via linjär tidslogik (LTL). Vi introducerar ett nytt begrepp vid namn temporala logiska träd (TLT) och visar hur man konstruerar TLT från en LTL-formel via uppnåelighets-analys för både autonoma system och kontrollerade övergångssystem. Vi visar approximationsförhållanden mellan TLT- och LTL-formler. Två tillräckliga villkor ges för att verifiera om ett övergångssystem uppfyller en LTL-formel. En online-styrsyntesalgoritm är utvecklad, under vilken en uppsättning möjliga styrsignaler kan genereras vid varje tidssteg, och vi visar att den är rekursivt görbar.

Som tredje bidrag studerar vi två viktiga fordonstillämpningar inom delade autonoma system, system med en mix av mänskliga och automatiserade beslut. I den första tillämpningen studerar vi ett bilparkeringsproblem, där en avlägsen mänsklig operatör guidas för att köra ett fordon till en tom parkeringsplats. En automatiserad regulator designas för att garantera säkerhet och uppdragets slutförande trots oförutsägbara mänskliga handlingar. I den andra tillämpningen studerar vi ett omkörningsproblem där ett automatiserat fordon kör om ett mänskligt fordon med osäker framtida körning. Vi designar en riskmedveten optimal omkörningsalgoritm med garanterade säkerhetsnivåer.

Acknowledgments

The time to pursue a PhD is destined to be an extraordinary stage of my life, which is full of good memories and unforgettable experiences. It is a journey with many companions without whose constant help and guidance this thesis could not have been possible.

First, I would like to express my deepest gratitude to my supervisors, Prof Karl Henrik Johansson and Prof Lihua Xie, for offering me a unique opportunity to work at the NTU-KTH Joint PhD Programme. I learn from Kalle persistent enthusiasm in research and open-mindedness to new questions. Kalle always guides me to pay attention to details, as well as encourages me to aim at the big picture. I learn from Prof Xie extensive domain knowledge and deep insight in new areas. Prof Xie always gives me timely and valuable feedback. I could not have wished for better supervisors. I hope to be able to help others in the future as they have helped me.

I am grateful to my master supervisor Prof Yuanqing Xia for guiding me into the field of systems and control. My warmest thanks to Prof Alessandro Abate for being the opponent of my licentiate defense and offering me an opportunity to learn formal methods at Department of Computer Science at University of Oxford. I am very impressed by many interesting and fruitful discussions in Oxford. I owe special thanks to Prof Mark Cannon, the collaboration with whom makes my visit to Oxford more rewarding. I want to extend my thanks to Prof John Baras for the inspiring discussion. Many thanks to Prof Henrik Sandberg for being the advance reviewer of both my licentiate and doctoral theses.

I also would like to thank Prof Frank Allgöwer from University of Stuttgart for being the opponent, and Dr. Thao Dang from Verimag, Prof Melanie Zeilinger from ETH, and Prof Rong Su from NTU for acting as the committee members. Many thanks to Prof Jonas Mårtensson for chairing the defence, and Prof Henrik Sandberg and Prof Jana Tumova for willing to be the substitutes.

Sincere thanks to Prof Dimos V. Dimarogonas, Prof Ling Shi, Frank J. Jiang, Matin Jafarian, Pian Yu, Li Dai, Mirco Giacobbe, Xiaoqiang Ren, and Shuang Wu for the great collaborations. Special appreciations to Miel Sharf, Amr Alanwar, Sebin Gracy, Frank J. Jiang, and Ting Bai for proof reading this thesis, and Elis Stefansson for translating the abstract into Swedish. Specifically, to Frank, thank you for bringing the concept of AVTCT to me and inspiring me a lot from the chat with you; to Miel, thank you for pointing out and helping me correct a mistake in one work.

Many thanks to all my colleagues (current and former) at the NetCon group as well as the division of DCS for creating a friendly environment and an active working atmosphere, and for their continuous support for everything that I need. I would, however, like to give my heartfelt thanks to Xinlei Yi, Jieqiang Wei, Takuya Iwaki, Junfeng Wu, Robert Mattila, and Ehsan Nekouei for their insightful discussions and assistance, and give my special thanks to Wei Chen for encouraging me at the beginning of my PhD study. Also thanks to Othmane Mazhar, Rong Du, and Manne Held for their kind assistance when working in the same office. I am grateful to Yufeng Yue, Liang Xu, Chule Yang, Yushen Long, Haoyuan Zhang, Mingxing Wen, and Kun Cao from NTU for their warm welcome and host when I was there, and to Weida Zhang, Yongchao Huang, Andrea Peruffo, Hosein Hasanbeig, Xiaoxuan Lu, and Shuhao Yan from Oxford for their help and discussion during my visit. I also want to thank the administrators (current and former) Gerd Franzon, Silvia Cardenas Svensson, Hanna Holmqvist, Anneli Ström, Felicia Gustafsson, Tord Christer Magnusson, and Karin Karlsson Eklund at KTH, and Chua-Goh Wei Jiuan and Goh-Fong Lai Peng from NTU, for their assistance and support.

Finally, I wish to dedicate this thesis to my family. To my parents and sister, thank you for always being there and for the unconditional support throughout my life and studies. A special thanks full of love goes to my wife Pian, thank you for the happiness that you have brought to me, which has made my life colorful, and being my greatest supporter.

Yulong Gao
Stockholm, Sweden
November, 2020

Contents

Abstract iii

Sammanfattning v

Acknowledgments vii

Notation xi

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Illustrative Examples . . . 3
- 1.3 Challenges . . . 7
- 1.4 Related Work . . . 11
- 1.5 Problem Formulation . . . 14
- 1.6 Thesis Outline and Contributions . . . 17

2 Mathematical Background 23
- 2.1 System Models . . . 23
- 2.2 Reachability Analysis . . . 30
- 2.3 Temporal Logic . . . 35
- 2.4 Dynamic Programming . . . 36

3 Computation of Probabilistic Controlled Invariant Sets 39
- 3.1 Introduction . . . 40
- 3.2 Finite-horizon Probabilistic Controlled Invariant Sets . . . 42
- 3.3 Infinite-horizon Probabilistic Controlled Invariant Sets . . . 54
- 3.4 Examples . . . 62
- 3.5 Summary . . . 68

4 Computation of Invariant Covers **69**
4.1 Introduction . . . 70
4.2 Problem Formulation . . . 71
4.3 Existence Conditions . . . 73
4.4 Cardinality Bounds . . . 83
4.5 Computational Algorithm . . . 87
4.6 Examples . . . 90
4.7 Summary . . . 98

5 Verification and Control based on Temporal Logic Trees **99**
5.1 Introduction . . . 100
5.2 Temporal Logic Trees . . . 104
5.3 Model Checking . . . 111
5.4 Control Synthesis . . . 113
5.5 Numerical Evaluations . . . 120
5.6 Summary . . . 128

6 Car Parking Application **135**
6.1 Introduction . . . 136
6.2 Problem Formulation . . . 138
6.3 Guiding Controller . . . 139
6.4 Experiments . . . 144
6.5 Summary . . . 149

7 Car Overtaking Application **151**
7.1 Introduction . . . 152
7.2 Preliminaries . . . 155
7.3 Problem Formulation . . . 157
7.4 Risk-Aware Reachability Analysis . . . 159
7.5 Risk-Aware Optimal Overtaking . . . 165
7.6 Numerical Evaluations . . . 171
7.7 Summary . . . 180

8 Conclusions and Future Research **181**
8.1 Conclusions . . . 181
8.2 Future Research Directions . . . 183

Notation

Real Analysis

$\mathbb{R}^n$	real Euclidean space of dimension n
$\mathbb{N}$	set of nonnegative integers
$\mathbb{N}_{\geq q}$	set $\{r \in \mathbb{N} \mid r \geq q\}$
$\mathbb{N}_{\leq q}$	set $\{r \in \mathbb{N} \mid r \leq q\}$
$\mathbb{N}_{[q,s]}$	set $\{r \in \mathbb{N} \mid q \leq r \leq s\}$
$\mathbb{B}_r(x)$	Euclidean ball with centre x and radius r
$\|\cdot\|$	Euclidean norm for vectors
$\mathrm{conv}(\mathbb{X})$	convex hull of $\mathbb{X}$
$\mathbb{X} \oplus \mathbb{Y}$	Minkowski sum $\{x + y \mid x \in \mathbb{X},\ y \in \mathbb{Y}\}$
$\mathbb{X} \ominus \mathbb{Y}$	Pontryagin difference $\{x \mid x + y \in \mathbb{X},\ \forall y \in \mathbb{Y}\}$
$\mathbb{X} \setminus \mathbb{Y}$	set difference $\{x \mid x \in \mathbb{X}, x \notin \mathbb{Y}\}$
$\lvert\mathbb{X}\rvert$	cardinality of set $\mathbb{X}$
$\mathcal{B}(\mathbb{X})$	Borel space of set $\mathbb{X}$
$\mathrm{cl}(\mathbb{X})$	closure of set $\mathbb{X}$
$\mathbb{1}_{\mathbb{X}}(\cdot)$	indicator function of set $\mathbb{X}$

Linear Algebra

I	identity matrix
$\mathbf{1}$	matrix or vector of ones
$\mathbf{0}$	matrix or vector of zeros
$[A]_i$	ith row of matrix $A \in \mathbb{R}^{r \times n}$
$[A]_{ij}$	(i,j)-th element of matrix $A \in \mathbb{R}^{r \times n}$

Others

Given a polytope	$\mathbb{P} = \{x \in \mathbb{R}^n : Vx \leq v\}$
$\mathrm{vert}(\mathbb{P})$	vertices of $\mathbb{P}$
$\mathrm{vert}_k(\mathbb{P})$	set $\{x \in \mathrm{vert}(\mathbb{P}) : [V]_k x < [v]_k\}$
$x_{k+i\|k}$	a prediction of x with i steps ahead from time k
E	expectation
$\Pr$	probability

Chapter 1

Introduction

Safety is a primary requirement for many autonomous systems: mobile robots in the warehouse should perform actions for avoiding obstacles, and automated vehicles on the roads have to be safe for human passengers. Ensuring design correctness and safety has significant implications for the deployment of autonomous systems. However, this is challenging due to inevitable uncertainties and task complexity.

Uncertainties are present in most control systems. For example, plants are usually corrupted by external disturbances and accompanied by model errors. These uncertainties make it difficult to predict and reason about system behaviors. On the other hand, there is an increasing need for more complex control objectives, such as constrained spatial and temporal tasks, to be achieved by autonomous systems. These complex tasks pose new challenges in both computation and control design. At this background, an open question is

How to assure safety for uncertain systems under complex tasks?

In this thesis, safety is in the sense of avoiding unsafe subsets of the state space. We answer the above question for certain system classes and uncertainty descriptions by developing computational tools, designing verification and control synthesis algorithms, and evaluating them on two applications.

The outline of this chapter is as follows. Section 1.1 provides the motivations. Section 1.2 gives several illustrative examples. Sections 1.3 highlights the main challenges in safe and shared autonomy. Section 1.4 briefly reviews related work. Section 1.5 formalizes the problems that will be studied. Section 1.6 gives the outline and contributions of the thesis.

1.1 Motivation

The development in automation technology over the past few decades has changed the way we live as individuals and as a society. There is an increasing number of machines and systems capable of performing tasks, even within unstructured environments. Robotic vacuum cleaners clean our houses; drones are used for photography and search and rescue missions; and self-driving cars are being tested. We usually call such systems autonomous systems, which can be defined as systems capable of make decisions and performing actions by themselves, without explicit human control [1]. It is believed that autonomous systems have the potential to affect every part of life, business, industry, and education [2].

The need for autonomous systems is driven by various business cases but also areas where autonomy promises to do things that could not be done before. For example, autonomous systems has made the space and deep sea exploration possible. As shown in Figure 1.1, Curiosity Rover, an autonomous exploration vehicle, has been delivered to Mars studying the climate and geology. Autonomous underwater vehicles are becoming important to study the ocean, see Figure 1.2. A variety of sensors affixed to the vehicle can map the seafloor or analyze properties of the water.

Autonomous systems are expected to improve safety and energy efficiency. Comparing with human-driven vehicles, it is estimated that self-driving cars can achieve up to 80% reduction of road crashes and 60% reduction of carbon dioxide [3]. They could also offer new mobility options to millions of elderly or disabled people. An increasing number of corporations, e.g., Uber, Tesla, and Google, and traditional car manufacturers, e.g., Volvo, Ford, and General Motors, are investing in self-driving car technology. The market is estimated to be worth 54 billion and is expected to increase tenfold in the next 5–7 years [3].

The introduction of autonomous systems offers benefits to many applications where humans are present today but also in future operations. Shared-autonomy systems are systems that can mix human and automated decisions in a systematic way. Using shared control, teleoperated assistive robots promise to help people with disabilities eat independently, without relying on caregivers [4], [5]. In the mixed traffic with automated and human-driven cars, automated cars can be used to control the traffic flow and improve traffic efficiency [6]–[9]. The interaction between automated cars and human-driven

Figure 1.1: Mars Curiosity Rover helps to determine whether Mars could ever have supported life and study the planet's climate and geology. *Source*: https://mars.nasa.gov/msl/home/

cars is a research topic that has gained a lot of attentions recently [10]–[12].

Nowadays, technological advances and cost reductions in computing, sensing, and communication are bringing autonomous systems closer to reality. The powerful computing capabilities of microcomputers and the cloud are important for processing data in real time and making quick decisions. The increasing availability of cheap sensors, such as cameras and radars, and new technologies for sensor fusion make the operation of systems feasible in unknown environments. The expansion of 5G and other communication networks is offering more opportunities for autonomous systems to cooperate with each other. In addition, artificial intelligence, machine learning, and edge computing are also accelerating the advancement of autonomous systems.

1.2 Illustrative Examples

In this section, three examples are provided to motivate the problems considered in this thesis.

Air Traffic Management System

With long-term increasing number of commercial air travels, an advanced and efficient air traffic management system is needed for mitigating air congestion

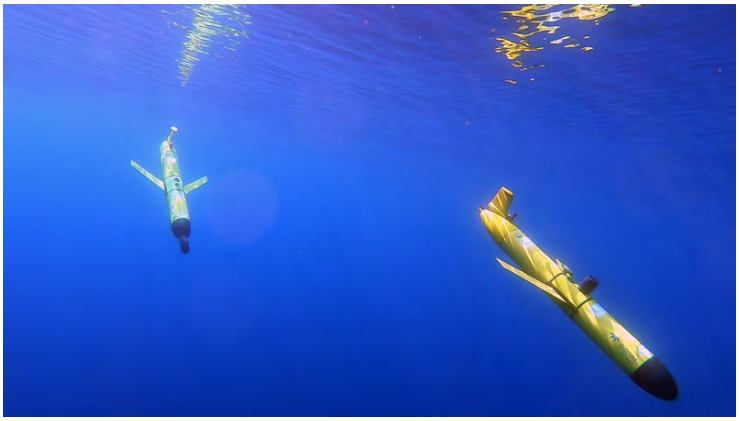


Figure 1.2: Autonomous underwater vehicles map the seafloor and analyze water properties. *Source*: https://transmitter.ieee.org/auvs-how-autonomous-underwater-vehicles-protect-oceans-and-divers/

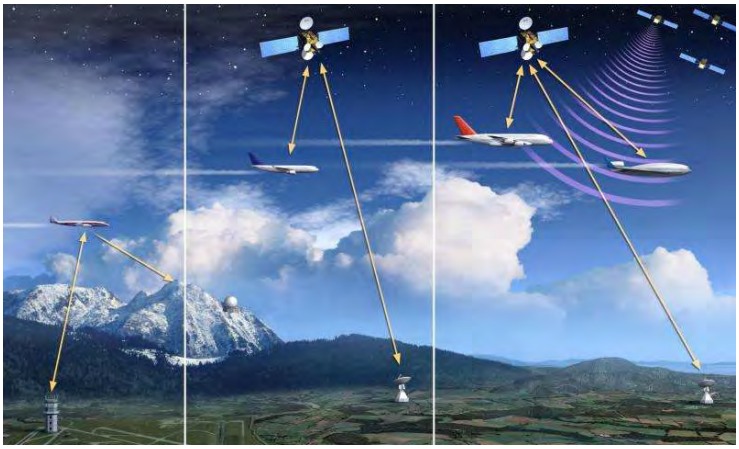


Figure 1.3: Illustration of an automated air traffic management system supported by satellite communication. *Source*: https://www.esa.int/Applications/Telecommunications_Integrated_Applications/ESA_launches_new_programme_for_air_traffic_management_via_satellite

and improving air safety, see Figure 1.3. In the past decades, NASA has been being active in developing such a new, highly-automated, and safe air traffic management system. An important problem for air traffic management is to resolve potential conflicts between aircrafts in the presence of uncertainties, including sensing noises and environmental disturbances.

In [13], [14], an air traffic management system is modeled as a stochastic hybrid control system. To ensure safety, the problem is formulated as minimizing the probability that the system states reach unsafe regions. Stochastic reachability can be used for solving this problem [15]. In this thesis, we consider stochastic invariance. We introduce the notion of probabilistic controlled invariance sets (PCISs), within which the controller is able to keep the system state with a certain probability. The computation problem of PCIS is investigated in Chapter 3.

Automated Driving

Automated vehicles are expected to move in complex and dynamic traffic environments. Since it is impossible to enumerate all possible traffic scenarios, expressing the task specifications in an efficient way is important. It has been shown in [16]–[18] that formal language, e.g., linear temporal logic (LTL) or signal temporal logic (STL) formulae, is expressive enough for defining specifications of automated vehicles. Thereby, formal verification and control can become crucial for the design of automated vehicles. Automaton-based approaches have been demonstrated on Alice, an autonomous vehicle developed by the California Institute of Technology for the 2007 DARPA Urban Challenge [16]. Similar techniques have been applied to the autonomous cars developed by nuTonomy [19], see Figure 1.4. Chapter 5 proposes a new approach for formal verification and control of discrete-time uncertain systems under LTL specifications. The application to automated driving even in uncertain and unstructured traffic environments is investigated.

Automated vehicles should perform safe actions even when human-driven vehicles are present. Let us consider a car overtaking scenario as shown in Figure 1.5, where an automated vehicle V_1 tries to overtake a human-driven vehicle V_2. Due to the inherent uncertainties of human decisions, V_2 may move at a non-constant velocity. This poses some challenges for prediction and control. Chapter 7 investigates this problem and develops a risk-aware solution to the car overtaking problem even though the overtaken vehicle is with uncertain motion.

Figure 1.4: nuTonomy's robo-taxi tested in Singapore. *Source*: https://www.campus.sg/singapore-is-testing-worlds-first-robo-taxis/

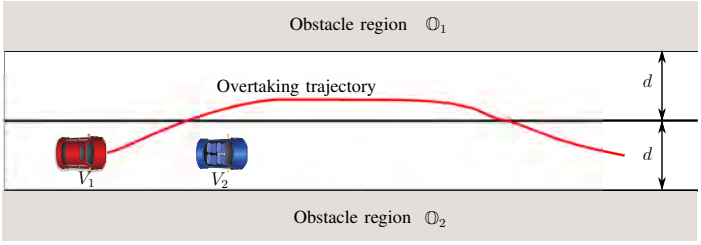


Figure 1.5: A car overtaking scenario where an automated vehicle V_1 tries to overtake a human-driven vehicle V_2.

Teleoperation

Teleoperation refers to operating a system at a distance over a communication network. It has been widely used for a variety of applications such as telesurgery and mining. The autonomous trucking startup Einride proposed a hybrid control solution that combines automated control technology and teleoperation such that remote human operators can take over as needed. Figure 1.6 shows how human operators are able to control multiple autonomous trucks from a single remote drive station.

Another teleoperation scenario is a remote car parking, see Figure 1.7, human operator can monitor the real-time parking environment and remotely perform parking maneuvers. The problem is how to ensure that the human operator safely drive the vehicle to the empty parking spot $\mathbb{P}_1$ or $\mathbb{P}_2$. Chapter 6 investigates this remote parking scenario and develops a guiding controller to address the problem.

Figure 1.6: Teleoperation of trucks by Einride where human operators are able to control multiple autonomous trucks from a single remote drive station [20].

1.3 Challenges

In the past decades, researchers and engineers have paid significant attention to safety of autonomous systems. Still, in recent years, we have seen fatal accidents caused by self-driving vehicles [21], and many remaining challenges exist [22].

Autonomous systems often need to interact or cooperate with human operators. Shared-autonomy systems are systems with a mix of human and automated decisions. Unpredictable behaviors of humans pose many uncertainties for the design, which makes the safe control of shared-autonomy systems a challenging problem.

In the following, we will detail some of the main challenges in safe autonomy and shared autonomy.

1.3.1 Challenges in Safe Autonomy

Existing approaches fail to provide provable guarantees of safety for uncertain systems under complex tasks. The approaches in the industrial field to ensure safety are often based on large-scale simulations and field tests. In addition to their huge financial cost, these approaches are fragile since there will always exist untested scenarios [23]. Although robust control approaches have been developed in the literature for simple safe tasks, e.g., collision-free in a well-

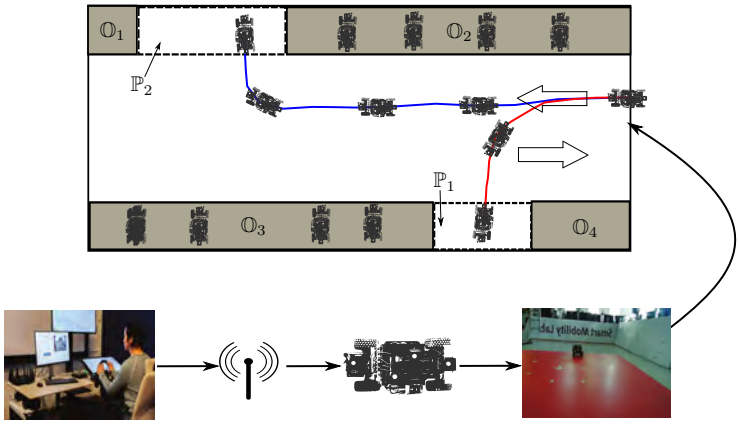


Figure 1.7: A parking situation where a remote human operator would like to drive a vehicle to a narrow parking space $\mathbb{P}_1$ or a broad parking space $\mathbb{P}_2$.

defined environment [24], how to formally deal with uncertainties for complex tasks is still a challenging problem.

Let us take automated driving as an example. Automated vehicles should be able to move without colliding with obstacles, such as other moving vehicles or pedestrians. To reason about such scenarios, it is natural to introduce safe regions, which are sets where the safety requirements are satisfied. A first essential question is how to compute safe regions despite environmental uncertainties. Automated vehicles are expected to perform desired driving tasks in the safe regions. These tasks can be complex combinations of different driving maneuvers. It is of great interest to automate the control synthesis for such tasks.

Let us next discuss particular challenges on computing safe sets and synthesizing controllers.

Computational Challenges

Computing safe sets for autonomous systems is usually challenging or even intractable. In the literature, invariant sets are often used to act as safe sets [25]. Invariant sets are sets where the system is able to stay under some admissible control input. The problem of computing safe regions can be formulated as computing invariant sets, which are closely related to reachable sets. If uncertainties are bounded, robust controlled invariant sets address the in-

variance under any realization of the disturbance. The computation of robust controlled invariant sets has been studied in [26]–[28]. If the uncertainties are with known probability distributions instead, it is natural to define sets where the state can be kept with a required probability. In Chapter 3, we formally define such sets as PCISs. The main challenges in computing PCISs are the computational tractability and convergence of the algorithms.

In some scenarios, the system is remotely controlled for staying in a safe set through a wireless communication network [29]–[31]. Limited network recourses pose a set of essential questions. How much information exchange is needed to enforce invariance? How can we design a coder-controller pair to ensure invariance of a given set for an uncertain system with finite data-rate communication? The first question is conceptually answered by the notion of invariance feedback entropy [32]–[34]. The second question has not been explored in the literature. In Chapter 4, we propose an answer to this question by addressing the existence and computation of an invariant cover. The invariant covers are covers of sets that can be enforced to be invariant by a finite number of control inputs despite disturbances. Note that the invariant cover acts as the basis of the invariance feedback entropy.

Control Challenges

Automated control synthesis under complex tasks is a hard problem. Formal languages can encode safety requirements into specifications more complex than invariance. For example, an LTL formula is expressive enough to capture complex combinations of Boolean and temporal statements. LTL formulae have recently been used in robotics and automated driving [35].

Formal methods are used for model checking and control synthesis [36]. Model checking is to automatically verify whether the behavior of the system satisfies a given specification. Control synthesis is to automatically design controllers so that the behavior of the system provably satisfies a given specification. Formal verification and control have shown great promise for safe autonomy in the past decade [37]. However, model checking and control synthesis are at the same time challenging when considering dynamical systems affected by uncertainties, and in particular uncertain infinite systems under complex, temporal logic specifications. The main reasons are that: the traditional automaton-based methods rely on the abstraction to finite systems, which is restrictive due to heavy computation and lack of generality; it is difficult to reason about uncertainties propagating along infinite trajectories. In

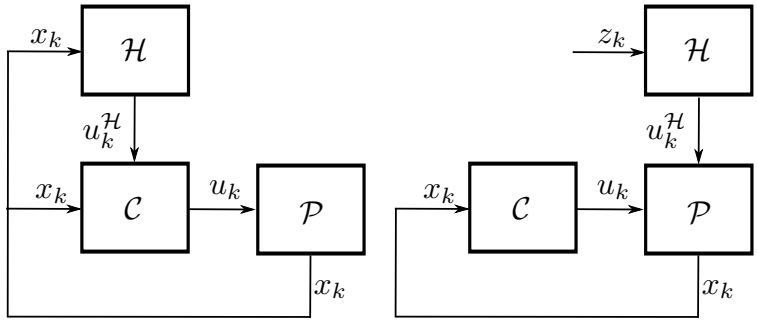


Figure 1.8: Block diagram of shared-autonomy systems where the human actions are used proactively (a) or reactively (b).

Chapter 5, we propose a new tool, temporal logic trees (TLT), and develop TLT-based approaches to model checking and control synthesis problems to tackle these challenges, particularly for discrete-time uncertain systems under LTL specifications.

1.3.2 Challenges in Shared Autonomy

In this subsection we discuss the challenges for two kinds of shared-autonomy systems based on whether the human actions are integrated into the control systems proactively or reactively, i.e., if the control command is based on knowledge of the human decisions or not.

Proactive Human Actions

Let us consider the remote parking example in Figure 1.7. In this example, the human operator is given the priority to make decisions for performing the parking task while the on-board controller of the vehicle is able to mix human's actions for ensuring safety. Such a system can be illustrated by the block diagram in Figure 1.8(a). This block diagram represents shared-autonomy systems where the human actions are used proactively. The human operator $\mathcal{H}$ makes decisions based on the state x_k and perform actions $u_k^{\mathcal{H}}$. These actions are not directly implemented onto the plant $\mathcal{P}$ (e.g., the vehicle in Figure 1.7), but are intervened by the controller $\mathcal{C}$ (e.g., the on-board controller of the vehicle). The controller $\mathcal{C}$ can check if the human decisions are safe and filter them

whenever needed. This control system architecture makes it possible to handle unpredictable human decisions. However, a challenging problem is how to integrate the automated controller with the human decision making for achieving complex tasks as well as ensuring safety [38], [39]. In Chapter 6, we propose such a guiding controller solution for the remote car parking problem.

Reactive Human Actions

Different from the remote parking example, the car overtaking scenario in Figure 1.5 exemplifies another kind of shared-autonomy systems. The human driver performs actions on the vehicle V_2, which has a direct influence on the automated vehicle V_1. Note that the human actions can not be intervened by the controller of V_1. We illustrate this system through the block diagram in Figure 1.8(b), which represents a shared-autonomy system where the human actions are used reactively. The plant $\mathcal{P}$ refers to the automated vehicle V_1 in Figure 1.5 and the controller $\mathcal{C}$ refers to the controller of V_1. The information available to the human operator is denoted by z_k, which is different from the information available in Figure 1.8(a). For such kind of shared-autonomy systems, it is natural to treat the human behaviors as uncertainties when designing the controller. If the human is able to do anything possible, the safe operation space of the automated vehicle will become very small. A challenging problem is thus how to model the human behavior such that the system can complete the desired task as well as perform safe actions for the human. In Chapter 7, we study the car overtaking problem in the presence of uncertain motion of the human-driven vehicle and propose a risk-aware overtaking algorithm.

1.4 Related Work

This section will briefly review the related work on safe and shared autonomy.

1.4.1 Safe Autonomy

There are numerous examples that ensure safety for machines through automatic control. An early example goes back to the steam engine invented by James Watt in 1776, which uses the centrifugal governor to guarantee the engine moving at a nearly constant and safe speed [40]. During World War II, the bang-bang control was successfully applied to the automatic flight control

systems of aircrafts [41]. Nowadays, advanced driver assistance systems are making automobiles much safer than before [42]. There is a large amount of literature on safe autonomy. We restrict our attention to literature on reachability and invariance, and literature on formal verification and control.

Reachability and Invariance

Reachability and invariance are fundamental notions in systems and control. Reachability is coupled to controllability, while invariance is a property closely related to system stability. There is a rich body of research in these two topics. Seminal papers include [43], [44] on reachability and [28], [45], [46] on invariance. Representative textbooks on set-valued control are [47], [48].

Robust approaches have been developed for performing reachability analysis for systems with bounded uncertainties [44], [49]. These results are essential for computing robust (controlled) invariant sets [45], [50]. For systems with random uncertainties, stochastic approaches guarantee the reachability with a given probability [15], [51]. Some recent results are devoted to stochastic invariance [52], [53].

Reachability and set invariance are often used to give formal safety guarantees for autonomous systems [25], [54]–[58]. For example, a method based on discrepancy functions is developed in [54] for computing a bounded reachable set around a simulation trajectory; in [25], safety of robots is explicitly expressed in the form of set invariance; performing reachability analysis is used for safety assessment of autonomous cars in [56]. Furthermore, there are a lot of approaches for computing reachable sets or invariant sets in the literature [27], [59], [60]. In addition, many software packages or toolboxes have been developed for efficiently computing reachable sets or invariant sets, e.g., the multi-parametric toolbox [61], the Hamilton-Jacobi toolbox [62], C2E2 [54], SReachTools [63], and JuliaReach [64].

Formal Verification and Control

Model checking and control synthesis are two fundamental problems. For finite transition systems, a typical approach is to use automata theory [36]. In order to adapt automaton-based methods to infinite systems, abstraction is a central step. A lot of recent results, e.g., [65]–[67], are devoted to equivalence or inclusion relation between the abstract finite system and the original infinite system. For more discussion of formal verification and control, we refer to the textbooks [35], [68], [69].

LTL formulae are useful to define complex specifications for autonomous systems. The applications of control synthesis under LTL specifications include automated driving [16], single-robot control in dynamic environments [70], multi-robot control [71], and transportation control [72]. The control of stochastic systems under LTL is considered in [73] and further applied to multi-robot coordination in [74]. Control synthesis for dynamical systems has been extended also to other specifications like STL [75], and probabilistic computational tree logic [76].

1.4.2 Shared Autonomy

Shared control is rooted at the idea of combining human and machine intelligence. Early examples include Ray Goertz's leader-follower manipulator in 1949 [77] and Marvin Minsky's call for telepresence in 1980 [78]. Due to the rapid development in robotics, shared autonomy has become an active research topic in the past decade. An intuitive design for shared autonomy is to switch the control authority between a human and an automatic controller. This idea has been used in [39], [79], [80]. One drawback of authority switching is the decrease in human's satisfaction [81]. To overcome this, a policy-blending formulation for shared control is provided in [38] by integrating the prediction of user intent and the arbitration with the user input. In this work and some other recent work on shared control [82], inverse reinforcement learning [83] is used to learn reward functions from the human behaviors. In [84], a deep reinforcement learning is used to design a model-free shared autonomy framework.

Shared control has shown great promise in driver-automation cooperation. For example, a lane keeping assist system is designed by using shared control in [85]. Two survey papers on application of shared control to automated vehicles refer to [86], [87]. In addition, modeling and prediction of human driver behavior are crucial to enable safe and efficient automated driving as well as leverage on the shared control. A broad, up-to-date review of the state of the art on this topic is provided in [88].

1.5 Problem Formulation

This section formulates the problems to be addressed in this thesis. Let us consider a discrete-time uncertain control system

$$x_{k+1} = f(x_k, u_k, w_k), \tag{1.1}$$

where $x_k \in \mathbb{S}$ is the state, $u_k \in \mathbb{U}$ the control input, $w_k \in \mathbb{W}$ the uncertainty, and $f : \mathbb{S} \times \mathbb{U} \times \mathbb{W} \to \mathbb{S}$ the system model. In Chapter 2, we will explicitly detail the models considered in this thesis.

Computation of Probabilistic Controlled Invariant Sets

An invariant set is a set where the state can stay and a controlled invariant set is a set where the state can be enforced to stay by a controller. In Chapter 3, we consider the control system (1.1) with stochastic uncertainty w_k. For such a stochastic control system, a PCIS is a set within which the controller is able to keep the state with a certain probability. Formally, we say that a set $\mathbb{Q} \subset \mathbb{S}$ is an ϵ-PCIS if for any $x_0 \in \mathbb{Q}$, there exists a feedback controller $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots)$ such that the generated trajectory $\boldsymbol{p} = x_0 x_1 \ldots$ can be kept into $\mathbb{Q}$ with probability at least ϵ, where $0 \leq \epsilon \leq 1$ is a prescribed number. See Figure 1.9 for an illustration. The following question is on computing an ϵ-PCIS:

Q1 Given a state set $\mathbb{Q}$ and a parameter $0 \leq \epsilon \leq 1$, how to compute a set $\tilde{\mathbb{Q}} \subseteq \mathbb{Q}$ that is invariant with probability ϵ?

This question is essential for computing safe region for stochastic systems and is answered in Chapter 3.

Computation of Invariant Cover

A cover of a set is a collection of sets whose union includes this set as a subset. In Chapter 4, we consider the control system (1.1), but the uncertainty set $\mathbb{W}$ is bounded. Given a set $\mathbb{Q} \subseteq \mathbb{S}$, a pair $(\mathcal{A}, G)$ is an invariant cover of (1.1) and the set $\mathbb{Q}$ if

(i) $\mathcal{A}$ is a cover of $\mathbb{Q}$ and $|\mathcal{A}|$ is finite,

(ii) G is a map from $\mathcal{A}$ to $\mathbb{U}$, and for all $\mathbb{X}^{\text{ic}} \in \mathcal{A}$, $f(\mathbb{X}^{\text{ic}}, G(\mathbb{X}^{\text{ic}}), \mathbb{W}) \subseteq \mathbb{Q}$,

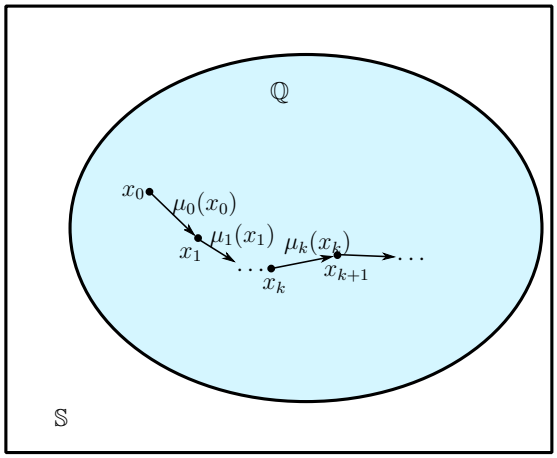


Figure 1.9: Illustration of an ϵ-PCIS, where $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots)$ is the feedback controller.

where $f(\mathbb{X}^{\mathrm{ic}}, u, \mathbb{W}) = \{z \mid z = f(x, u, w), x \in \mathbb{X}^{\mathrm{ic}}, w \in \mathbb{W}\}$. That is, each state set $\mathbb{X}^{\mathrm{ic}}$ from the finite cover $\mathcal{A}$ can be driven to the set $\mathbb{Q}$ by means of a single control input $G(\mathbb{X}^{\mathrm{ic}})$. An illustration of an invariant cover is shown in Figure 1.10, where the cover of $\mathbb{Q}$ is $\mathcal{A} = \{\mathbb{X}_i^{\mathrm{ic}}\}_{i=1}^5$. Define a map G from $\mathcal{A}$ to $\mathbb{U}$ as $G(\mathbb{X}_i^{\mathrm{ic}}) = u_i \in \mathbb{U}$. If for all i, $f(\mathbb{X}_i^{\mathrm{ic}}, u_i, \mathbb{W}) \subseteq \mathbb{Q}$, then $(\mathcal{A}, G)$ is an invariant cover of the set $\mathbb{Q}$. In Chapter 4, we investigate the existence and computation of an invariant cover and answer this question:

Q2 Given a state set $\mathbb{Q}$, provide a necessary and sufficient condition such that there exists an invariant cover $(\mathcal{A}, G)$ for the system and $\mathbb{Q}$. If an invariant cover exists, how to design an algorithm to compute an invariant cover $(\mathcal{A}, G)$?

Answering questions **Q2** provides a way of designing a coder-controller pair that can enforce the system state staying within a safe region through a finite data-rate network.

Verification and Control Synthesis

In Chapter 5, we consider the uncertain control system (1.1) under LTL specifications. We answer the following questions:

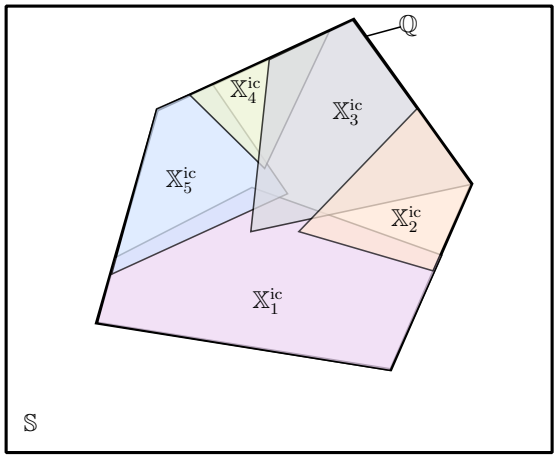


Figure 1.10: Illustration of an invariant cover for the set $\mathbb{Q}$. Here, the cover of $\mathbb{Q}$ is $\mathcal{A} = \{\mathbb{X}_i^{\text{ic}}\}_{i=1}^5$, i.e., $\mathbb{Q} \subseteq \cup_{i=1}^5 \mathbb{X}_i^{\text{ic}}$. Note that the sets $\mathbb{X}_i^{\text{ic}}$ are not necessarily disjoint.

Q3 Given an LTL formula φ and a given feedback controller $\boldsymbol{\mu} = \mu_0\mu_1\ldots$, how to verify if all the trajectories $\boldsymbol{p} = x_0x_1\ldots$, where $x_{k+1} = f(x_k, \mu_k(x_k), w_k), w_k \in \mathbb{W}$, satisfy φ?

Q4 Given an LTL formula φ, how to automatically synthesize a feedback controller $\boldsymbol{\mu} = \mu_0\mu_1\ldots$ such that all the resulting trajectories $\boldsymbol{p} = x_0x_1\ldots$, where $x_{k+1} = f(x_k, \mu_k(x_k), w_k), w_k \in \mathbb{W}$, satisfy φ?

These questions are essential for extending the applications of temporal logic control to dynamic and uncertain environment.

Car Parking Application

In Chapter 6, we consider a car parking problem, as shown in Figure 1.7, where a remote human operator intends to drive a vehicle to an empty parking spot in a parking lot. The question considered is

Q5 Given a set of LTL-specified parking tasks $\{\varphi_i\}$, how to design a guiding controller that is able to (i) infer the human intent on which task φ_i to be completed, (ii) mitigate unpredictable human actions, and (iii) safely park the car to an empty parking spot?

The answer to this question provides new insights for the design of safe teleoperation for connected vehicles.

Car Overtaking Application

In Chapter 7, we consider the car overtaking problem where an automated vehicle tries to overtake a human-driven vehicle, as shown in Figure 1.5. The question to be addressed is

Q6 How to design an algorithm such that the overtaking can be performed safely in the case of overtaken vehicle performing uncertain motion?

Answering question **Q6** is helpful to understand and model the human behaviors during the overtaking.

1.6 Thesis Outline and Contributions

In this section, we outline the contents of the thesis and the contributions.

Chapter 2: Mathematical Background

In Chapter 2, we provide mathematical background, including system modeling, reachability analysis, and dynamic program.

Chapter 3: Computation of Probabilistic Controlled Invariant Sets

This chapter investigates stochastic invariance for control systems through PCISs. As a natural complement to robust controlled invariant sets, we propose finite- and infinite-horizon PCISs, and explore their relation to robust controlled invariant sets. We design iterative algorithms to compute the PCIS within a given set. For systems with discrete spaces, the computations of the finite- and infinite-horizon PCISs at each iteration are based on linear programming and mixed integer linear programming, respectively. The algorithms are computationally tractable and terminate in a finite number of steps. For systems with continuous spaces, we show how to discretize the spaces and prove the convergence of the approximation when computing the finite-horizon PCISs. In addition, it is shown that an infinite-horizon PCIS can be computed by the stochastic backward reachable set from the robust controlled

invariant set contained in it. These PCIS algorithms are applicable to practical control systems. Simulations are given to illustrate the effectiveness of the theoretical results for motion planning.

This chapter is based on the following publication.

- Y. Gao, K. H. Johansson, and L. Xie, "Computing probabilistic controlled invariant sets," *IEEE Transaction on Automatic Control*. To appear.

Chapter 4: Computation of Invariant Covers

This chapter investigates some fundamental problems concerning existence and computation of an invariant cover for uncertain discrete-time linear control systems subject to state and control constraints. We develop necessary and sufficient conditions on the existence of an invariant cover for a polytopic set of states. The conditions can be checked by solving a set of linear programs, one for each extreme point of the state set. Based on these conditions, we give upper and lower bounds on the minimal cardinality of the invariant cover, and design an iterative algorithm with finite-time convergence to compute an invariant cover. We further show in two examples how to use an invariant cover in the design of a coder–controller pair that ensures invariance of a given set for a networked control system with a finite data-rate communication.

This chapter is based on the following publication.

- Y. Gao, M. Cannon, L. Xie, and K. H. Johansson, "Invariant cover: existence, cardinality bounds, and computation," *Automatica*. Submitted.

Chapter 5: Verification and Control based on Temporal Logic Trees

In this chapter, we propose algorithms for performing model checking and control synthesis for discrete-time uncertain systems under LTL specifications. We construct TLT from LTL formulae via reachability analysis. In contrast to automaton-based methods, the construction of the TLT is abstraction-free for infinite systems, that is, we do not construct discrete abstractions of the infinite systems. Moreover, for a given transition system and an LTL formula, we prove that there exist both a universal TLT and an existential TLT via minimal and maximal reachability analysis, respectively. We show that the universal TLT is an underapproximation for the LTL formula and the existential TLT is an overapproximation. We provide sufficient conditions and

necessary conditions to verify whether a transition system satisfies an LTL formula by using the TLT approximations. As a major contribution of this work, for a controlled transition system and an LTL formula, we prove that a controlled TLT can be constructed from the LTL formula via control-dependent reachability analysis. Based on the controlled TLT, we design an online control synthesis algorithm, under which a set of feasible control inputs can be generated at each time step. We also prove that this algorithm is recursively feasible. We illustrate the proposed methods for both finite and infinite systems and highlight the generality and online scalability with two simulated examples.

This chapter is based on the following publication.

- Y. Gao, A. Abate, F. J. Jiang, M. Giacobbe, L. Xie, and K. H. Johansson, "Temporal logic trees for model checking and control synthesis of uncertain discrete-time systems," *IEEE Transaction on Automatic Control*. Submitted.

Chapter 6: Car Parking Application

In this chapter, we propose a guiding controller solution to the car parking problem, where a human operator remotely drives the vehicle to an empty parking spot. We specify the parking task as a set of LTL formulae. Then, using the TLT-based approach, we synthesize a set of controllers for assisting the human operator to complete the mission, while guaranteeing that the system maintains specified spatial and temporal properties. We assume the human operator's exact preference of how to complete the mission is unknown. Instead, we use a data-driven approach to infer and update the human operator's preference over parking spots in real-time. If, while the human is operating the vehicle, she provides inputs that violate any of the invariances prescribed by the LTL formula, our verification-based controller will use its internal belief of the human operator's intended objective to guide the operator back on track. Moreover, we show that as long as the specifications are initially feasible, our controller will stay feasible and can guide the human to park the vehicle in the empty spot safely despite some unexpected human actions. We demonstrate the results on the Small Vehicles for Autonomy (SVEA) platform.

This chapter is based on the following publications.

- Y. Gao, F. J. Jiang, X. Ren, L. Xie, and K. H. Johansson, "Reachability-

based human-in-the-loop control with uncertain specifications," In *Proceedings of IFAC World Congress*, 2020.

- F. J. Jiang, Y. Gao, L. Xie, and K. H. Johansson, "Human-centered design for safe teleoperation of connected vehicles," In *Proceedings of IFAC Workshop on Cyber-Physical & Human Systems*, 2020. To appear.

Chapter 7: Car Overtaking Application

This chapter develops a solution to the car overtaking problem where an automated vehicle tries to overtake a human-driven vehicle with uncertain motion. The uncertainty in the predicted motion makes the automated overtaking hard due to feasibility issues. To counteract them, we introduce the weak assumption that the predicted velocity of the overtaken vehicle respects a supermartingale, meaning that its velocity is not increasing in expectation during the maneuver. We show that this formulation presents a natural notion of risk. Based on the martingale assumption, we perform a risk-aware reachability analysis by analytically characterizing the predicted collision probability. Then, we design a risk-aware optimal overtaking algorithm with guaranteed levels of collision avoidance. Finally, a simulated example illustrates the effectiveness of the proposed algorithm.

This chapter is based on the following publications.

- Y. Gao, F. J. Jiang, L. Xie, and K. H. Johansson, "Stochastic modeling and optimal control for automated overtaking," In *Proceedings of IEEE Conference on Decision and Control*, 2019, 1273-1278.

- Y. Gao, F. J. Jiang, L. Xie, and K. H. Johansson, "Risk-aware optimal control for automated overtaking with safety guarantee," *IEEE Transactions on Control Systems Technology*. Submitted.

Chapter 8: Conclusions and Future Research

In Chapter 8, we present a summary of the results, and discuss directions for future research.

Other Contributions Not Included in This Thesis

The following contributions have not been included in the thesis, but were developed during the author's Ph.D. studies:

- F. J. Jiang, Y. Gao, L. Xie, and K. H. Johansson, "Ensuring safety for vehicle parking tasks using Hamilton-Jacobi reachability analysis," In *Proceedings of IEEE Conference on Decision and Control*, 2020. To appear.
- Y. Gao, L. Xie, and K. H. Johansson, "Probabilistic characterization of target set and region of attraction for discrete-time control systems," In *Proceedings of IEEE International Conference on Control & Automation*, 2020.
- Y. Gao, P. Yu, D. V. Dimarogonas, K. H. Johansson, and L. Xie, "Self-triggered control for constrained systems via reachability analysis," *Automatica*, 2019, 107, 574-581.
- Y. Gao, S. Wu, K. H. Johansson, L. Shi, and L. Xie, "Stochastic optimal control of dynamic queue systems: a probabilistic perspective," In *Proceedings of International Conference on Control, Automation, Robotics and Vision*, 2018, 837-842.
- L. Dai, Y. Gao, L. Xie, K. H. Johansson, and Y. Xia "Stochastic self-triggered model predictive control for linear systems with probabilistic constraints," *Automatica*, 2018, 92, 9-17.
- Y. Gao, M. Jafarian, K. H. Johansson, and L. Xie, "Distributed freeway ramp metering: optimization on flow speed," In *Proceedings of IEEE Conference on Decision and Control*, 2017, 5654-5659.

Contribution by the Author

The order of authors reflects their contribution to each paper. The first author has the most important contribution, while the authors K. H. Johansson and L. Xie have taken the supervisory role. In all the listed publications, all the authors were actively involved in formulating the problems, developing the solutions, evaluating the results, and writing the paper.

Chapter 2

Mathematical Background

In this chapter, we provide mathematical preliminaries that are used in the remaining parts of this thesis.

2.1 System Models

This section introduces three kinds of systems for modeling discrete-time uncertain autonomous or control systems: transition systems, controlled transition systems, and stochastic control systems.

2.1.1 Transition System

This subsection will define the transition system.

Definition 2.1. *A transition system* TS *is a tuple* $\mathsf{TS} = (\mathbb{S}, \rightarrow, \mathbb{S}_0, \mathcal{AP}, L)$ *consisting of*

- *a set* $\mathbb{S}$ *of states;*
- *a transition relation* $\rightarrow \in \mathbb{S} \times \mathbb{S}$ [1]*;*
- *a set* $\mathbb{S}_0 \subseteq \mathbb{S}$ *of initial states;*
- *a set* $\mathcal{AP}$ *of atomic propositions;*
- *a labelling function* $L : \mathbb{S} \rightarrow 2^{\mathcal{AP}}$.

[1] Here, the transition relation is not a functional relation, which means that for some state x, there may exist two different states x' and x'' such that $x \rightarrow x'$ and $x \rightarrow x''$ hold. For notational simplicity, we use $\rightarrow \in \mathbb{S} \times \mathbb{S}$, rather than $\rightarrow \in \mathbb{S} \times 2^{\mathbb{S}}$. The same claim holds for the controlled transition systems in Section 2.1.2.

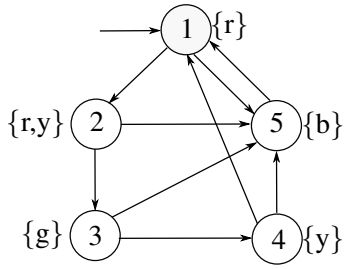


Figure 2.1: A transition system illustrating a "traffic light" example. Labels are shown aside the states. The initial state is denoted by an incoming edge.

Definition 2.2. *A transition system* TS *is said to be finite if* $|\mathbb{S}| < \infty$ *and* $|\mathcal{AP}| < \infty$.

Definition 2.3. *For* $x \in \mathbb{S}$*, the set* $\mathsf{Post}(x)$ *of direct successors of* x *is defined by* $\mathsf{Post}(x) = \{x' \in \mathbb{S} \mid x \to x'\}$.

Definition 2.4. *A transition system* TS *is said to be deterministic if* $|\mathbb{S}_0| = 1$ *and* $|\mathsf{Post}(x)| = 1, \forall x \in \mathbb{S}$.

Definition 2.5. *(Trajectory [2]) For a transition system* TS*, an infinite trajectory* $\boldsymbol{p}$ *starting from* x_0 *is a sequence of states* $\boldsymbol{p} = x_0 x_1 \dots x_k x_{k+1} \dots$ *such that* $\forall k \in \mathbb{N}, x_{k+1} \in \mathsf{Post}(x_k)$.

Denote by $\mathsf{Trajs}(x_0)$ the set of infinite trajectories starting from x_0. Let $\mathsf{Trajs}(\mathsf{TS}) = \cup_{x \in \mathbb{S}_0} \mathsf{Trajs}(x)$. For a trajectory $\boldsymbol{p}$, the k-th state is denoted by $\boldsymbol{p}[k]$, i.e., $\boldsymbol{p}[k] = x_k$ and the k-th prefix is denoted by $\boldsymbol{p}[..k]$, i.e., $\boldsymbol{p}[..k] = x_0 \dots x_k$.

Example 2.1. *A traffic light can be red, green, yellow or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red. An illustration for such a traffic light is shown in Figure 2.1. We can model a traffic light as a finite transition system* $\mathsf{TS} = (\mathbb{S}, \to, \mathbb{S}_0, \mathcal{AP}, L)$*:*

- $\mathbb{S} = \{1, 2, 3, 4, 5\}$;

[2] Notice that a trajectory $\boldsymbol{p} = x_0 x_1 \dots x_k x_{k+1} \dots$ is different from a *trace*, which is the sequence of corresponding sets of atomic propositions, and is denoted by $L(x_0)L(x_1)\dots L(x_k)L(x_{k+1})\dots$.

- $\rightarrow = \{(1,2),(2,3),(3,4),(4,1),(1,5),(2,5),(3,5),(4,5),(5,1)\}$;
- $\mathbb{S}_0 = \{1\}$;
- $\mathcal{AP} = \{r, y, g, b\}$;
- $L = \{1 \rightarrow \{r\}, 2 \rightarrow \{r, y\}, 3 \rightarrow \{g\}, 4 \rightarrow \{y\}, 5 \rightarrow \{b\}\}$.

Remark 2.1. *We can rewrite the following discrete-time autonomous system as an infinite transition system:*

$$\mathsf{S}: \begin{cases} x_{k+1} = f(x_k, w_k), \\ y_k = g(x_k), \end{cases}$$

where $x_k \in \mathbb{R}^{n_x}$, $w_k \in \mathbb{R}^{n_w}$, $y_k \in 2^{\mathcal{O}}$, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \rightarrow \mathbb{R}^{n_x}$, *and* $g : \mathbb{R}^{n_x} \rightarrow 2^{\mathcal{O}}$. *Here,* $\mathcal{O}$ *denotes the set of the observations. At each time instant* k, *the disturbance* w_k *belongs to a compact set* $\mathbb{W} \subset \mathbb{R}^{n_w}$. *Denote by* $\mathsf{Ini} \subseteq \mathbb{R}^{n_x}$ *the set of the initial states. If* $\mathcal{O}$ *is finite, the system* S *can be rewritten as an infinite transition system* $\mathsf{TS}_{\mathsf{S}} = (\mathbb{S}, \rightarrow, \mathbb{S}_0, \mathcal{AP}, L)$ *with*

- $\mathbb{S} = \mathbb{R}^{n_x}$;
- $\forall x, x' \in \mathbb{S}$, $x \rightarrow x'$ *if and only if there exists* $w \in \mathbb{W}$ *such that* $x' = f(x, w)$;
- $\mathbb{S}_0 = \mathsf{Ini}$;
- $\mathcal{AP} = \mathcal{O}$;
- $L = g$.

2.1.2 Controlled Transition System

In this subsection, we will introduce the notion of controlled transition system.

Definition 2.6. *A controlled transition system* CTS *is a tuple* $\mathsf{CTS} = (\mathbb{S}, \mathbb{U}, \rightarrow, \mathbb{S}_0, \mathcal{AP}, L)$ *consisting of*

- *a set* $\mathbb{S}$ *of states;*
- *a set* $\mathbb{U}$ *of control inputs;*
- *a transition relation* $\rightarrow \in \mathbb{S} \times \mathbb{U} \times \mathbb{S}$;

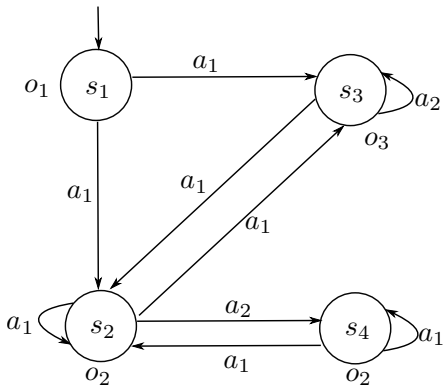


Figure 2.2: Graph description of a controlled transition system.

- *a set $\mathbb{S}_0$ of initial states;*
- *a set $\mathcal{AP}$ of atomic propositions;*
- *a labelling function $L : \mathbb{S} \to 2^{\mathcal{AP}}$.*

Definition 2.7. *A controlled transition system* CTS *is said to be finite if $|\mathbb{S}| < \infty$, $|\mathbb{U}| < \infty$, and $|\mathcal{AP}| < \infty$.*

Definition 2.8. *For $x \in \mathbb{S}$ and $u \in \mathbb{U}$, the set* $\mathsf{Post}(x, u)$ *of direct successors of x under u is defined by* $\mathsf{Post}(x, u) = \{x' \in \mathbb{S} \mid x \xrightarrow{u} x'\}$.

Definition 2.9. *For $x \in \mathbb{S}$, the set $\mathbb{U}(x)$ of admissible control inputs at state x is defined by $\mathbb{U}(x) = \{u \in \mathbb{U} \mid \mathsf{Post}(x, u) \neq \emptyset\}$.*

Definition 2.10. *(Policy) For a controlled transition system* CTS*, a* policy $\boldsymbol{\mu} = \mu_0\mu_1 \dots \mu_k \dots$ *is a sequence of maps $\mu_k : \mathbb{S} \to \mathbb{U}$. Denote by $\mathcal{M}$ the set of all policies.*

Definition 2.11. *(Trajectory) For a controlled transition system* CTS*, an infinite* trajectory $\boldsymbol{p}$ *starting from x_0 under a policy $\boldsymbol{\mu} = \mu_0\mu_1 \dots \mu_k \dots$ is a sequence of states $\boldsymbol{p} = x_0x_1 \dots x_k \dots$ such that $\forall k \in \mathbb{N}$, $x_{k+1} \in \mathsf{Post}(x_k, \mu_k(x_k))$. Denote by* $\mathsf{Trajs}(x_0, \boldsymbol{\mu})$ *the set of infinite trajectories starting from x_0 under $\boldsymbol{\mu}$.*

Example 2.2. *A controlled transition system* $\mathsf{CTS} = (\mathbb{S}, \mathbb{U}, \to, \mathbb{S}_0, \mathcal{AP}, L)$ *is shown in Figure 2.2, where*

- $\mathbb{S} = \{s_1, s_2, s_3, s_4\}$;
- $\mathbb{U} = \{a_1, a_2\}$;
- $\rightarrow = \{(s_1, a_1, s_2), (s_1, a_1, s_3), (s_2, a_1, s_2), (s_2, a_1, s_3), (s_2, a_1, s_3), (s_2, a_2, s_4), (s_3, a_1, s_2), (s_3, a_2, s_3), (s_4, a_1, s_2), (s_4, a_1, s_4)\}$;
- $\mathbb{S}_0 = \{s_1\}$;
- $\mathcal{AP} = \{o_1, o_2, o_3\}$;
- $L = \{s_1 \rightarrow \{o_1\}, s_2 \rightarrow \{o_2\}, s_3 \rightarrow \{o_3\}, s_4 \rightarrow \{o_2\}\}$.

Remark 2.2. *We express the following discrete-time uncertain control system as an infinite controlled transition system:*

$$\mathsf{CS}: \begin{cases} x_{k+1} = f(x_k, u_k, w_k), \\ y_k = g(x_k), \end{cases} \tag{2.1}$$

where $x_k \in \mathbb{R}^{n_x}$ *and* $u_k \in \mathbb{R}^{n_u}$, $w_k \in \mathbb{R}^{n_w}$, $y_k \in 2^{\mathcal{O}}$, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \rightarrow \mathbb{R}^{n_x}$, *and* $g : \mathbb{R}^{n_x} \rightarrow 2^{\mathcal{O}}$. *Here,* $\mathcal{O}$ *denotes the set of the observations. At each time instant* k, *the control input* u_k *is constrained by a compact set* $\mathbb{U}_{\mathsf{CS}} \subset \mathbb{R}^{n_u}$ *and the disturbance* w_k *belongs to a compact set* $\mathbb{W} \subset \mathbb{R}^{n_w}$. *Denote by* $\mathsf{Ini} \subseteq \mathbb{R}^{n_x}$ *the set of the initial states. If the set* $\mathcal{O}$ *is finite, system* CS *can be rewritten as an infinite controlled transition system,* $\mathsf{CTS}_{\mathsf{CS}} = (\mathbb{S}, \mathbb{U}, \rightarrow, \mathbb{S}_0, \mathcal{AP}, L)$ *where*

- $\mathbb{S} = \mathbb{R}^{n_x}$;
- $\mathbb{U} = \mathbb{U}_{\mathsf{CS}}$;
- $\forall x, x' \in \mathbb{S}$ *and* $\forall u \in \mathbb{U}$, $x \xrightarrow{u} x'$ *if and only if there exists* $w \in \mathbb{W}$ *such that* $x' = f(x, u, w)$;
- $\mathbb{S}_0 = \mathsf{Ini}$;
- $\mathcal{AP} = \mathcal{O}$;
- $L = g$.

The systems considered in Chapters 4, 6, and 7 are written in the form of (2.1). Thus, they are some special cases of the controlled transition system CTS.

2.1.3 Stochastic Control System

When refining the transition relation of CTS to be a stochastic kernel, we can define a stochastic control system described by a Markov controlled process $\mathsf{SCS} = (\mathbb{S}, \mathbb{U}, T)$, where

- $\mathbb{S}$ is a state space endowed with a Borel σ-algebra $\mathcal{B}(\mathbb{S})$;
- $\mathbb{U}$ is a compact control space endowed with a Borel σ-algebra $\mathcal{B}(\mathbb{U})$;
- $T : \mathcal{B}(\mathbb{S}) \times \mathbb{S} \times \mathbb{U} \to \mathbb{R}$ is a Borel-measurable stochastic kernel given $\mathbb{S} \times \mathbb{U}$, which assigns to each $x \in \mathbb{S}$ and $u \in \mathbb{U}$ a probability measure on the Borel space $(\mathbb{S}, \mathcal{B}(\mathbb{S}))$: $T(\cdot|x, u)$.

Let us denote by $\mathbb{U}_x$ the set of the admissible control actions for each $x \in \mathbb{S}$. Assume that $\mathbb{U}_x$ is nonempty for each $x \in \mathbb{S}$.

Consider a finite horizon $N \in \mathbb{N}$. A policy is said to be a Markov policy if the control inputs are only dependent on the current state, i.e., $u_k = \mu_k(x_k)$.

Definition 2.12. *(Markov Policy) A Markov policy $\boldsymbol{\mu}$ for system* SCS *is a sequence $\boldsymbol{\mu} = (\mu_0, \mu_1, \ldots, \mu_{N-1})$ of universally measurable maps*

$$\mu_k : \mathbb{S} \to \mathbb{U}, \forall k \in \mathbb{N}_{[0,N-1]}.$$

Remark 2.3. *Given a space $\mathbb{Y}$, a subset $\mathbb{A}$ in this space is universally measurable if it is measurable with respect to every complete probability measure on $\mathbb{Y}$ that measures all Borel sets in $\mathcal{B}(\mathbb{Y})$. A function $\mu : \mathbb{Y} \to \mathbb{W}$ is universally measurable if $\mu^{-1}(\mathbb{A})$ is universally measurable in $\mathbb{Y}$ for every $\mathbb{A} \in \mathcal{B}(\mathbb{W})$. As stated in [51], [89], the condition of universal measurability is weaker than the condition of Borel measurability for showing the existence of a solution to a stochastic optimal problem. Roughly speaking, this is because the projections of measurable sets are analytic sets and analytic sets are universally measurable but not always Borel measurable [89], [90].*

Remark 2.4. *For a large class of stochastic optimal control problems, Markov policies are sufficient to characterize the optimal policy [89]. Furthermore, since a randomized Markov policy does not increase the largest probability that the states remain in a set, we focus on deterministic Markov policies in the following.*

We denote the set of Markov policies as $\mathcal{M}$. Consider a set $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$. Given an initial state $x_0 \in \mathbb{S}$ and a Markov policy $\boldsymbol{\mu} \in \mathcal{M}$, a trajectory is a

sequence of states $(x_0, x_1, \ldots, x_N)$. Introduce the probability with which the state x_k will remain within $\mathbb{Q}$ for all $k \in \mathbb{N}_{[0,N]}$:

$$p^{\boldsymbol{\mu}}_{N,\mathbb{Q}}(x_0) = \Pr\{\forall k \in \mathbb{N}_{[0,N]}, x_k \in \mathbb{Q}\}.$$

Let $p^*_{N,\mathbb{Q}}(x) = \sup_{\boldsymbol{\mu}\in\mathcal{M}} p^{\boldsymbol{\mu}}_{N,\mathbb{Q}}(x), \forall x \in \mathbb{Q}$. We call $p^*_{N,\mathbb{Q}}(x)$ the N-step invariance probability at x in the set $\mathbb{Q}$. We will detail how to compute $p^*_{N,\mathbb{Q}}(x)$ in Section 2.4.

Extending the finite horizon to infinite horizon, we need to introduce stationary policies.

Definition 2.13. *(Stationary Policy) A Markov policy $\boldsymbol{\mu} \in \mathcal{M}$ is said to be stationary if $\boldsymbol{\mu} = (\bar{\mu}, \bar{\mu}, \ldots)$ with $\bar{\mu} : \mathbb{S} \to \mathbb{U}$ universally measurable.*

Given an initial state $x_0 \in \mathbb{S}$ and a stationary policy $\boldsymbol{\mu} \in \mathcal{M}$, a trajectory is denoted by a sequence of states $(x_0, x_1, \ldots)$. We introduce the probability with which the state x_k will remain within $\mathbb{Q}$ for all $k \in \mathbb{N}_{\geq 0}$:

$$p^{\boldsymbol{\mu}}_{\infty,\mathbb{Q}}(x_0) = \Pr\{\forall k \in \mathbb{N}, x_k \in \mathbb{Q}\}.$$

Denote $p^*_{\infty,\mathbb{Q}}(x_0) = \sup_{\boldsymbol{\mu}\in\mathcal{M}} p^{\boldsymbol{\mu}}_{\infty,\mathbb{Q}}(x_0)$. We call $p^*_{\infty,\mathbb{Q}}(x)$ the infinite-horizon invariance probability at x in the set $\mathbb{Q}$. We will also detail how to compute $p^*_{\infty,\mathbb{Q}}(x)$ in Section 2.4.

Example 2.3. *A stochastic control system $\mathsf{SCS} = (\mathbb{S}, \mathbb{U}, T)$ modeled by a finite Markov Decision Process is given as follows*

- $\mathbb{S} = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$;
- $\mathbb{U} = \{a_1, a_2, a_3\}$;
- $T(s_1|s_1, a_1) = 0.20, T(s_2|s_1, a_1) = 0.80, T(s_2|s_1, a_2) = 0.90,$
  $T(s_3|s_1, a_2) = 0.10, T(s_1|s_2, a_1) = 0.90, T(s_2|s_2, a_1) = 0.10,$
  $T(s_4|s_2, a_3) = 1.00, T(s_4|s_3, a_2) = 0.49, T(s_5|s_3, a_2) = 0.49,$
  $T(s_6|s_3, a_2) = 0.02, T(s_3|s_3, a_3) = 0.99, T(s_6|s_3, a_3) = 0.01,$
  $T(s_3|s_4, a_1) = 0.50, T(s_4|s_4, a_1) = 0.50, T(s_2|s_4, a_2) = 0.80,$
  $T(s_5|s_4, a_2) = 0.20, T(s_3|s_5, a_1) = 0.90, T(s_5|s_5, a_1) = 0.10,$
  $T(s_6|s_5, a_3) = 1.00, T(s_2|s_6, a_2) = 1.00, T(s_5|s_6, a_3) = 0.80,$
  $T(s_7|s_6, a_3) = 0.20, T(s_6|s_7, a_1) = 0.30, T(s_7|s_7, a_1) = 0.70,$
  $T(s_4|s_7, a_2) = 0.90, T(s_5|s_7, a_2) = 0.10, T(s_3|s_7, a_3) = 1.00.$

2.2 Reachability Analysis

This section provides reachability analysis for transition systems and controlled transition systems, respectively.

2.2.1 Reachability Analysis of Transition Systems

This subsection specifies the reachability analysis for a transition system TS. We first define the minimal reachable set and the maximal reachable set.

Definition 2.14. *Consider a transition system* TS *and two sets* $\Omega_1, \Omega_2 \subseteq \mathbb{S}$. *The* k*-step minimal reachable set from* Ω_1 *to* Ω_2 *is defined as*

$$\mathcal{R}^{\mathrm{m}}(\Omega_1, \Omega_2, k) = \Big\{ x_0 \in \mathbb{S} \mid \forall \boldsymbol{p} \in \mathsf{Trajs}(x_0), \text{ s.t.,} \\ \boldsymbol{p}[..k] = x_0 \dots x_k, \forall i \in \mathbb{N}_{[0,k-1]}, x_i \in \Omega_1, x_k \in \Omega_2 \Big\}.$$

The minimal reachable set from Ω_1 *to* Ω_2 *is defined as*

$$\mathcal{R}^{\mathrm{m}}(\Omega_1, \Omega_2) = \bigcup_{k \in \mathbb{N}} \mathcal{R}^{\mathrm{m}}(\Omega_1, \Omega_2, k).$$

The minimal reachable set can also be considered up to a finite time: $\bar{\mathcal{R}}^{\mathrm{m}}(\Omega_1, \Omega_2, k) = \bigcup_{i \in \mathbb{N}_{[0,k]}} \mathcal{R}^{\mathrm{m}}(\Omega_1, \Omega_2, i)$. An illustration of minimal reachable sets is shown in Figure 2.3(a).

Lemma 2.1. *For two sets* $\Omega_1, \Omega_2 \subseteq \mathbb{S}$*, define*

$$\mathbb{Q}_{k+1} = \{ x \in \Omega_1 \mid \mathsf{Post}(x) \subseteq \mathbb{Q}_k \} \cup \mathbb{Q}_k, \\ \mathbb{Q}_0 = \Omega_2.$$

Then, $\mathcal{R}^{\mathrm{m}}(\Omega_1, \Omega_2) = \lim_{k \to \infty} \mathbb{Q}_k$.

Proof. From Definition 2.14, it is easy to see that

$$\mathbb{Q}_k = \bigcup_{i \in \mathbb{N}_{[0,k]}} \mathcal{R}^{\mathrm{m}}(\Omega_1, \Omega_2, i).$$

It follows from the Knaster-Tarski Theorem [91] that $\lim_{k \to \infty} \mathbb{Q}_k$ exists and is a fixed point to the monotone function $F(\mathbb{P}) = \{ x \in \Omega_1 \mid \mathsf{Post}(x) \subseteq \mathbb{P} \} \cup \mathbb{P}$. Thus, we have that $\mathcal{R}^{\mathrm{m}}(\Omega_1, \Omega_2) = \lim_{k \to \infty} \mathbb{Q}_k$. □

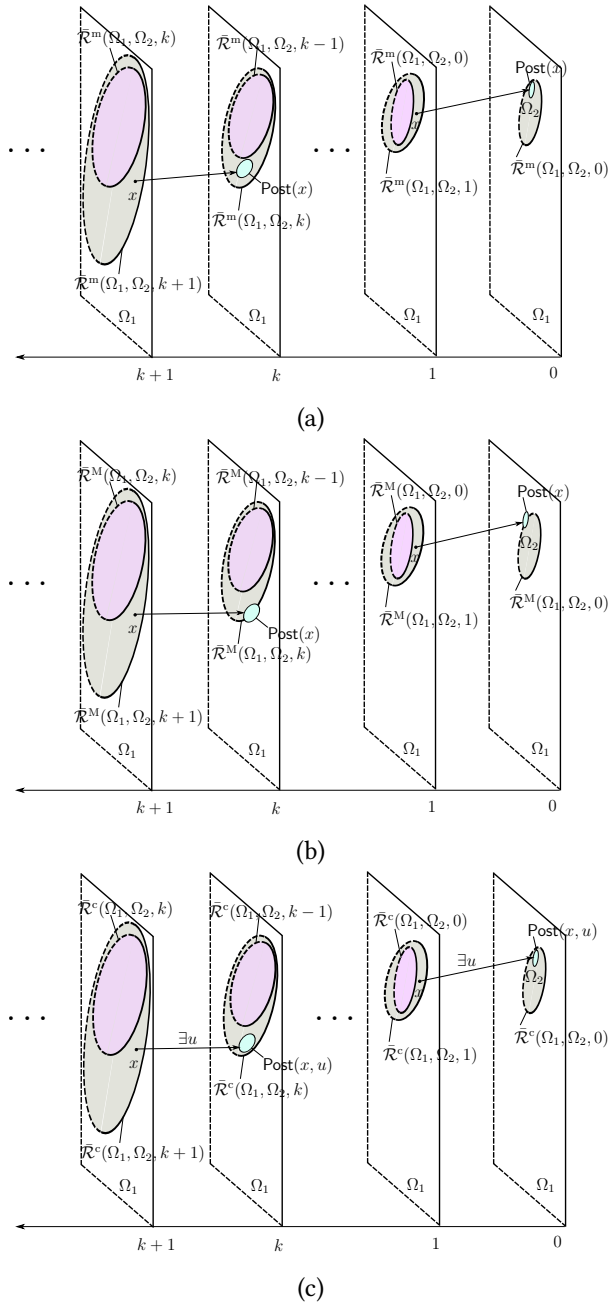


Figure 2.3: Illustration of (a) minimal reachable sets, (b) maximal reachable sets, and (c) controlled reachable sets.

Definition 2.15. *Consider a transition system* TS *and two sets* $\Omega_1, \Omega_2 \subseteq \mathbb{S}$. *The* k*-step maximal reachable set from* Ω_1 *to* Ω_2 *is defined as*

$$\mathcal{R}^{\mathrm{M}}(\Omega_1, \Omega_2, k) = \Big\{ x_0 \in \mathbb{S} \mid \exists \boldsymbol{p} \in \mathsf{Trajs}(x_0), \text{ s.t.,} \\ \boldsymbol{p}[..k] = x_0 \dots x_N, \forall i \in \mathbb{N}_{[0,k-1]}, x_i \in \Omega_1, x_k \in \Omega_2 \Big\}.$$

The maximal reachable set from Ω_1 *to* Ω_2 *is defined as*

$$\mathcal{R}^{\mathrm{M}}(\Omega_1, \Omega_2) = \bigcup_{k \in \mathbb{N}} \mathcal{R}^{\mathrm{M}}(\Omega_1, \Omega_2, k).$$

Similarly, the maximal reachable set can also be considered up to a finite time: $\bar{\mathcal{R}}^{\mathrm{M}}(\Omega_1, \Omega_2, k) = \bigcup_{i \in \mathbb{N}_{[0,k]}} \mathcal{R}^{\mathrm{M}}(\Omega_1, \Omega_2, i)$. An illustration of maximal reachable sets is shown in Figure 2.3(b).

Lemma 2.2. *For two sets* $\Omega_1, \Omega_2 \subseteq \mathbb{S}$, *define*

$$\mathbb{Q}_{k+1} = \{ x \in \Omega_1 \mid \mathsf{Post}(x) \cap \mathbb{Q}_k \neq \emptyset \} \cup \mathbb{Q}_k, \\ \mathbb{Q}_0 = \Omega_2.$$

Then, $\mathcal{R}^{\mathrm{M}}(\Omega_1, \Omega_2) = \lim_{k \to \infty} \mathbb{Q}_k$.

Proof. Similar to the proof of Lemma 2.1. □

We define the robust invariant set and the invariant set in the following.

Definition 2.16. *A set* $\Omega_f \subseteq \mathbb{S}$ *is said to be a robust invariant set of a transition system* TS *if for any* $x \in \Omega_f$, $\mathsf{Post}(x) \subseteq \Omega_f$.

Definition 2.17. *For a set* $\Omega \subseteq \mathbb{S}$, *a set* $\mathcal{RI}(\Omega) \subseteq \mathbb{S}$ *is said to be the largest robust invariant set in* $\mathbb{S}$ *if each robust invariant set* $\Omega_f \subseteq \Omega$ *satisfies* $\Omega_f \subseteq \mathcal{RI}(\Omega)$.

An illustration of robust invariant set is shown in Figure 2.4(a).

Lemma 2.3. *For a set* $\Omega \subseteq \mathbb{S}$, *define*

$$\mathbb{Q}_{k+1} = \{ x \in \mathbb{Q}_k \mid \mathsf{Post}(x) \subseteq \mathbb{Q}_k \}, \\ \mathbb{Q}_0 = \Omega.$$

Then, $\mathcal{RI}(\Omega) = \lim_{k \to \infty} \mathbb{Q}_k$.

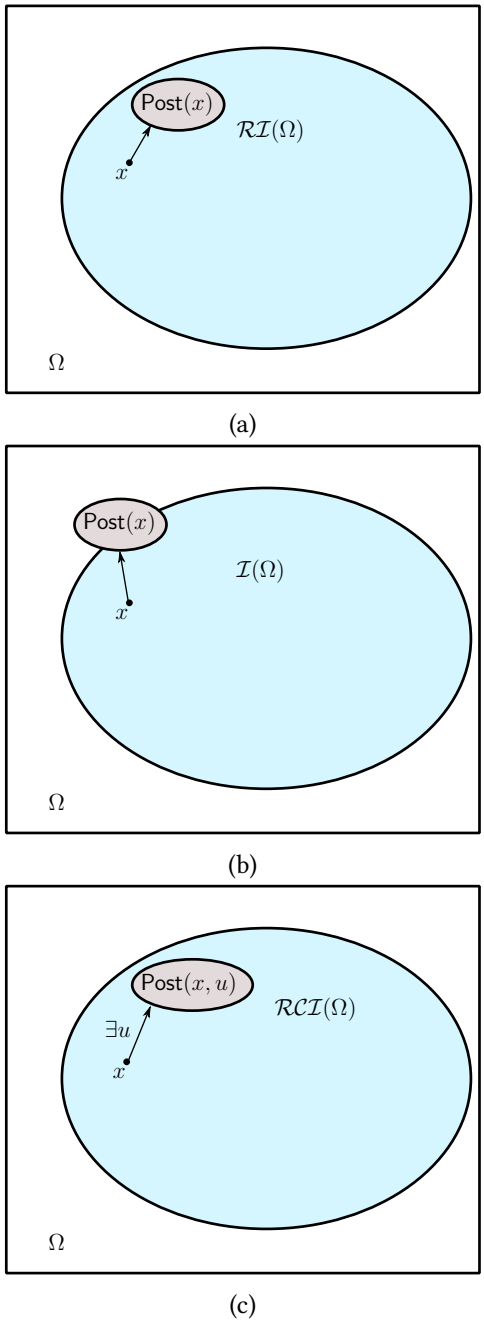


Figure 2.4: Illustration of (a) robust invariant set, (b) invariant set, and (c) robust controlled invariant set.

Proof. It follows again from the Knaster-Tarski Theorem [91] that $\lim_{k\to\infty} \mathbb{Q}_k$ exists and it is a fixed point to the monotone function $F(\mathbb{P}) = \{x \in \mathbb{P} \mid \mathsf{Post}(x) \subseteq \mathbb{P}\} \cap \mathbb{P}$. Thus, we have that $\mathcal{RI}(\Omega) = \lim_{k\to\infty} \mathbb{Q}_k$. □

Definition 2.18. *A set $\Omega_f \subseteq \mathbb{S}$ is said to be an invariant set of a transition system* TS *if for any $x \in \Omega_f$, $\mathsf{Post}(x) \cap \Omega_f \neq \emptyset$.*

Definition 2.19. *For a set $\Omega \subseteq \mathbb{S}$, a set $\mathcal{I}(\Omega) \subseteq \mathbb{S}$ is said to be the largest invariant set in $\mathbb{S}$ if each invariant set $\Omega_f \subseteq \Omega$ satisfies $\Omega_f \subseteq \mathcal{I}(\Omega)$.*

An illustration of invariant set is shown in Figure 2.4(b).

Lemma 2.4. *For a set $\Omega \subseteq \mathbb{S}$, define*

$$\begin{aligned} \mathbb{Q}_{k+1} &= \{x \in \mathbb{Q}_k \mid \mathsf{Post}(x) \cap \mathbb{Q}_k \neq \emptyset\}, \\ \mathbb{Q}_0 &= \Omega. \end{aligned}$$

Then, $\mathcal{I}(\Omega) = \lim_{k\to\infty} \mathbb{Q}_k$.

Proof. Similar to the proof of Lemma 2.3. □

We can understand the reachable sets and invariant sets defined above as maps $\mathcal{R}^{\mathrm{m}} : 2^{\mathbb{S}} \times 2^{\mathbb{S}} \to 2^{\mathbb{S}}$, $\mathcal{R}^{\mathrm{M}} : 2^{\mathbb{S}} \times 2^{\mathbb{S}} \to 2^{\mathbb{S}}$, $\mathcal{RI} : 2^{\mathbb{S}} \to 2^{\mathbb{S}}$, and $\mathcal{I} : 2^{\mathbb{S}} \to 2^{\mathbb{S}}$, respectively. In Chapter 5, we will refer to them as "reachability operators".

2.2.2 Reachability Analysis of Controlled Transition System

This subsection will specify the controlled reachability analysis of a controlled transition system CTS.

Definition 2.20. *Consider a controlled transition system* CTS *and two sets $\Omega_1, \Omega_2 \subseteq \mathbb{S}$. The k-step controlled reachable set from Ω_1 to Ω_2 is defined as*

$$\begin{aligned} \mathcal{R}^{\mathrm{c}}(\Omega_1, \Omega_2, k) = \Big\{ & x_0 \in \mathbb{S} \mid \exists \boldsymbol{\mu} \in \mathcal{M} \text{ s.t., } \forall \boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu}), \\ & \boldsymbol{p}[..k] = x_0 \dots x_k, \forall i \in \mathbb{N}_{[0,k-1]}, x_i \in \Omega_1, x_k \in \Omega_2 \Big\}. \end{aligned}$$

The controlled reachable set from Ω_1 to Ω_2 is defined as

$$\mathcal{R}^{\mathrm{c}}(\Omega_1, \Omega_2) = \bigcup_{k\in\mathbb{N}} \mathcal{R}^{\mathrm{c}}(\Omega_1, \Omega_2, k).$$

The controlled reachable set can also be considered up to a finite time: $\bar{\mathcal{R}}^{\mathrm{c}}(\Omega_1, \Omega_2, k) = \bigcup_{i \in \mathbb{N}_{[0,k]}} \mathcal{R}^{\mathrm{c}}(\Omega_1, \Omega_2, i)$. An illustration of maximal reachable sets is shown in Figure 2.3(c).

Lemma 2.5. *For two sets $\Omega_1, \Omega_2 \subseteq \mathbb{S}$, define*

$$\begin{aligned} \mathbb{Q}_{k+1} &= \{x \in \Omega_1 \mid \exists u \in \mathbb{U}(x), \mathsf{Post}(x, u) \subseteq \mathbb{Q}_k\} \cup \mathbb{Q}_k, \\ \mathbb{Q}_0 &= \Omega_2. \end{aligned}$$

Then, $\mathcal{R}^{\mathrm{c}}(\Omega_1, \Omega_2) = \lim_{k \to \infty} \mathbb{Q}_k$.

Proof. Similar to the proof of Lemma 2.1. □

Definition 2.21. *A set $\Omega_f \subseteq \mathbb{S}$ is said to be a robust controlled invariant set (RCIS) of a transition system TS if for any $x \in \Omega_f$, there exists $u \in \mathbb{U}(x)$ such that $\mathsf{Post}(x, u) \subseteq \Omega_f$.*

Definition 2.22. *For a set $\Omega \subseteq \mathbb{S}$, a set $\mathcal{RCI}(\Omega) \subseteq \mathbb{S}$ is said to be the largest RCIS in $\mathbb{S}$ if each RCIS $\Omega_f \subseteq \Omega$ satisfies $\Omega_f \subseteq \mathcal{RCI}(\Omega)$.*

An illustration of RCIS is shown in Figure 2.4(c).

Lemma 2.6. *For a set $\Omega \subseteq \mathbb{S}$, define*

$$\begin{aligned} \mathbb{Q}_{k+1} &= \{x \in \mathbb{Q}_k \mid \exists u \in \mathbb{U}(x), \mathsf{Post}(x, u) \subseteq \mathbb{Q}_k\}, \\ \mathbb{Q}_0 &= \Omega. \end{aligned}$$

Then, $\mathcal{RCI}(\Omega) = \lim_{k \to \infty} \mathbb{Q}_k$.

Proof. Similar to the proof of Lemma 2.3. □

The definitions of controlled reachable sets and RCISs provide us a way to synthesize the feasible control set, which is detailed in Chapter 5. We treat the maps $\mathcal{R}^{\mathrm{c}} : 2^{\mathbb{S}} \times 2^{\mathbb{S}} \to 2^{\mathbb{S}}$ and $\mathcal{RCI} : 2^{\mathbb{S}} \to 2^{\mathbb{S}}$ as the reachability operators.

2.3 Temporal Logic

An LTL formula is defined over a finite set of atomic propositions $\mathcal{AP}$ and both logic and temporal operators. The syntax of LTL can be described as:

$$\varphi ::= \mathrm{true} \mid a \in \mathcal{AP} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2,$$

where $\bigcirc$ and U denote the "next" and "until" operators, respectively. By using the negation and conjunction operators, we can define disjunction: $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$. By employing the until operator, we can define: (1) eventually, $\Diamond\varphi = \text{true} \cup \varphi$; (2) always, $\Box\varphi = \neg\Diamond\neg\varphi$; and (3) weak-until, $\varphi_1 \mathsf{W} \varphi_2 = \varphi_1 \mathsf{U} \varphi_2 \vee \Box\varphi_1$.

Definition 2.23. *(LTL semantics) For an LTL formula φ and a trajectory $\boldsymbol{p}$, the satisfaction relation $\boldsymbol{p} \vDash \varphi$ is defined as*

$$
\begin{aligned}
&\boldsymbol{p} \vDash a \Leftrightarrow a \in L(x_0),\\
&\boldsymbol{p} \vDash \neg a \Leftrightarrow a \notin L(x_0),\\
&\boldsymbol{p} \vDash \varphi_1 \wedge \varphi_2 \Leftrightarrow \boldsymbol{p} \vDash \varphi_1 \wedge \boldsymbol{p} \vDash \varphi_2,\\
&\boldsymbol{p} \vDash \varphi_1 \vee \varphi_2 \Leftrightarrow \boldsymbol{p} \vDash \varphi_1 \vee \boldsymbol{p} \vDash \varphi_2,\\
&\boldsymbol{p} \vDash \bigcirc\varphi \Leftrightarrow \boldsymbol{p}[1..] \vDash \varphi,\\
&\boldsymbol{p} \vDash \varphi_1 \mathsf{U} \varphi_2 \Leftrightarrow \exists j \in \mathbb{N} \text{ s.t. } \begin{cases} \boldsymbol{p}[j..] \vDash \varphi_2, \\ \forall i \in \mathbb{N}_{[0,j-1]}, \boldsymbol{p}[i..] \vDash \varphi_1, \end{cases}\\
&\boldsymbol{p} \vDash \Diamond\varphi \Leftrightarrow \exists j \in \mathbb{N}, \text{ s.t. } \boldsymbol{p}[j..] \vDash \varphi,\\
&\boldsymbol{p} \vDash \Box\varphi \Leftrightarrow \forall j \in \mathbb{N}, \text{ s.t. } \boldsymbol{p}[j..] \vDash \varphi,\\
&\boldsymbol{p} \vDash \varphi_1 \mathsf{W} \varphi_2 \Leftrightarrow \begin{cases} \forall j \in \mathbb{N}, \boldsymbol{p}[j..] \vDash \varphi_1, \text{ or} \\ \exists j \in \mathbb{N} \text{ s.t. } \begin{cases} \boldsymbol{p}[j..] \vDash \varphi_2, \\ \forall i \in \mathbb{N}_{[0,j-1]}, \boldsymbol{p}[i..] \vDash \varphi_1, \end{cases} \end{cases}
\end{aligned}
$$

where $a \in \mathcal{AP}$.

Definition 2.24. *Consider a transition system TS and an LTL formula φ. The semantics of the universal form of φ, denoted by $\forall\varphi$, is*

$$x_0 \vDash \forall\varphi \Leftrightarrow \forall \boldsymbol{p} \in \mathsf{Trajs}(x_0), \boldsymbol{p} \vDash \varphi.$$

The semantics of the existential form of φ, denoted by $\exists\varphi$, is

$$x_0 \vDash \exists\varphi \Leftrightarrow \exists \boldsymbol{p} \in \mathsf{Trajs}(x_0), \boldsymbol{p} \vDash \varphi.$$

2.4 Dynamic Programming

This section will show how to use dynamic programming (DP) for computing the invariance probability defined in Section 2.1.3.

Consider a stochastic control system $\mathsf{SCS} = (\mathbb{S}, \mathbb{U}, T)$ and a nonempty set $\mathbb{Q} \subseteq \mathbb{S}$. We first solve the finite-horizon optimal control problem:

$$p^*_{N,\mathbb{Q}}(x) = \sup_{\boldsymbol{\mu} \in \mathcal{M}} p^{\boldsymbol{\mu}}_{N,\mathbb{Q}}(x), \forall x \in \mathbb{Q}.$$

Following the DP in [51], define the value function $V^*_{k,\mathbb{Q}} : \mathbb{S} \to [0,1], k = 0, 1, \ldots, N$, by the backward recursion:

$$V^*_{k,\mathbb{Q}}(x) = \sup_{u \in \mathbb{U}} \mathbb{1}_{\mathbb{Q}}(x) \int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y) T(dy|x,u), x \in \mathbb{S}, \tag{2.2}$$

with initialization $V^*_{N,\mathbb{Q}}(x) = 1, x \in \mathbb{Q}$.

Assumption 2.1. *The set*

$$\mathbb{U}_k(x,\lambda) = \left\{ u \in \mathbb{U} \mid \int_{\mathbb{S}} V^*_{k+1,\mathbb{Q}}(y) T(dy|x,u) \geq \lambda \right\}$$

is compact for all $x \in \mathbb{Q}$, $\lambda \in \mathbb{R}$, and $k \in \mathbb{N}_{[0,N-1]}$.

Lemma 2.7. *[51] For all $x \in \mathbb{Q}$, $p^*_{N,\mathbb{Q}}(x) = V^*_{0,\mathbb{Q}}(x)$. If Assumption 2.1 holds, the optimal Markov policy $\boldsymbol{\mu}^*_{\mathbb{Q}} = (\mu^*_{0,\mathbb{Q}}, \mu^*_{1,\mathbb{Q}}, \ldots, \mu^*_{N-1,\mathbb{Q}})$ exists and is given by*

$$\mu^*_{k,\mathbb{Q}}(x) = \arg\sup_{u \in \mathbb{U}} \mathbb{1}_{\mathbb{Q}}(x) \int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y) T(dy|x,u),\ x \in \mathbb{Q}, k \in \mathbb{N}_{[0,N-1]}.$$

Next we solve the infinite-horizon optimal control problem:

$$p^*_{\infty,\mathbb{Q}}(x) = \sup_{\boldsymbol{\mu} \in \mathcal{M}} p^{\boldsymbol{\mu}}_{\infty,\mathbb{Q}}(x), \forall x \in \mathbb{Q}.$$

Define the value function $G^*_{k,\mathbb{Q}} : \mathbb{S} \to [0,1], k \in \mathbb{N}_{\geq 0}$, through the forward recursion:

$$G^*_{k+1,\mathbb{Q}}(x) = \sup_{u \in \mathbb{U}} \mathbb{1}_{\mathbb{Q}}(x) \int_{\mathbb{Q}} G^*_{k,\mathbb{Q}}(y) T(dy|x,u), x \in \mathbb{S}, \tag{2.3}$$

initialized with $G^*_{0,\mathbb{Q}}(x) = 1, x \in \mathbb{Q}$.

Assumption 2.2. *There exists a $\bar{k} \geq 0$ such that the set*

$$\mathbb{U}_k(x,\lambda) = \left\{ u \in \mathbb{U} \mid \int_{\mathbb{S}} G^*_{k,\mathbb{Q}}(y) T(dy|x,u) \geq \lambda \right\}$$

is compact for all $x \in \mathbb{Q}$, $\lambda \in \mathbb{R}$, and $k \in \mathbb{N}_{\geq \bar{k}}$.

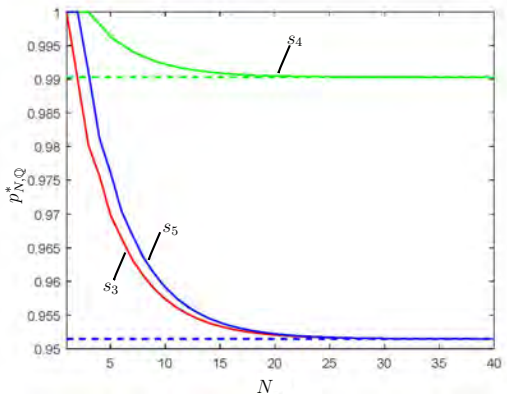


Figure 2.5: The probability $p^*_{N,\mathbb{Q}}(x)$ for $x = s_3, s_4, s_5$.

Lemma 2.8. *[51] Suppose that Assumption 2.2 holds. Then, for all $x \in \mathbb{Q}$, the limit $G^*_{\infty,\mathbb{Q}}(x)$ exists and satisfies*

$$G^*_{\infty,\mathbb{Q}}(x) = \sup_{u\in\mathbb{U}} \mathbb{1}_{\mathbb{Q}}(x) \int_{\mathbb{Q}} G^*_{\infty,\mathbb{Q}}(y)T(dy|x,u)), \tag{2.4}$$

and $p^*_{\infty,\mathbb{Q}}(x) = G^*_{\infty,\mathbb{Q}}(x)$. Furthermore, an optimal stationary policy $\boldsymbol{\mu}^*_{\mathbb{Q}} = (\bar{\mu}^*_{\mathbb{Q}}, \bar{\mu}^*_{\mathbb{Q}}, \ldots)$ exists and is given by

$$\bar{\mu}^*_{\mathbb{Q}}(x) = \arg\sup_{u\in\mathbb{U}} \mathbb{1}_{\mathbb{Q}}(x) \int_{\mathbb{Q}} G^*_{\infty,\mathbb{Q}}(y)T(dy|x,u), x \in \mathbb{Q}.$$

Example 2.4. *Consider the stochastic control system in Example 2.3. Given a set $\mathbb{Q}$ is $\{s_1, s_2, s_3, s_4, s_5\}$, we compute the N-step invariance probability $p^*_{N,\mathbb{Q}}(\cdot)$ with $N = 5$ and the infinite-horizon invariance probability $p^*_{\infty,\mathbb{Q}}(\cdot)$*

$$\begin{cases} p^*_{5,\mathbb{Q}}(s_1) = 1.0000, \\ p^*_{5,\mathbb{Q}}(s_2) = 1.0000, \\ p^*_{5,\mathbb{Q}}(s_3) = 0.9665, \\ p^*_{5,\mathbb{Q}}(s_4) = 0.9952, \\ p^*_{5,\mathbb{Q}}(s_5) = 0.9705, \end{cases} \quad \textit{and} \quad \begin{cases} p^*_{\infty,\mathbb{Q}}(s_1) = 1.0000, \\ p^*_{\infty,\mathbb{Q}}(s_2) = 1.0000, \\ p^*_{\infty,\mathbb{Q}}(s_3) = 0.9515, \\ p^*_{\infty,\mathbb{Q}}(s_4) = 0.9903, \\ p^*_{\infty,\mathbb{Q}}(s_5) = 0.9515. \end{cases}$$

It is shown in Figure 2.5 that $p^*_{N,\mathbb{Q}}$ converges to $p^*_{\infty,\mathbb{Q}}$ as $N \to \infty$ element-wisely.

Chapter 3

Computation of Probabilistic Controlled Invariant Sets

This chapter investigates stochastic invariance for control systems through probabilistic controlled invariant sets (PCISs). As a natural complement to robust controlled invariant sets (RCISs), we propose finite- and infinite-horizon PCISs, and explore their relation to RICSs. We design iterative algorithms to compute the PCIS within a given set. For systems with discrete spaces, the computations of the finite- and infinite-horizon PCISs at each iteration are based on linear programming (LP) and mixed integer linear programming (MILP), respectively. The algorithms are computationally tractable and terminate in a finite number of steps. For systems with continuous spaces, we show how to discretize these spaces and prove the convergence of the approximation when computing the finite-horizon PCISs. In addition, it is shown that an infinite-horizon PCIS can be computed by the stochastic backward reachable set from the RCIS contained in it. These PCIS algorithms are applicable to practical control systems. Simulations are given to illustrate the effectiveness of the theoretical results for motion planning.

The remainder of this chapter is organized as follows. Section 3.1 reviews the related work. Section 3.2 presents the definition, properties, and computation algorithms of finite-horizon PCISs. Section 3.3 extends the results to the infinite-horizon case. Examples in Section 3.4 illustrate the effectiveness of our approach. Section 3.5 concludes this chapter.

3.1 Introduction

3.1.1 Motivation and Related Work

Invariance is a fundamental concept in systems and control [43], [46], [47]. A controlled invariant set captures the region where the states can be maintained by some admissible control inputs. RCISs are defined for control systems with bounded external disturbances and address the invariance despite any realization of the disturbances. In the past decades, there have been lots of research results on RCISs and their computations [26]–[28]. This chapter studies PCISs, which is a natural complement to RCISs suitable in many applications. A PCIS is a set within which the controller is able to keep the system state with a certain probability. Such sets not only alleviate the inherent conservatism of RCISs by allowing probabilistic violations but also enlarge the applications of RCISs by being able to address unbounded disturbances. The study of PCISs is motivated by safety-critical control [92], stochastic model predictive control (MPC) [93], [94], reliable control [95], [96], and relevant applications, e.g., air traffic management systems [13], [14] and motion planning [97].

When computing RCISs, one essential component in iterative approaches is to compute the robust backward reachable set, in which each state can be steered to the current set by an admissible input for all possible uncertainties [26]–[28]. The PCIS computation in this chapter follows the same idea, but the robust backward reachable set is replaced with the stochastic backward reachable sets which require different mathematical tools. Some challenges related to such an approach should be highlighted: (i) how to make it tractable to compute the stochastic backward reachable set, in particular for systems with continuous spaces; (ii) how to mitigate the conservatism when characterizing the stochastic backward reachable set subject to the prescribed probability; (iii) how to guarantee convergence of the iterations.

Controlled invariant sets have recently been extended to stochastic systems. In [98], a target set, which is similar to the PCIS of this chapter, is used to define stabilization in probability. In [95], a reliable control set, another similar notion to a PCIS, is used to guarantee the reliability of Markov-jump linear systems. The reliability is further studied for such systems with bounded disturbances in [96]. A definition of PCIS for nonlinear systems is provided in [99] by using reachability analysis. It is later applied to portfolio optimization [100]. Another definition of probabilistic invariance originates from stochastic MPC [101] and captures one-step invariance. In [101], an el-

lipsoidal approximation is given for linear systems with specific uncertainty structure. Similar invariant sets are used in [102] to construct a convex lifting function for linear stochastic control systems. A definition of a probabilistic invariant set is proposed in [52], [53] for linear stochastic systems without control inputs. This definition captures the probabilistic inclusion of the state at each time instant. A recent work [103] explores the correspondence between probabilistic and robust invariant sets for linear systems. In [52], [53], polyhedral probabilistic invariant sets are approximated by using Chebyshev's inequality for linear systems with Gaussian noise. Recursive satisfaction is usually computationally intractable for general stochastic control systems.

The results of this chapter build on the above work but make significant additions and improvements. (i) All the above references focus on some specific stochastic systems (e.g., linear or one-dimensional affine nonlinear systems) or on some specific class of stochastic disturbances (e.g., Gaussian or state-independent noise). In our model, we consider general Markov controlled processes, which include general system dynamics and stochastic disturbances. (ii) Different from [52], [53], our invariant sets are defined based on trajectory inclusion as in [99] and, particularly, incorporate control inputs constrained by a compact set. An accompanying question is how to find an admissible control input when verifying or computing a PCIS. (iii) The PCISs in this chapter are different from the maximal probabilistic safe sets in [51]. Every trajectory in a PCIS is required by our definition to admit the same probability level, which does not hold for the maximal probabilistic safe set. (vi) The stochastic reachability analysis studied in [51] provides an important tool for maximizing the probability of staying in a set. Based on this, we compute a PCIS within a set with a prescribed probability level. This extends the results of [51], [99], [104].

3.1.2 Main Contributions

The objective of this chapter is to provide a novel tool for analyzing invariance in stochastic control systems. The contributions are summarized as follows.

(C3.1) We propose two novel definitions of PCIS: N-step ϵ-PCIS and infinite-horizon ϵ-PCIS (Definitions 3.1 and 3.2). An N-step ϵ-PCIS is a set within which the state can stay for N steps with probability ϵ under some admissible controller while an infinite-horizon ϵ-PCIS is a set within which the state can stay forever with probability ϵ under some admissi-

ble controller. These invariant sets are different from the ones proposed in [52], [101], which address probabilistic set invariance at each time step. Our definitions are applicable for general discrete-time stochastic control systems. We provide fundamental properties of PCISs and explore their relation to RCISs. Furthermore, we propose conditions for the existence of infinite-horizon ϵ-PCIS (Theorem 3.3).

(C3.2) We design iterative algorithms to compute the largest finite- and infinite-horizon PCIS within a given set for systems with discrete and continuous spaces. The PCIS computation is based on the stochastic backward reachable set. For discrete state and control spaces, it is shown that at each iteration, the stochastic backward reachable set computation of an N-step ϵ-PCIS can be reformulated as an LP (Theorem 3.1 and Corollary 3.1) and an infinite-horizon ϵ-PCIS as a computationally tractable MILP (Theorem 3.4). Furthermore, we prove that these algorithms terminate in a finite number of steps. For continuous state and control spaces, we present a discretization procedure. Under weaker assumptions than [105], we prove the convergence of such approximations for N-step ϵ-PCISs (Theorem 3.2). The approximations generalize the case in [51], which only discretizes the state space for a given discrete control space. Furthermore, in order to compute an infinite-horizon ϵ-PCIS, we propose an algorithm based on that an infinite-horizon PCIS always contains an RCIS.

3.2 Finite-horizon Probabilistic Controlled Invariant Sets

Recall the stochastic control system defined in Section 2.1.3. Consider a stochastic control system $\mathsf{SCS} = (\mathbb{S}, \mathbb{U}, T)$. In this section, we first define finite-horizon ϵ-PCIS for the system SCS and provide the properties of this set. Then, we explore how to compute the finite-horizon ϵ-PCIS within a given set.

Definition 3.1. *(N-step ϵ-PCIS) Consider a stochastic control system $\mathsf{SCS} = (\mathbb{S}, \mathbb{U}, T)$. Given a probability level $0 \leq \epsilon \leq 1$, a set $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ is an N-step ϵ-PCIS for SCS if for any $x \in \mathbb{Q}$, there exists at least one Markov policy $\boldsymbol{\mu} \in \mathcal{M}$ such that $p^{\boldsymbol{\mu}}_{N,\mathbb{Q}}(x) \geq \epsilon$.*

We define the stochastic backward reachable set $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q})$ by collecting all the states $x \in \mathbb{Q}$ at which the N-step invariance probability $p^*_{N,\mathbb{Q}}(x) \geq \epsilon$, i.e.,

$$\begin{aligned}\mathcal{R}^*_{\epsilon,N}(\mathbb{Q}) &= \{x \in \mathbb{Q} \mid \exists \boldsymbol{\mu} \in \mathcal{M}, p^{\boldsymbol{\mu}}_{N,\mathbb{Q}}(x) \geq \epsilon\} \\ &= \{x \in \mathbb{Q} \mid \sup_{\boldsymbol{\mu}\in\mathcal{M}} p^{\boldsymbol{\mu}}_{N,\mathbb{Q}}(x) \geq \epsilon\} \\ &= \{x \in \mathbb{Q} \mid V^*_{0,\mathbb{Q}}(x) \geq \epsilon\},\end{aligned}$$

where $V^*_{0,\mathbb{Q}}(x)$ is defined in (2.2)

If $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q}) = \mathbb{Q}$, it yields from $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ that $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q})$ is also Borel-measurable. If $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q}) \subset \mathbb{Q}$, the following lemma addresses the measurability of the set $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q})$.

Lemma 3.1. *For any $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$, the set $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q}) \subseteq \mathbb{Q}$ is universally measurable.*

Proof. Define the functions $J^*_{k,\mathbb{Q}} : \mathbb{S} \to \mathbb{R}$, $k \in \mathbb{N}_{[0,N]}$, as

$$J^*_{k,\mathbb{Q}}(x) = -V^*_{N-k,\mathbb{Q}}(x), \forall x \in \mathbb{S}.$$

As shown in [51], the function $J^*_{N,\mathbb{Q}}$ is lower-semianalytic for any $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$. From Definitions 7.20 and 7.21 in [89], we have that the function $J^*_{N,\mathbb{Q}}$ is also analytically measurable and thus is universally measurable for any $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$. According to the definition of universal measurability, the set $J^{*,-1}_{N,\mathbb{Q}}(\mathbb{B}) = \{x \in \mathbb{S} \mid J^*_{k,\mathbb{Q}}(x) \in \mathbb{B}\}$ for $\mathbb{B} \in \mathcal{B}(\mathbb{R})$ is universally measurable.

Recall the definition of the stochastic backward reachable set $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q})$, we have that

$$\begin{aligned}\mathcal{R}^*_{\epsilon,N}(\mathbb{Q}) &= \{x \in \mathbb{Q} \mid V^*_{0,\mathbb{Q}}(x) \geq \epsilon\} \\ &= \{x \in \mathbb{Q} \mid -1 \leq J^*_{N,\mathbb{Q}}(x) \leq -\epsilon\} \\ &= J^{*,-1}_{N,\mathbb{Q}}(\mathbb{Y})\end{aligned}$$

where $\mathbb{Y} = [-1, -\epsilon] \in \mathcal{B}(\mathbb{R})$. Thus, the set $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q})$ is universally measurable for any $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$. □

Let us denote by $\mathcal{P}(\mathbb{S})$ the set of all probability measures on $\mathbb{S}$. The following proposition shows that despite the universal measurability of $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q})$, for any probability measure on $\mathbb{S}$, one can find another Borel-measurable set $\tilde{\mathcal{R}}^*_{\epsilon,N}(\mathbb{Q}))$ for which the difference to $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q})$ is measure-zero.

Proposition 3.1. *For any $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ and any $p \in \mathcal{P}(\mathbb{S})$, there exists a set $\tilde{\mathcal{R}}^*_{\epsilon,N}(\mathbb{Q}) \in \mathcal{B}(\mathbb{S})$ with $\tilde{\mathcal{R}}^*_{\epsilon,N}(\mathbb{Q}) \subseteq \mathbb{Q}$ such that $p(\tilde{\mathcal{R}}^*_{\epsilon,N}(\mathbb{Q}) \triangle \mathcal{R}^*_{\epsilon,N}(\mathbb{Q})) = 0$.*

Proof. It follows from the universal measurability of $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q})$ as shown in Lemma 3.1, the Borel measurability of $\mathbb{Q}$, $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q}) \subseteq \mathbb{Q}$, and Lemma 7.26 in [89]. □

From Lemma 2.7 and the definition of $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q})$, we can verify whether a set $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ is an N-step ϵ-PCIS or not by checking if either $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q}) = \mathbb{Q}$, or $V^*_{0,\mathbb{Q}}(x) \geq \epsilon, \forall x \in \mathbb{Q}$, where $V^*_{0,\mathbb{Q}}(x)$ is defined in (2.2).

Remark 3.1. *The stochastic backward reachable set $\mathcal{R}^*_{\epsilon,N}(\mathbb{Q})$ is called the maximal probabilistic safe set in [51]. The N-step ϵ-PCIS $\mathbb{Q}$ in Definition 3.1 refines the maximal probabilistic safe set by requiring that for any initial state $x_0 \in \mathbb{Q}$, the N-step invariance probability $p^*_{\infty,\mathbb{Q}}(x_0)$ is no less than ϵ.*

In the following, we show that finite-horizon PCISs are closed under union.

Proposition 3.2. *Consider a collection of sets $\mathbb{Q}_i \in \mathcal{B}(\mathbb{S})$, $i = 1, \ldots, r$. If each $\mathbb{Q}_i$ is an N_i-step ϵ_i-PCIS for the same system* SCS*, then the union $\bigcup_{i=1}^r \mathbb{Q}_i$ is an N-step ϵ-PCIS, where $N = \min_i N_i$ and $\epsilon = \min_i \epsilon_i$.*

Proof. The result follows from the following two facts:

(i) for any $\mathbb{Q}, \mathbb{P} \in \mathcal{B}(\mathbb{S})$ with $\mathbb{Q} \subseteq \mathbb{P}$,

$$\sup_{\boldsymbol{\mu} \in \mathcal{M}} p^{\boldsymbol{\mu}}_{N,\mathbb{Q}}(x) \leq \sup_{\boldsymbol{\mu} \in \mathcal{M}} p^{\boldsymbol{\mu}}_{N,\mathbb{P}}(x), \forall N \in \mathbb{N} \text{ and } \forall x \in \mathbb{Q};$$

(ii) for any $N, N' \in \mathbb{N}$ with $N \leq N'$,

$$\sup_{\boldsymbol{\mu} \in \mathcal{M}} p^{\boldsymbol{\mu}}_{N',\mathbb{Q}}(x) \leq \sup_{\boldsymbol{\mu} \in \mathcal{M}} p^{\boldsymbol{\mu}}_{N,\mathbb{Q}}(x), \forall Q \in \mathcal{B}(\mathbb{S}) \text{ and } \forall x \in \mathbb{Q}.$$

The proof is completed. □

3.2.1 Finite-horizon ϵ-PCIS Computation

This subsection will address the following problem.

Problem 3.1. *Given a set $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ and a prescribed probability $0 \leq \epsilon \leq 1$, compute an N-step ϵ-PCIS $\tilde{\mathbb{Q}} \subseteq \mathbb{Q}$.*

To handle this problem, our basic idea is to iteratively compute stochastic backward reachable sets until convergence. A general procedure is presented in the following algorithm.

Algorithm 3.1 N-step ϵ-PCIS

1: Initialize $i = 0$ and $\mathbb{P}_i = \mathbb{Q}$.
2: Compute $V^*_{0,\mathbb{P}_i}(x), \forall x \in \mathbb{P}_i$.
3: Compute $\mathcal{R}^*_{\epsilon,N}(\mathbb{P}_i)$ and construct a Borel-measurable set $\tilde{\mathcal{R}}^*_{\epsilon,N}(\mathbb{P}_i)$ such that $p(\tilde{\mathcal{R}}^*_{\epsilon,N}(\mathbb{P}_i) \triangle \mathcal{R}^*_{\epsilon,N}(\mathbb{P}_i)) = 0$ for some $p \in \mathcal{P}(\mathbb{S})$;
4: Update $\mathbb{P}_{i+1} = \tilde{\mathcal{R}}^*_{\epsilon,N}(\mathbb{P}_i)$;
5: If $\mathbb{P}_{i+1} = \mathbb{P}_i$, stop. Else, set $i = i + 1$ and go to step 2.

In Algorithm 3.1, we first compute the stochastic backward reachable set $\mathcal{R}^*_{\epsilon,N}(\mathbb{P}_i)$ within $\mathbb{P}_i$ and then update $\mathbb{P}_{i+1}$ to be the corresponding Borel-measurable set $\tilde{\mathcal{R}}^*_{\epsilon,N}(\mathbb{P}_i)$, which is tailored by picking up a $p \in \mathcal{P}(\mathbb{S})$ such that $p(\tilde{\mathcal{R}}^*_{\epsilon,N}(\mathbb{P}_i) \triangle \mathcal{R}^*_{\epsilon,N}(\mathbb{P}_i)) = 0$ (see Proposition 3.1). The following theorem shows convergence of $\mathbb{P}_i$. The terminal condition guarantees that the resulting set by this algorithm is an N-step ϵ-PCIS $\tilde{\mathbb{Q}} \subseteq \mathbb{Q}$.

Theorem 3.1. *Let Assumption 2.1 hold. For any $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$, Algorithm 3.1 converges, i.e., $\lim_{i\to\infty} \mathbb{P}_i$ exists. If $\lim_{i\to\infty} \mathbb{P}_i \neq \emptyset$, it is the an N-step ϵ-PCIS within $\mathbb{Q}$.*

Proof. From Algorithm 3.1 and Lemma 3.1, we have that if the termination condition does not hold, $\mathbb{P}_{i+1} \subset \mathbb{P}_i$. It follows that the sequence $\{\mathbb{P}_i\}_{i\in\mathbb{N}}$ is nonincreasing. Then,

$$\liminf_{i\to\infty} \mathbb{P}_i = \bigcup_{i\geq 1} \bigcap_{j\geq i} \mathbb{P}_j = \bigcap_{j\geq 1} \mathbb{P}_j = \bigcap_{i\geq 1} \bigcup_{j\geq i} \mathbb{P}_j = \limsup_{i\to\infty} \mathbb{P}_i,$$

which suggests the existence of $\lim_{i\to\infty} \mathbb{P}_i$. Furthermore, if $\lim_{i\to\infty} \mathbb{P}_i$ is nonempty, we conclude that it is an N-step PCIS within $\mathbb{Q}$ based on the fixed-point theory. □

To facilitate the practical implementation of Algorithm 3.1, we need to address two important properties: the computational tractability of $V^*_{0,\mathbb{P}_i}(x)$, $\forall x \in \mathbb{P}_i$, and the finite-step convergence of Algorithm 3.1. In the following,

we will derive these two properties for discrete and continuous spaces, respectively. It is shown that if the spaces are discrete, the properties are guaranteed and in particular at each iteration we only need to solve an LP to compute the exact value of $V^*_{0,\mathbb{P}_i}$. If the spaces are continuous, we will design a discretization algorithm with convergence guarantee, which enables us to preserve the above two properties.

Discrete state and control spaces

If the state and control spaces are discrete, i.e., they are finite sets, the stochastic kernel $T(y|x,u)$ denotes the transition probability from state $x \in \mathbb{S}$ to state $y \in \mathbb{S}$ under control action $u \in \mathbb{U}_x$, which satisfies that $\sum_{y\in\mathbb{S}} T(y|x,u) = 1$, $\forall x \in \mathbb{S}$ and $u \in \mathbb{U}_x$.

In this case, according to Theorem 1 of [106], we can exactly compute $V^*_{0,\mathbb{P}_i}(x)$ via an LP. Moreover, the existence of the optimal Markov policy can be always guaranteed.

Lemma 3.2. *Given any set $\mathbb{P}_i \subset \mathbb{S}$, the value functions $V^*_{k,\mathbb{P}_i}$ in (2.2) can be obtained by solving an LP:*

$$\min \sum_{k=0}^{N} \sum_{x\in\mathbb{P}_i} v_k(x) \tag{3.1a}$$

$$\text{s.t. } \forall x \in \mathbb{P}_i$$

$$v_k(x) \geq \sum_{y\in\mathbb{P}_i} v_{k+1}(y)T(y|x,u), \forall u \in \mathbb{U}_x, \forall k \in \mathbb{N}_{[0,N-1]}, \tag{3.1b}$$

$$v_N(x) \geq 1, \tag{3.1c}$$

which gives $V^*_{k,\mathbb{P}_i}(x) = v^*_k(x), \forall x \in \mathbb{P}_i$ and $\forall k \in \mathbb{N}_{[0,N]}$, where v^*_k is the optimal solution of (3.1). The optimal Markov policy $\boldsymbol{\mu}^*_{\mathbb{P}_i} = (\mu^*_{0,\mathbb{P}_i}, \mu^*_{1,\mathbb{P}_i}, \ldots, \mu^*_{N-1,\mathbb{P}_i})$ is given by $\mu^*_{k,\mathbb{P}_i}(x) = u$ where $u \in \mathbb{U}_x$ is such that

$$v^*_k(x) = \sum_{y\in\mathbb{P}_i} v^*_{k+1}(y)T(y|x,u). \tag{3.2}$$

Proof. See Theorem 1 in [106] for the proof. □

Corollary 3.1. *For discrete state and control spaces, Algorithm 3.1 converges in a finite number of iterations. Furthermore, at each iteration, the N-step invariance probability $V^*_{0,\mathbb{P}_i}(x)$, $\forall x \in \mathbb{P}_i$, can be computed via the LP (3.1) and the corresponding optimal policy is determined by (3.2).*

Proof. The finite-step convergence of Algorithm 3.1 follows from Theorem 3.1 and the finite cardinality of $\mathbb{Q}$. The remaining part directly follows from Lemma 3.2. □

Remark 3.2. *When implementing Algorithm 3.1 to a system with discrete spaces, the maximal number of iterations is $|\mathbb{Q}|$. The resulting set is the largest PCIS within the give set. At each iteration, an LP is solved to compute the value of $V^*_{0,\mathbb{P}_i}(x)$, $\forall x \in \mathbb{P}_i$. The number of the decision values in the LP is at most $|\mathbb{Q}|(N+1)$ and the number of constraints is at most $|\mathbb{Q}|(N|\mathbb{U}|+1)$. It follows from [107] that Algorithm 3.1 can be implemented in $O(|\mathbb{Q}|^2(N|\mathbb{U}|+1))$ time.*

Continuous state and control action spaces

In order to preserve the computational tractability of $V^*_{0,\mathbb{P}_i}$ and the finite-step convergence of Algorithm 3.1, if the state and control spaces are both continuous, we first discretize the spaces with convergence guarantee. Then, we adapt Algorithm 3.1 to compute an approximate N-step ϵ-PCIS within a given set.

Assume that $\mathbb{S} \subseteq \mathbb{R}^{n_x}$ and $\mathbb{U} \subseteq \mathbb{R}^{n_u}$ for some $n_x, n_u \in \mathbb{N}$. For simplicity, we use Euclidean metric for the spaces $\mathbb{S}$ and $\mathbb{U}$. For any $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$, we define $\phi(\mathbb{Q}) = Leb(\mathbb{Q})$ where $Leb(\cdot)$ denotes the Lebesgue measure of sets. We suppose that the stochastic kernel $T(\cdot|x,u)$ admits a density $t(y|x,u)$, which represents the probability density of y given the current state x and the control action u.

Now we consider Problem 3.1, where we assume that the given set $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ is compact, which implies that $\phi(\mathbb{Q})$ is bounded. We further suppose that the density function satisfies the following assumption.

Assumption 3.1. *There exists a constant L such that for any $x, x', y, y' \in \mathbb{Q}$, and $u, u' \in \mathbb{U}$,*

$$|t(y|x,u) - t(y'|x',u')| \leq L(\|y-y'\| + \|x-x'\| + \|u-u'\|).$$

Discretization We discretize the compact set $\mathbb{Q} \subset \mathbb{S}$ into m_x pair-wise disjoint nonempty Borel sets $\mathbb{Q}_i$, $i \in \mathbb{N}_{[1,m_x]}$, i.e., $\mathbb{Q} = \cup_{i=1}^{m_x} \mathbb{Q}_i$. We pick a representative state from each set $\mathbb{Q}_i$, denoted by q_i. Let $\hat{\mathbb{Q}} = \{q_i, i \in \mathbb{N}_{[1,m_x]}\}$, $d_i = \sup_{x,y\in\mathbb{Q}_i} \|x-y\|$, and $D_x = \max_{i\in\mathbb{N}_{[1,m_x]}} d_i$.

Similarly, the compact control space $\mathbb{U}$ is divided into m_u pair-wise disjoint nonempty Borel sets $\mathbb{C}_i$, $i \in \mathbb{N}_{[1,m_u]}$, i.e., $\mathbb{U} = \cup_{i=1}^{m_u} \mathbb{C}_i$. We pick a representative element from the set $\mathbb{C}_i$, denoted by $\hat{u}_i$. Let $\hat{\mathbb{U}} = \{\hat{u}_i, i \in \mathbb{N}_{[1,m_u]}\}$, $l_i = \sup_{x,y\in\mathbb{C}_i} \|x - y\|$, and $D_u = \max_{i\in\mathbb{N}_{[1,m_u]}} l_i$.

Let the grid size be a constant $\delta \geq \max\{D_x, D_u\}$. For each $x \in \mathbb{Q}$, define the set of admissible discrete control actions as

$$\hat{\mathbb{U}}_x = \{\hat{u} \in \hat{\mathbb{U}} \mid \|u - \hat{u}\| \leq \delta \text{ for some } u \in \mathbb{U}_{s_x}\}, \tag{3.3}$$

where s_x is the representative state of $\mathbb{Q}_i$ to which x belongs, i.e., $s_x = q_i$ if $x \in \mathbb{Q}_i$. Following [105], the following lemma shows that each $x \in \mathbb{Q}$ has a nonempty admissible discretized control set.

Lemma 3.3. *For each $q_i \in \hat{\mathbb{Q}}$, the set $\hat{\mathbb{U}}_{q_i}$ is nonempty and $\hat{\mathbb{U}}_x = \hat{\mathbb{U}}_{q_i}, \forall x \in \mathbb{Q}_i$.*

Proof. Since the admissible control set $\mathbb{U}_{s_x}$ is nonempty, $\forall x \in \mathbb{Q}$, there exists $\hat{u} \in \hat{\mathbb{U}}$ such that $\|u - \hat{u}\| \leq \delta$, $\forall u \in \mathbb{U}_{s_x}$. Hence, by the definition of s_x, we have that the set $\hat{\mathbb{U}}_{q_i}$ is nonempty for each $q_i \in \hat{\mathbb{Q}}$. Furthermore, from (3.3), it is easy to obtain that $\hat{\mathbb{U}}_x = \hat{\mathbb{U}}_{q_i}, \forall x \in \mathbb{Q}_i$. □

As in [105], let us define the function $\hat{t} : \mathbb{Q} \times \mathbb{Q} \times \hat{\mathbb{U}} \to \mathbb{R}$

$$\hat{t}(y|x,\hat{u}) = \begin{cases} \frac{t(s_y|s_x,\hat{u})}{\int_{\mathbb{Q}} t(s_z|s_x,\hat{u})dz}, & \text{if } \int_{\mathbb{Q}} t(s_z|s_x,\hat{u})dz \geq 1, \\ t(s_y|s_x,\hat{u}), & \text{otherwise.} \end{cases} \tag{3.4}$$

From (3.4), we observe that all states $y \in \mathbb{Q}_i$ enjoy the same stochastic kernel. An approximate stochastic control system is given by a triple $\widehat{\mathsf{SCS}}_{\mathbb{Q}} = (\hat{\mathbb{Q}}, \hat{\mathbb{U}}, \hat{T})$. Here the transition probability $\hat{T}(q_j|q_i,\hat{u})$ is defined by

$$\hat{T}(q_j|q_i,\hat{u}) = \int_{\mathbb{Q}_j} \hat{t}(y|q_i,\hat{u})dy,$$

where $q_i, q_j \in \hat{\mathbb{Q}}$ with $q_i \in \mathbb{Q}_i$ and $q_j \in \mathbb{Q}_j$, and $\hat{u} \in \hat{\mathbb{U}}$.

Approximation of PCISs For the approximate system $\widehat{\mathsf{SCS}}_{\mathbb{Q}}$, the discretized version of the (dynamic programming) DP (2.2) is given by

$$\begin{cases} \hat{V}^*_{N,\mathbb{Q}}(q_i) = 1, \\ \hat{V}^*_{k,\mathbb{Q}}(q_i) = \max\limits_{\hat{u}\in\hat{\mathbb{U}}} \big(\sum\limits_{j=1}^{m_x} \hat{V}^*_{k+1,\mathbb{Q}}(q_j)\hat{T}(q_j|q_i,\hat{u})\big), \forall k \in \mathbb{N}_{[0,N-1]}. \end{cases}$$

For each $x \in \mathbb{Q}_i$, $\hat{V}^*_{k,\mathbb{Q}}(x) = \hat{V}^*_{k,\mathbb{Q}}(q_i), \forall k \in \mathbb{N}_{[0,N]}$. We define the discretized optimal Markov policy $\hat{\boldsymbol{\mu}}^*_{\mathbb{Q}} = (\hat{\mu}^*_{0,\mathbb{Q}}, \ldots, \hat{\mu}^*_{N-1,\mathbb{Q}})$ as

$$\begin{aligned}\hat{\mu}^*_{k,\mathbb{Q}}(q_i) &= \arg\max_{\hat{u}\in\hat{\mathbb{U}}} \int_{\mathbb{Q}} \hat{V}^*_{k+1,\mathbb{Q}}(y)\hat{t}(y|q_i,\hat{u})dy, \\ &= \arg\max_{\hat{u}\in\hat{\mathbb{U}}} \big(\sum_{j=1}^{m_x} \hat{V}^*_{k+1,\mathbb{Q}}(q_j)\hat{T}(q_j|q_i,\hat{u})\big).\end{aligned}$$

For each $x \in \mathbb{Q}_i$, $\hat{\mu}^*_{k,\mathbb{Q}}(x) = \hat{\mu}^*_{k,\mathbb{Q}}(q_i),\ \forall k \in \mathbb{N}_{[0,N-1]}$.

Remark 3.3. *Since the state and control action spaces of the approximated system $\widehat{\mathsf{SCS}}_{\mathbb{Q}}$ are finite, the value of $\hat{V}^*_{k,\mathbb{Q}}$ can be computed via the LP (3.1) and the corresponding optimal policy can be determined by (3.2). In addition, all the states in each $\mathbb{Q}_i$ share the same approximate N-step invariance probability and optimal policy as the representative state $q_i \in \mathbb{Q}_i$.*

Before showing the error bound on the value function in Lemma 3.6, we need two auxiliary lemmas. Lemma 3.4 shows that the value functions in (2.2) are Lipschitz continuous. It is adapted from Theorem 8 in [51]. Lemma 3.5 shows that the difference between the approximate density function and the original density function is bounded.

Lemma 3.4. *Under Assumptions 2.1 and 3.1, for any $x, x' \in \mathbb{Q}$, the value functions $V^*_{k,\mathbb{Q}}$ in (2.2) satisfy*

$$|V^*_{k,\mathbb{Q}}(x) - V^*_{k,\mathbb{Q}}(x')| \leq \phi(\mathbb{Q})L\|x - x'\|, \forall k \in \mathbb{N}_{[0,N]}. \tag{3.5}$$

Proof. Since $V^*_{N,\mathbb{Q}}(x) = 1$ for all $x \in \mathbb{Q}$, the inequality (3.5) holds for $k = N$. When $k \in \mathbb{N}_{[0,N-1]}$, for any $x, x' \in \mathbb{Q}$, we have

$$\begin{aligned}&|V^*_{k,\mathbb{Q}}(x) - V^*_{k,\mathbb{Q}}(x')| \\ &= |\sup_{u\in\mathbb{U}} \int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)t(y|x,u)dy - \sup_{u\in\mathbb{U}} \int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)t(y|x',u)dy| \\ &\leq \sup_{u\in\mathbb{U}} |\int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)(t(y|x,u) - t(y|x',u))|dy \\ &\leq \sup_{u\in\mathbb{U}} \int_{\mathbb{Q}} |(t(y|x,u) - t(y|x',u))|dy \\ &\leq \phi(\mathbb{Q})L(\|x - x'\|),\end{aligned}$$

which completes the proof. □

Lemma 3.5. *Under Assumptions 3.1, for all $y \in \mathbb{Q}$ and $q_i \in \hat{\mathbb{Q}}$,*

$$\int_{\mathbb{Q}} |\hat{t}(y|q_i, \hat{u}) - t(y|q_i, \hat{u})|dy \leq 2\phi(\mathbb{Q})L\delta, \forall \hat{u} \in \hat{\mathbb{U}}.$$

Proof. If $\int_{\mathbb{Q}} t(s_z|s_x, \hat{u})dz < 1$, it follows from Assumption 3.1 that

$$\int_{\mathbb{Q}} |\hat{t}(y|q_i, \hat{u}) - t(y|q_i, \hat{u})|dy \leq \phi(\mathbb{Q})L\delta.$$

And if $\int_{\mathbb{Q}} t(s_z|s_x, \hat{u})dz \geq 1$, we first have

$$\begin{aligned}
0 &\leq \int_{\mathbb{Q}} t(s_y|q_i, \hat{u})dy - 1 \\
&\leq \int_{\mathbb{Q}} t(s_y|q_i, \hat{u})dy - \int_{\mathbb{Q}} t(y|q_i, \hat{u})dy \\
&\leq \int_{\mathbb{Q}} |t(s_y|q_i, \hat{u}) - t(y|q_i, \hat{u})|dy \\
&\leq \phi(\mathbb{Q})L\delta.
\end{aligned}$$

Furthermore, we have

$$\begin{aligned}
&\int_{\mathbb{Q}} |\hat{t}(y|q_i, \hat{u}) - t(y|q_i, \hat{u})|dy \\
=& \int_{\mathbb{Q}} \frac{|t(s_y|q_i, \hat{u}) - t(y|q_i, \hat{u}) \int_{\mathbb{Q}} t(s_z|s_x, \hat{u})dz|}{\int_{\mathbb{Q}} t(s_z|s_x, \hat{u})dz} dy \\
\leq& \int_{\mathbb{Q}} |t(s_y|q_i, \hat{u}) - t(y|q_i, \hat{u}) \int_{\mathbb{Q}} t(s_z|s_x, \hat{u})dz|dy \\
\leq& \int_{\mathbb{Q}} |t(s_y|q_i, \hat{u}) - t(y|q_i, \hat{u})|dy \\
&+|\int_{\mathbb{Q}} t(s_z|s_x, \hat{u})dz - 1| \int_{\mathbb{Q}} |t(y|q_i, \hat{u})|dy \\
\leq& 2\phi(\mathbb{Q})L\delta.
\end{aligned}$$

This completes the proof. □

Lemma 3.6. *Under Assumptions 2.1 and 3.1, the functions* $V^*_{k,\mathbb{Q}}(x)$ *and* $\hat{V}^*_{k,\mathbb{Q}}(x)$ *satisfy that* $\forall x \in \mathbb{Q}$,

$$|V^*_{k,\mathbb{Q}}(x) - \hat{V}^*_{k,\mathbb{Q}}(x)| \leq \tau_k(\mathbb{Q})\delta, \tag{3.6}$$

where

$$\begin{cases} \tau_N(\mathbb{Q}) = 0, \\ \tau_k(\mathbb{Q}) = 4\phi(\mathbb{Q})L + \tau_{k+1}(\mathbb{Q}),\ \forall k \in \mathbb{N}_{[0,N-1]}. \end{cases} \tag{3.7}$$

Proof. It is easy to check it for $k = N$ since $V^*_{N,\mathbb{Q}}(x) = \hat{V}^*_{k,\mathbb{Q}}(x) = 1, \forall x \in \mathbb{Q}$. By induction, we assume that $|V^*_{k+1,\mathbb{Q}}(x) - \hat{V}^*_{k+1,\mathbb{Q}}(x)| \leq \tau_{k+1}(\mathbb{Q})\delta$, $x \in \mathbb{Q}$. For any $q_i \in \mathbb{Q}_i$, $i \in \mathbb{N}_{[1,m_x]}$, we define

$$\mu^*_k = \arg\sup_{u\in\mathbb{U}} \int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)t(y|q_i,u)dy$$

and

$$\hat{\mu}^*_k = \arg\max_{\hat{u}\in\hat{\mathbb{U}}} \int_{\mathbb{Q}} \hat{V}^*_{k+1,\mathbb{Q}}(y)\hat{t}(y|q_i,\hat{u})dy.$$

According to the dicretization procedure of the control space, we can choose some $\hat{\nu}_k \in \hat{\mathbb{U}}$ such that $\|\mu^*_k - \hat{\nu}_k\| \leq \delta$. Then, we have that

$$\begin{aligned}
&V^*_{k,\mathbb{Q}}(q_i) - \hat{V}^*_{k,\mathbb{Q}}(q_i) \\
=& \int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)t(y|q_i,\mu^*_k)dy - \int_{\mathbb{Q}} \hat{V}^*_{k+1,\mathbb{Q}}(y)\hat{t}(y|q_i,\hat{\mu}^*_k)dy \\
\leq& \int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)t(y|q_i,\mu^*_k)dy - \int_{\mathbb{Q}} \hat{V}^*_{k+1,\mathbb{Q}}(y)\hat{t}(y|q_i,\hat{\nu}_k)dy \\
\leq& |\int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)t(y|q_i,\mu^*_k)dy - \int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)t(y|q_i,\hat{\nu}_k)dy| \\
&+|\int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)t(y|q_i,\hat{\nu}_k)dy - \int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)\hat{t}(y|q_i,\hat{\nu}_k)dy| \\
&+|\int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)\hat{t}(y|q_i,\hat{\nu}_k)dy - \int_{\mathbb{Q}} \hat{V}^*_{k+1,\mathbb{Q}}(y)\hat{t}(y|q_i,\hat{\nu}_k)dy| \\
\leq& \phi(\mathbb{Q})L\delta + 2\phi(\mathbb{Q})L\delta + \tau_{k+1}(\mathbb{Q})\delta \\
=& (3\phi(\mathbb{Q})L + \tau_{k+1}(\mathbb{Q}))\delta,
\end{aligned}$$

and

$$\begin{aligned}
&\hat{V}^*_{k,\mathbb{Q}}(q_i) - V^*_{k,\mathbb{Q}}(q_i) \\
\leq &\int_{\mathbb{Q}} \hat{V}^*_{k+1,\mathbb{Q}}(y)\hat{t}(y|q_i,\hat{\mu}^*_k)dy - \int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)t(y|q_i,\hat{\mu}^*_k)dy \\
\leq &|\int_{\mathbb{Q}} \hat{V}^*_{k+1,\mathbb{Q}}(y)\hat{t}(y|q_i,\hat{\mu}^*_k)dy - \int_{\mathbb{Q}} \hat{V}^*_{k+1,\mathbb{Q}}(y)t(y|q_i,\hat{\mu}^*_k)dy| \\
&+|\int_{\mathbb{Q}} \hat{V}^*_{k+1,\mathbb{Q}}(y)t(y|q_i,\hat{\mu}^*_k)dy - \int_{\mathbb{Q}} V^*_{k+1,\mathbb{Q}}(y)t(y|q_i,\hat{\mu}^*_k)dy| \\
\leq &(2\phi(\mathbb{Q})L + \tau_{k+1}(\mathbb{Q}))\delta.
\end{aligned}$$

Thus, we have

$$|V^*_{k,\mathbb{Q}}(q_i) - \hat{V}^*_{k,\mathbb{Q}}(q_i)| \leq (3\phi(\mathbb{Q})L + \tau_{k+1}(\mathbb{Q}))\delta.$$

For any $x \in \mathbb{Q}_i$, $i \in \mathbb{N}_{[1,m_x]}$, it follows that

$$\begin{aligned}
|V^*_{k,\mathbb{Q}}(x) - \hat{V}^*_{k,\mathbb{Q}}(x)| &= |V^*_{k,\mathbb{Q}}(x) - \hat{V}^*_{k,\mathbb{Q}}(q_i)| \\
&\leq |V^*_{k,\mathbb{Q}}(x) - V^*_{k,\mathbb{Q}}(q_i)| + |V^*_{k,\mathbb{Q}}(q_i) - \hat{V}^*_{k,\mathbb{Q}}(q_i)| \\
&\leq (4\phi(\mathbb{Q})L + \tau_{k+1}(\mathbb{Q}))\delta = \tau_k(\mathbb{Q})\delta,
\end{aligned}$$

which completes the proof of the inequality (3.6). □

Remark 3.4. *Lemma 3.6 guarantees convergence as the grid size tends to zero and generalizes the case considered in [51], which only discretizes the state space for a given finite control space. To prove Lemma 3.6, we need to show that (i) the value functions in (2.2) are Lipschitz continuous (Lemma 3.4), which is similar to Theorem 8 in [51], and (ii) the difference between the approximate density function and the original density function is bounded (Lemma 3.5), which is different from that in [51].*

Theorem 3.2. *Let Assumptions 2.1 and 3.1 hold. Consider a compact set $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ and a corresponding discretized set $\hat{\mathbb{Q}}$ of $\mathbb{Q}$. If $\hat{\mathbb{Q}}$ is an N-step $\hat{\epsilon}$-PCIS for the approximate system $\widehat{\mathsf{SCS}}_{\mathbb{Q}} = (\hat{\mathbb{Q}}, \hat{\mathbb{U}}, \hat{T})$, and $\hat{\epsilon} \geq \tau_0(\mathbb{Q})\delta$, the set $\mathbb{Q}$ is an N-step ϵ-PCIS for the system SCS, where $\epsilon = \hat{\epsilon} - \tau_0(\mathbb{Q})\delta$.*

Proof. According to the construction of the discretized system $\widehat{\mathsf{SCS}}_{\mathbb{Q}}$, we have that $\forall k \in \mathbb{N}_{[0,N]}$, $\forall i \in \mathbb{N}_{[1,m_x]}$ and $\forall x \in \mathbb{Q}_i$, $\hat{V}^*_{k,\mathbb{Q}}(x) = \hat{V}^*_{k,\mathbb{Q}}(q_i)$. Since $\hat{\mathbb{Q}}$

is an N-step $\hat{\epsilon}$-PCIS, it follows that $\forall x \in \mathbb{Q}$, $\hat{V}^*_{0,\mathbb{Q}}(x) \geq \hat{\epsilon}$. By Lemma 3.6 and triangle inequality, we have

$$V^*_{0,\mathbb{Q}}(x) \geq \hat{V}^*_{0,\mathbb{Q}}(x) - \tau_0(\mathbb{Q})\delta \geq \hat{\epsilon} - \tau_0(\mathbb{Q})\delta, \forall x \in \mathbb{Q}.$$

Then, when $\hat{\epsilon} \geq \tau_0(\mathbb{Q})\delta$, we conclude that the set $\mathbb{Q}$ is an N-step ϵ-PCIS where $0 \leq \epsilon = \hat{\epsilon} - \tau_0(\mathbb{Q})\delta$. □

Remark 3.5. *From Theorem 3.2, if $0 \leq \epsilon < 1$, by choosing a suitable grid size $0 < \delta \leq \frac{1-\epsilon}{\tau_0(\mathbb{Q})}$, the problem of computing an N-step ϵ-PCIS within $\mathbb{Q}$ for SCS can be transformed into that of computing an approximate N-step $\hat{\epsilon}$-PCIS with probability $\hat{\epsilon} \geq \epsilon + \tau_0(\mathbb{Q})\delta$ for $\widehat{\mathsf{SCS}}_{\mathbb{Q}}$.*

Computation algorithm Assume that a probability level $0 \leq \epsilon < 1$ is given. After discretizing the set $\mathbb{Q}$ and the control space $\mathbb{U}$, we modify Algorithm 3.1 to compute an N-step ϵ-PCIS $\tilde{\mathbb{Q}} \subseteq \mathbb{Q}$, as shown in the following.

Algorithm 3.2 Approximate N-step ϵ-PCIS

1: Choose grid size $0 < \delta < \frac{1-\epsilon}{\tau_0(\mathbb{Q})}$, discretize the sets $\mathbb{Q}$ and $\mathbb{U}$, construct an approximate system $\widehat{\mathsf{SCS}}_{\mathbb{Q}} = (\hat{\mathbb{Q}}, \hat{\mathbb{U}}, \hat{T})$.
2: Initialize $i = 0$, $\mathbb{P}_i = \mathbb{Q}$, and $\hat{\mathbb{P}}_i = \hat{\mathbb{Q}}$.
3: Compute $\hat{V}^*_{0,\mathbb{P}_i}(q_j), \forall q_j \in \hat{\mathbb{P}}_i$.
4: Compute $\tau_0(\mathbb{P}_i)$ by (3.7) and $\hat{\epsilon} = \epsilon + \tau_0(\mathbb{P}_i)\delta$.
5: Compute the set $\hat{\mathbb{P}}_{i+1} = \mathcal{R}^*_{\hat{\epsilon},N}(\hat{\mathbb{P}}_i)$ for $\widehat{\mathsf{SCS}}_{\mathbb{Q}}$ and $\mathbb{P}_i = \cup_{q_j \in \hat{\mathbb{P}}_i} \mathbb{Q}_j$
6: If $\hat{\mathbb{P}}_{i+1} = \hat{\mathbb{P}}_i$, stop. Else, set $i = i + 1$ and go to step 3.

In Algorithm 3.2, we first construct an approximate system $\widehat{\mathsf{SCS}}_{\mathbb{Q}} = (\hat{\mathbb{Q}}, \hat{\mathbb{U}}, \hat{T})$ with grid size $0 < \delta < \frac{1-\epsilon}{\tau_0(\mathbb{Q})}$. Then, following similar steps as in Algorithm 3.1, we compute the stochastic backward reachable set iteratively for the system $\widehat{\mathsf{SCS}}_{\mathbb{Q}}$. At each iteration, an LP is solved to obtain the N-step invariance probability. One difference is that the stochastic backward reachable set is computed with respect to $\hat{\epsilon} = \epsilon + \tau_0(\mathbb{P}_i)\delta$ and the updated set for the system SCS is the union of the subsets of $\mathbb{Q}$ corresponding to the stochastic backward reachable set. By Theorem 3.2, the resulting set by Algorithm 3.2 is an N-step ϵ-PCIS.

Corollary 3.2. *Let Assumptions 2.1 and 3.1 hold. For continuous state and control spaces, Algorithm 3.2 converges in a finite number of iterations and generates an N-step ϵ-PCIS. Furthermore, at each iteration, the N-step invariance probability $\hat{V}^*_{0,\mathbb{P}_i}(q_j), \forall q_j \in \hat{\mathbb{P}}_i$, can be computed via the LP (3.1) and the corresponding optimal policy is determined by (3.2).*

Proof. By Theorem 3.2 and the Borel measurability of the subsets $\mathbb{Q}_i, \forall i \in \mathbb{N}_{[1,m_x]}$, it follows that the set generated by Algorithm 3.2 is an N-step ϵ-PCIS. The remaining part is similar to the proof of Corollary 3.1. □

Remark 3.6. *When implementing Algorithm 3.2 to a system with continuous spaces, it follows from [107] that Algorithm 3.2 can be implemented in $O(m_x^2(Nm_u+1))$ time, cf. Remark 3.2.*

3.3 Infinite-horizon Probabilistic Controlled Invariant Sets

Now let us extend finite-horizon ϵ-PCISs to infinite-horizon ϵ-PCISs. In this section, we define the infinite-horizon ϵ-PCIS and explore the conditions of its existence. Furthermore, we provide algorithms to compute an infinite-horizon ϵ-PCIS within a given set.

Definition 3.2. *(Infinite-horizon PCIS) Consider a stochastic control system* $\mathsf{SCS} = (\mathbb{S}, \mathbb{U}, T)$. *Given a confidence level $0 \leq \epsilon \leq 1$, a set $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ is an infinite-horizon ϵ-PCIS for* SCS *if for any $x \in \mathbb{Q}$, there exists at least one stationary policy $\boldsymbol{\mu} \in \mathcal{M}$ such that $p^{\boldsymbol{\mu}}_{\infty,\mathbb{Q}}(x) \geq \epsilon$.*

We define the stochastic backward reachable set $\mathcal{R}^*_{\epsilon,\infty}(\mathbb{Q})$ by collecting all the states $x \in \mathbb{Q}$ at which the infinite-horizon invariance probability $p^*_{\infty,\mathbb{Q}}(x) \geq \epsilon$, i.e.,

$$\begin{aligned}\mathcal{R}^*_{\epsilon,\infty}(\mathbb{Q}) &= \{x \in \mathbb{Q} \mid \exists \boldsymbol{\mu} \in \mathcal{M}, p^{\boldsymbol{\mu}}_{\infty,\mathbb{Q}}(x) \geq \epsilon\} \\ &= \{x \in \mathbb{Q} \mid \sup_{\boldsymbol{\mu}\in\mathcal{M}} p^{\boldsymbol{\mu}}_{\infty,\mathbb{Q}}(x) \geq \epsilon\} \\ &= \{x \in \mathbb{Q} \mid G^*_{\infty,\mathbb{Q}}(x) \geq \epsilon\},\end{aligned}$$

where $G^*_{\infty,\mathbb{Q}}(x)$ is defined by (2.3)–(2.4).

For the infinite-horizon case, Lemma 3.1 and Proposition 3.1 still hold. That is, the set $\mathcal{R}^*_{\epsilon,\infty}(\mathbb{Q})$ is universally measurable and for any $p \in \mathcal{P}(\mathbb{S})$, there

exists another Borel-measurable set $\tilde{\mathcal{R}}^*_{\epsilon,\infty}(\mathbb{Q}) \subseteq \mathbb{Q}$ such that $p(\tilde{\mathcal{R}}^*_{\epsilon,\infty}(\mathbb{Q}) \triangle \mathcal{R}^*_{\epsilon,\infty}(\mathbb{Q})) = 0$.

Under Assumption 2.2, by Lemma 2.8 and the definition of $\mathcal{R}^*_{\epsilon,\infty}(\mathbb{Q})$, we can verify whether a set $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ is an infinite-horizon ϵ-PCIS or not by checking if either $\mathcal{R}^*_{\epsilon,\infty}(\mathbb{Q}) = \mathbb{Q}$, or $G^*_{\infty,\mathbb{Q}}(x) \geq \epsilon, \forall x \in \mathbb{Q}$, where $G^*_{\infty,\mathbb{Q}}(x)$ is defined by (2.3)–(2.4).

Let us adapt the RCIS in Definition 2.21 to a stochastic control system $\mathsf{SCS} = (\mathbb{S}, \mathbb{U}, T)$. We say an RCIS $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ for SCS is an N-step ϵ-PCIS with $N = 1$ and $\epsilon = 1$.

Remark 3.7. *Another interpretation of RCIS is that a set $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ is an RCIS if for any $x \in \mathbb{Q}$, there exists at least one control input $u \in \mathbb{U}$ such that $T(\mathbb{Q}|x, u) = 1$. It is easy to verify that an RCIS is also an infinite-horizon ϵ-PCIS with $\epsilon = 1$. It is called an absorbing set in [108] where there is no control input. In the following, we show that the RCIS plays an important role in the existence of infinite-horizon PCIS and provide how to design an algorithm to compute such PCIS based on RCIS.*

Remark 3.8. *Note that infinite-horizon ϵ-PCISs are also closed under union, as shown in Proposition 3.2 when N is replaced by ∞.*

3.3.1 Existence of Infinite-horizon PCIS

Intuitively, the monotone decrease of the value functions $G^*_{k,\mathbb{Q}}(x)$ defined in (2.3) may imply that the value of $G^*_{\infty,\mathbb{Q}}(x)$ is one or zero. However, it is possible to get $0 < G^*_{\infty,\mathbb{Q}}(x) < 1$ in some cases (see Examples 1 and 2 in Section 3.4). The following theorem provides necessary conditions and sufficient conditions for the existence of an infinite-horizon ϵ-PCIS with $\epsilon > 0$.

Theorem 3.3. *Suppose that Assumption 2.2 holds and let $0 < \epsilon \leq 1$ be fixed. Given a nonempty set $\mathbb{Q}$, let u_x be the control input such that (2.4) holds for each $x \in \mathbb{Q}$. The set $\mathbb{Q}$ is an infinite-horizon ϵ-PCIS*

(i) only if *there exists an RCIS $\mathbb{Q}_f \subseteq \mathbb{Q}$ such that $\forall x \in \mathbb{Q} \setminus \mathbb{Q}_f$,*

$$T(\mathbb{Q}_f|x, u_x) + \int_{\mathbb{Q}\setminus\mathbb{Q}_f} T(\mathbb{Q}_f|y, u_y) T(dy|x, u_x) + \frac{\rho^2}{1-\rho} \geq \epsilon, \quad (3.8)$$

where $\rho = \sup_{x\in\mathbb{Q}\setminus\mathbb{Q}_f} \int_{\mathbb{Q}\setminus\mathbb{Q}_f} T(dy|x, u_x)$;

(ii) if *there exists an RCIS* $\mathbb{Q}_f \subseteq \mathbb{Q}$ *such that* $\forall x \in \mathbb{Q} \setminus \mathbb{Q}_f$,

$$T(\mathbb{Q}_f|x,u_x) + \int_{\mathbb{Q}\setminus\mathbb{Q}_f} T(\mathbb{Q}_f|y,u_y)T(dy|x,u_x) \geq \epsilon. \tag{3.9}$$

Proof. Only-if-part: Under Assumption 2.2, the fact that the set $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ is an infinite-horizon ϵ-PCIS is equivalent to $G^*_{\infty,\mathbb{Q}}(x) \geq \epsilon, \forall x \in \mathbb{Q}$. Let $\theta = \sup_{x\in\mathbb{Q}} G^*_{\infty,\mathbb{Q}}(x)$. Under Assumption 2.2, $G^*_{\infty,\mathbb{Q}}(x)$ exists for all $x \in \mathbb{Q}$. The set $\tilde{\mathbb{Q}}_f = \{x \in \mathbb{Q} \mid G^*_{\infty,\mathbb{Q}}(x) = \theta\}$ collects all the states for which the value of $G^*_{\infty,\mathbb{Q}}$ is maximal over the set $\mathbb{Q}$. Extending Lemma 3.1 to infinite-horizon case, we have that the set $\tilde{\mathbb{Q}}_f$ is universally measurable. By Lemma 7.16 in [89], we have that for any $p \in \mathcal{P}(\mathbb{S})$, there exists a Borel-measurable set $\mathbb{Q}_f \subseteq \mathbb{Q}$ such that $p(\mathbb{Q}_f \triangle \tilde{\mathbb{Q}}_f) = 0$. Let us specify $\mathbb{Q}_f$ by choosing a p.

Next we will show that the set $\mathbb{Q}_f$ is an RCIS. It follows from Assumption 2.2 and Lemma 2.8 that $\forall x \in \mathbb{Q}_f$,

$$\begin{aligned}
G^*_{\infty,\mathbb{Q}}(x) &= \int_{\mathbb{Q}_f} G^*_{\infty,\mathbb{Q}}(y)T(dy|x,u_x) + \int_{\mathbb{Q}\setminus\mathbb{Q}_f} G^*_{\infty,\mathbb{Q}}(y)T(dy|x,u_x) \\
&= G^*_{\infty,\mathbb{Q}}(x)\int_{\mathbb{Q}_f} T(dy|x,u_x) + \int_{\mathbb{Q}\setminus\mathbb{Q}_f} G^*_{\infty,\mathbb{Q}}(y)T(dy|x,u_x) \quad (3.10)\\
&\leq G^*_{\infty,\mathbb{Q}}(x)T(\mathbb{Q}_f|x,u_x) + G^*_{\infty,\mathbb{Q}}(x)T(\mathbb{Q}\setminus\mathbb{Q}_f|x,u_x) \quad (3.11)\\
&= G^*_{\infty,\mathbb{Q}}(x)(T(\mathbb{Q}_f|x,u_x) + T(\mathbb{Q}\setminus\mathbb{Q}_f|x,u_x)),
\end{aligned}$$

where Eq. (3.10) follows from $G^*_{\infty,\mathbb{Q}}(x) = G^*_{\infty,\mathbb{Q}}(y), \forall x, y \in \mathbb{Q}_f$ and Eq. (3.11) follows from that $G^*_{\infty,\mathbb{Q}}(x) > G^*_{\infty,\mathbb{Q}}(y), \forall x \in Q_f, \forall y \in \mathbb{Q}\setminus\mathbb{Q}_f$. Furthermore, since $G^*_{\infty,\mathbb{Q}}(x) \geq \epsilon > 0, \forall x \in \mathbb{Q}$, and $0 \leq T(\mathbb{Q}|x,u_x) \leq 1$, the equality in Eq. (3.11) holds if and only if $T(\mathbb{Q}_f|x,u_x) = 1$ and thereby $T(\mathbb{Q}\setminus\mathbb{Q}_f|x,u_x)) = 0$. Based on the recursion in (2.3), we have $G^*_{\infty,\mathbb{Q}}(x) = 1, \forall x \in \mathbb{Q}_f$. Hence, the set $\mathbb{Q}_f \subseteq \mathbb{Q}$ is an RCIS.

Next let us prove that $\forall x \in \mathbb{Q} \setminus \mathbb{Q}_f$, Eq.(3.8) holds. That is to prove that

$$G^*_{\infty,\mathbb{Q}}(x) \leq T(\mathbb{Q}_f|x,u_x) + \int_{\mathbb{Q}\setminus\mathbb{Q}_f} T(\mathbb{Q}_f|y,u_y)T(dy|x,u_x) + \frac{\rho^2}{1-\rho}. \tag{3.12}$$

By Theorem 7 in [51], the control input u_x is also optimal to the recursion (2.3). For all $k \in \mathbb{N}$, we have $\forall x \in \mathbb{Q}_f$, $G^*_{k,\mathbb{Q}}(x) = 1$ and $\forall x \in \mathbb{Q} \setminus \mathbb{Q}_f$,

$$G^*_{k+1,\mathbb{Q}}(x) = T(\mathbb{Q}_f|x,u_x) + \int_{\mathbb{Q}\setminus\mathbb{Q}_f} G^*_{k,\mathbb{Q}}(y)T(dy|x,u_x).$$

Let

$$\rho = \sup_{x \in \mathbb{Q} \setminus \mathbb{Q}_f} \int_{\mathbb{Q} \setminus \mathbb{Q}_f} T(dy|x, u_x).$$

Note that $0 \leq \rho < 1$. Then, $\forall x \in \mathbb{Q} \setminus \mathbb{Q}_f$, we can follow the induction rule to prove that

$$G^*_{k,\mathbb{Q}}(x) \leq T(\mathbb{Q}_f|x, u_x) + \int_{\mathbb{Q} \setminus \mathbb{Q}_f} T(\mathbb{Q}_f|y, u_y) T(dy|x, u_x) + \frac{\rho^2 - \rho^k}{1 - \rho},$$

which by taking limitation yields that (3.12) holds.

If-part: The proof for the existence of an RCIS $\mathbb{Q}_f \subseteq \mathbb{Q}$ is the same as that of the only if part. As shown above, the condition $T(\mathbb{Q}_f|x, u_x) = 1$ is equivalent to $G^*_{\infty,\mathbb{Q}}(x) = 1, \forall x \in \mathbb{Q}_f$. We can use induction to prove that $\forall x \in \mathbb{Q} \setminus \mathbb{Q}_f$,

$$G^*_{k,\mathbb{Q}}(x) \geq T(\mathbb{Q}_f|x, u_x) + \int_{\mathbb{Q} \setminus \mathbb{Q}_f} T(\mathbb{Q}_f|y, u_y) T(dy|x, u_x),$$

which further implies that

$$G^*_{\infty,\mathbb{Q}}(x) \geq T(\mathbb{Q}_f|x, u_x) + \int_{\mathbb{Q} \setminus \mathbb{Q}_f} T(\mathbb{Q}_f|y, u_y) T(dy|x, u_x).$$

One sufficient condition to guarantee $G^*_{\infty,\mathbb{Q}}(x) \geq \epsilon$ is (3.9), i.e.,

$$T(\mathbb{Q}_f|x, u_x) + \int_{\mathbb{Q} \setminus \mathbb{Q}_f} T(\mathbb{Q}_f|y, u_y) T(dy|x, u_x) \geq \epsilon.$$

The proof is completed. □

Remark 3.9. *The value of ρ is the largest probability that the next state y remains outside the RCIS $\mathbb{Q}_f$ from any $x \in \mathbb{Q} \setminus \mathbb{Q}_f$ under the optimal stationary policy in Lemma 2.8. Note that $\frac{\rho^2}{1-\rho}$ is the gap between the necessary condition and the sufficient condition. In addition, the second item in (3.8)–(3.9) denotes the probability that the state is steered into the RCIS $\mathbb{Q}_f$ by two transitions from $x \in \mathbb{Q} \setminus \mathbb{Q}_f$ with an intermediate state y outside $\mathbb{Q}_f$.*

Corollary 3.3. *Suppose that Assumption 2.2 holds and let $0 < \epsilon \leq 1$ be fixed. A nonempty set $\mathbb{Q}$ is an infinite-horizon ϵ-PCIS*

(i) only if *there exists an RCIS* $\mathbb{Q}_f \subseteq \mathbb{Q}$ *such that* $\forall x \in \mathbb{Q} \setminus \mathbb{Q}_f$,

$$T(\mathbb{Q}|x,u) \geq \epsilon \text{ for some } u \in \mathbb{U};$$

(ii) if *there exists an RCIS* $\mathbb{Q}_f \subseteq \mathbb{Q}$ *such that* $\forall x \in \mathbb{Q} \setminus \mathbb{Q}_f$,

$$T(\mathbb{Q}_f|x,u) + \epsilon T(\mathbb{Q} \setminus \mathbb{Q}_f|x,u) \geq \epsilon \text{ for some } u \in \mathbb{U}.$$

Proof. By Lemma 2.8 and Theorem 3.3, the necessary condition in Corollary 3.3 can be proven by showing that $\forall x \in \mathbb{Q} \setminus \mathbb{Q}_f$, there exists a $u \in \mathbb{U}$ such that

$$\begin{aligned} \epsilon \leq G^*_{\infty,\mathbb{Q}}(x) &= \int_{\mathbb{Q}_f} G^*_{\infty,\mathbb{Q}}(y) T(dy|x,u) + \int_{\mathbb{Q}\setminus\mathbb{Q}_f} G^*_{\infty,\mathbb{Q}}(y) T(dy|x,u) \\ &\leq T(\mathbb{Q}_f|x,u) + T(\mathbb{Q} \setminus \mathbb{Q}_f|x,u) \qquad (3.13) \\ &= T(\mathbb{Q}|x,u), \end{aligned}$$

where Eq. (3.13) follows from $0 < G^*_{\infty,\mathbb{Q}}(x) \leq 1, \forall x \in \mathbb{Q}$.

The sufficient condition in Corollary 3.3 can be proven by showing that $\forall x \in \mathbb{Q} \setminus \mathbb{Q}_f$, there exists a $u \in \mathbb{U}$

$$\begin{aligned} G^*_{\infty,\mathbb{Q}}(x) &= \int_{\mathbb{Q}_f} G^*_{\infty,\mathbb{Q}}(y) T(dy|x,u) + \int_{\mathbb{Q}\setminus\mathbb{Q}_f} G^*_{\infty,\mathbb{Q}}(y) T(dy|x,u) \\ &\geq T(\mathbb{Q}_f|x,u) + \epsilon T(\mathbb{Q} \setminus \mathbb{Q}_f|x,u), \qquad (3.14) \end{aligned}$$

where Eq. (3.14) follows from $G^*_{\infty,\mathbb{Q}}(x) \geq \epsilon > 0, \forall x \in \mathbb{Q}$. One sufficient condition to guarantee $G^*_{\infty,\mathbb{Q}}(x) \geq \epsilon$ is

$$T(\mathbb{Q}_f|x,u) + \epsilon T(\mathbb{Q} \setminus \mathbb{Q}_f|x,u) \geq \epsilon.$$

The proof is completed. □

Remark 3.10. *A nonempty set* $\mathbb{Q}$ *is an infinite-horizon* ϵ*-PCIS if there exists an RCIS* $\mathbb{Q}_f \subseteq \mathbb{Q}$ *such that* $\forall x \in \mathbb{Q} \setminus \mathbb{Q}_f$, $T(\mathbb{Q}_f|x,u) \geq \epsilon$ *for some* $u \in \mathbb{U}$. *This implication will facilitate the design of an algorithm for an infinite-horizon* ϵ*-PCIS, see Algorithm 3.4.*

Remark 3.11. *Considering the similarity between the reliability defined in [96] and the infinite-horizon invariance probability in this chapter, we can extend the results on infinite-horizon PICSs, including the existence condition above and the computational algorithms in the following, to the reliable control set in [95] to general stochastic systems.*

3.3.2 Infinite-horizon ϵ-PCIS Computation

This subsection will address the following problem.

Problem 3.2. *Given a set $\mathbb{Q} \in \mathcal{B}(\mathbb{S})$ and a prescribed probability $0 \leq \epsilon \leq 1$, compute an infinite-horizon ϵ-PCIS $\tilde{\mathbb{Q}} \subseteq \mathbb{Q}$.*

To handle this problem, the key point is to compute the infinite-horizon invariance probability $G^*_{\infty,\mathbb{Q}}$. For discrete spaces, it is shown that computationally tractable MILP can be used to compute the exact value of $G^*_{\infty,\mathbb{Q}}$. In this case, we can compute the largest infinite-horizon ϵ-PCIS by computing iteratively the stochastic backward reachable sets until convergence. For continuous spaces, it is in general computationally intractable to compute $G^*_{\infty,\mathbb{Q}}$ and the discretization method fails to work since the approximation error in (3.6) increases with the horizon. In this case, we design another computational algorithm based on the sufficient conditions in Remark 3.10.

Discrete state and control spaces

If the state and control spaces are discrete, we adopt the same assumptions as in Section 3.2.1. We will first show how to compute the exact value of $G^*_{\infty,\mathbb{Q}}$ in (2.3)–(2.4) through an MILP. Then, we will adapt Algorithm 3.1 to compute the largest infinite-horizon ϵ-PCIS within a given set.

MILP reformulation Since 0 is a trivial solution of (2.4), we cannot directly reformulate (2.3)–(2.4) as an LP, which is the traditional way to deal with infinite-horizon stochastic optimal control problems [109].

The following lemma provides a computationally tractable MILP reformulation when computing $G^*_{\infty,\mathbb{Q}}$

Lemma 3.7. *Given any set $\mathbb{Q} \subseteq \mathbb{S}$, the value of $G^*_{\infty,\mathbb{Q}}$ in (2.4) can be obtained by solving the MILP:*

$$\max_{g(x),\kappa(x,u)} \sum_{x\in\mathbb{Q}} g(x) \tag{3.15a}$$

$$\text{s.t. } \forall x \in \mathbb{Q},$$

$$g(x) \geq \sum_{y\in\mathbb{Q}} g(y)T(y|x,u), \forall u \in \mathbb{U}_x, \tag{3.15b}$$

$$g(x) \leq \sum_{y\in\mathbb{Q}} g(y)T(y|x,u) + (1-\kappa(x,u))\Delta, \forall u \in \mathbb{U}_x, \tag{3.15c}$$

$$\sum_{u\in\mathbb{U}_x} \kappa(x,u) \geq 1, \tag{3.15d}$$

$$0 \leq g(x) \leq 1, \kappa(x,u) \in \{0,1\}, \forall u \in \mathbb{U}_x, \tag{3.15e}$$

where Δ is a constant greater than one. That is, $G^*_{\infty,\mathbb{Q}}(x) = g^*(x), \forall x \in \mathbb{Q}$, where g^* is obtained from the optimal solution (g^*, κ^*) of the MILP (3.15). The optimal stationary policy is $\bar{\mu}^*_{\mathbb{Q}}(x) = u$ where $u \in \mathbb{U}_x$ such that $\kappa^*(x,u) = 1$ and κ^* is the optimal solution of the MILP (3.15).

Proof. From the monotone decrease of the sequence $(G^*_{0,\mathbb{Q}}, G^*_{1,\mathbb{Q}}, \ldots)$ and Lemma 2.8, $G^*_{\infty,\mathbb{Q}}$ is the maximum fixed point satisfying (2.4). Hence, the equivalent form of $G^*_{\infty,\mathbb{Q}}$ can be written as MILP (3.15), where the constraints (3.15b)–(3.15d) guarantee that there exists $u \in \mathbb{U}_x$ such that the equality in (2.4) holds. □

Computation algorithm As an adaption of Algorithm 3.1, the following algorithm provides a way to compute the largest infinite-horizon ϵ-PCIS within $\mathbb{Q}$.

Algorithm 3.3 Infinite-horizon ϵ-PCIS

1: Initialize $i = 0$ and $\mathbb{P}_i = \mathbb{Q}$.
2: Compute $G^*_{\infty,\mathbb{P}_i}(x)$ for all $x \in \mathbb{P}_i$.
3: Compute the set $\mathbb{P}_{i+1} = \mathcal{R}^*_{\epsilon,\infty}(\mathbb{P}_i)$.
4: If $\mathbb{P}_{i+1} = \mathbb{P}_i$, stop. Else, set $i = i + 1$ and go to step 2.

The difference between Algorithms 3.1 and 3.3 is that the value of $G^*_{\infty,\mathbb{P}_i}(x)$, instead of $V^*_{0,\mathbb{P}_i}(x)$, $\forall x \in \mathbb{P}_i$, is computed by (3.15) (replacing $\mathbb{Q}$ with $\mathbb{P}_i$). Furthermore, the updated set $\mathbb{P}_{i+1} = \mathcal{R}^*_{\epsilon,\infty}(\mathbb{P}_i)$, which is a stochastic backward reachable set within $\mathbb{P}_i$ with respect to infinite horizon and a probability level ϵ. The following theorem provides the convergence of $\mathbb{P}_i$ and shows that the resulting set $\tilde{\mathbb{Q}}$ by this algorithm is an infinite-horizon ϵ-PCIS.

Theorem 3.4. *For discrete state and control spaces, Algorithm 3.3 converges in a finite number of iterations and generates the largest infinite-horizon ϵ-PCIS within $\mathbb{Q}$. Furthermore, at each iteration, the infinite-horizon invariance probability $G^*_{\infty,\mathbb{P}_i}(x), \forall x \in \mathbb{P}_i$, can be computed via the MILP (3.15).*

Proof. The finite-step convergence of Algorithm 3.3 follows from the finite cardinality of the set $\mathbb{Q}$. Similar to Theorem 3.1, the generated infinite-horizon ϵ-PCIS is the largest one within $\mathbb{Q}$. The MILP reformulation refers to Lemma 3.7. □

Remark 3.12. *When implementing Algorithm 3.3 to a system with discrete spaces, the maximal iteration number is $|\mathbb{Q}|$. An MILP is used to compute the value of $G^*_{\infty,\mathbb{P}_i}(x)$, $\forall x \in \mathbb{P}_i$, at each iteration. The number of real-valued decision values is at most $|\mathbb{Q}|$, the number of binary decision values is at most $|\mathbb{Q}||\mathbb{U}|$, and the number of constraints is at most $|\mathbb{Q}|(2|\mathbb{U}| + 3)$. In general, MILPs are NP-hard and can be solved by cutting plane algorithm or branch-and-bound algorithm [110]. Some advanced softwares have been developed to solve large MILPs efficiently [111], [112].*

Continuous state and control spaces

If the state and control spaces are continuous, it is computationally intractable to compute the exact value of infinite-horizon invarinace probability $G^*_{\infty,\mathbb{Q}}(x)$. Based on Remark 3.10, this subsection provides another way to compute an infinite-horizon ϵ-PCIS within a given set $\mathbb{Q}$.

Different from Algorithm 3.3, which computes iteratively the stochastic backward reachable sets, the following algorithm generates an infinite-horizon ϵ-PCIS by computing a backward stochastic reachable set from the RCIS $\mathbb{Q}_f$ contained in $\mathbb{Q}$.

Algorithm 3.4 Infinite-horizon ϵ-PCIS

1: Compute the RCIS within $\mathbb{Q}$, denoted by $\mathbb{Q}_f$.
2: Compute the stochastic backward reachable set from $\mathbb{Q}_f$, i.e.,

$$\tilde{\mathbb{Q}} = \{x \in \mathbb{Q} \mid \exists u \in \mathbb{U}, \int_{\mathbb{Q}_f} T(dy|x,u) \geq \epsilon\}.$$

The first step in Algorithm 3.4 is the computation of RCIS within a given set, which is a well-studied topic in the literature [26]–[28]. Then, based on RCIS $\mathbb{Q}_f$ within $\mathbb{Q}$, the stochastic backward reachable set

$$\tilde{\mathbb{Q}} = \{x \in \mathbb{Q} \mid \exists u \in \mathbb{U}, \int_{\mathbb{Q}_f} T(dy|x,u) \geq \epsilon\}$$

is an infinite-horizon ϵ-PCIS within $\mathbb{Q}$. In comparison with Algorithms 3.1–3.3, the iteration is avoided in Algorithm 3.4, which only needs two steps.

Remark 3.13. *Note that the resulting set from Algorithm 3.4 is in general not the largest infinite-horizon ϵ-PCIS within the given set $\mathbb{Q}$. It is possible to obtain a larger infinite-horizon ϵ-PCIS if we can reformulate the existence conditions in Theorem 3.3 and Corollary 3.3 in a recursive form and thereby modify Algorithm 3.4 to be a recursive algorithm.*

Remark 3.14. *The complexity of Algorithm 3.4 depends on the computation of the RCIS [26]–[28], [47], and the computation of the backward stochastic reachable set. The later can be reformulated as a chance-constrained problem and then approximately solved. Some results on computation of the backward stochastic reachable set have been reported in [113]. The first example in Section 3.4 will show how to compute the backward stochastic reachable set.*

3.4 Examples

In this section, two examples are provided to illustrate the effectiveness of the proposed theoretical results. The first one is concerned with comparison between PCIS and RCIS. Then we consider an application to motion planning of a mobile robot in a partitioned space with obstacles.

3.4.1 Example 1: Comparison between PCIS and RCIS

Consider the following example from [114]:

$$x_{k+1} = Ax_k + Bu_k + w_k,$$

where $A = \begin{bmatrix} 1.6 & 1.1 \\ -0.7 & 1.2 \end{bmatrix}$ and $B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. The control input is constrained by $|u_k| \leq 0.25$. We consider w_k to be either non-stochastic or stochastic when computing RCIS and PCIS, respectively. The region of interest is $\mathbb{Q} = \{x \in \mathbb{R}^2 \mid \|x\|_\infty \leq 0.5\}$. We will compare the largest RCIS and PCIS within $\mathbb{Q}$.

To derive an RCIS for this system, we assume the disturbance belongs to the compact set $\mathbb{W} = \{w \in \mathbb{R}^2 \mid \|w\|_\infty \leq 0.05\}$. By using the methods in [28], [43], we obtain the largest RCIS, which is the blue region shown in Figure 3.1. The gray region is an infinite-horizon ϵ-PCIS described in the end of this example.

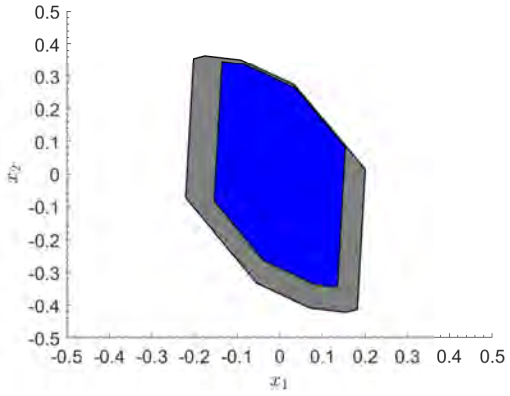


Figure 3.1: Computations of the largest RCIS (blue) and an infinite-horizon ϵ-PCIS with $\epsilon = 0.80$ (gray) by Algorithm 3.4 for Example 1.

When computing a finite-horzion PCIS, assume that elements of w_k are i.i.d. Gaussian random variables with zero mean and variance $\sigma^2 = 1/30^2$. This system can be represented as a triple $\mathsf{SCS} = \{\mathbb{X}, \mathbb{U}, T\}$:

$$\begin{cases} \mathbb{S} = \mathbb{R}^2, \\ \mathbb{U} = \{u \in \mathbb{R} \mid |u| \leq 0.25\}, \\ t(x_{k+1}|x_k, u_k) = \psi(\Lambda^{-1}(x_{k+1} - Ax_k - Bu_k)), \end{cases}$$

where $\psi(\cdot)$ is the density function of the standard normal distribution and $\Lambda = \mathrm{diag}\{\sigma, \sigma\}$. In this case, since the Lipschitz constant L in Assumption 3.1 is small, we ignore the approximation error τ_0 in (3.7). We discretize the continuous spaces and implement Algorithm 3.2 to compute the N-step ϵ-PCIS $\tilde{\mathbb{Q}}$. First consider $N = 5$ and $\epsilon = 0.80$. Figure 3.2(a) shows the evolution of the set $\mathbb{P}_i$ in Algorithm 3.2. The color indicates the corresponding N-step invariance probability $p^*_{N,\mathbb{P}_i}(x)$ and the z-axes the iteration index i. The algorithm converges in 8 steps. Figure 3.2(b) shows $\mathbb{P}_8$, which corresponds to the N-step ϵ-PCIS $\tilde{\mathbb{Q}}$ for $N = 5$ and $\epsilon = 0.80$. Figures 3.3 and 3.4 show the N-step ϵ-PCISs for $N = 1, 3$, $\epsilon = 0.80$ and $N = 5$, $\epsilon = 0.70, 0.90$, respectively. Note that in all cases the probability density is concentrated in the interior of the sets. Note also that the numerical results indicate that the sets are nonincreasing in both N and ϵ, as expected from the analysis in Section 3.2.

When computing an infinite-horizon PCIS, we choose the same bound on

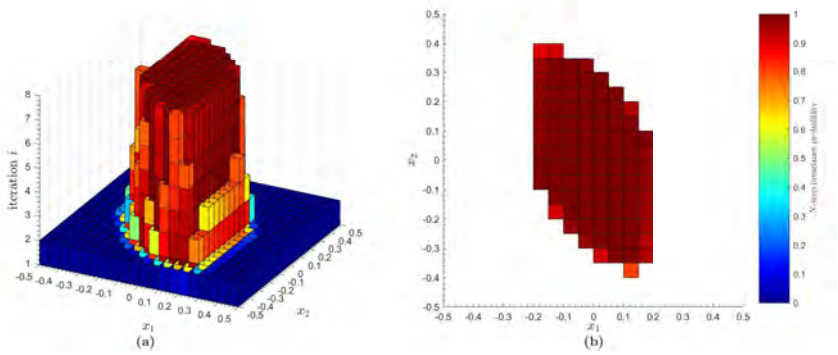


Figure 3.2: Computation of N-step ϵ-PCIS with $N = 5$ and $\epsilon = 0.80$ for Example 1: (a) The sets $\mathbb{P}_i$ and the corresponding N-step invariance probability in Algorithm 3.2; (b) The N-step ϵ-PCIS $\tilde{\mathbb{Q}}$.

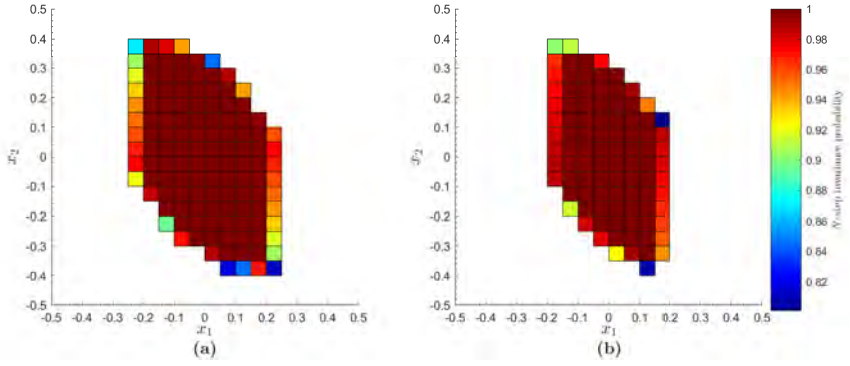


Figure 3.3: N-step ϵ-PCISs for Example 1 with $\epsilon = 0.80$ but different values of N: (a) $N = 1$; (b) $N = 3$.

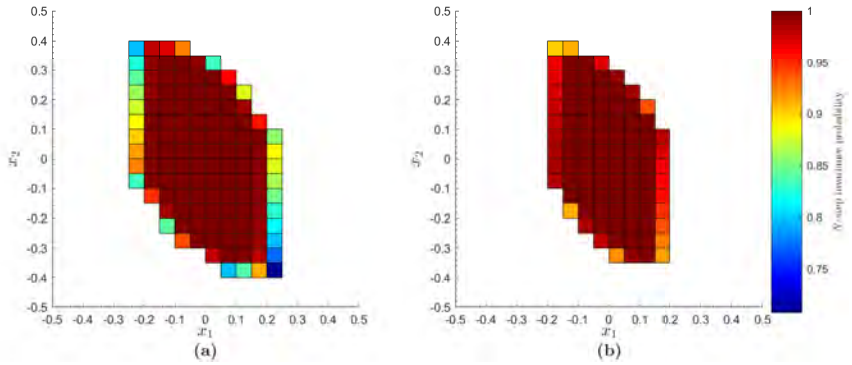


Figure 3.4: N-step ϵ-PCISs for Example 1 with $N = 5$ but different values of ϵ: (a) $\epsilon = 0.70$; (b) $\epsilon = 0.90$.

the disturbance as for the RCIS. The elements of w_k are truncated i.i.d. Gaussian random variables with zero mean and variance $\sigma^2 = 1/30^2$. Denote the largest RCIS computed above by $\mathbb{Q}_f = \{x \in \mathbb{R}^2 \mid Hx \leq h\}$, where the matrix H and the vector h are with appropriate dimensions. As stated in Algorithm 3.4, the one-step stochastic backward reachable set from the RCIS associated with probability 0.80 is an infinite-horizon ϵ-PCIS with $\epsilon = 0.80$, i.e.,

$$\tilde{\mathbb{Q}} = \{x \in \mathbb{Q} \mid \exists u \in \mathbb{U}, \Pr\{H(Ax + Bu + w) \leq h\} \geq 0.80\}.$$

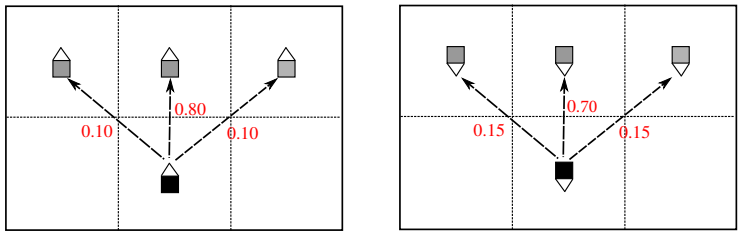
This set can be represented as

$$\tilde{\mathbb{Q}} = \{x \in \mathbb{Q} \mid \exists u \in \mathbb{U}, H(Ax + Bu) + h' \leq h\},$$

where h' is the optimal solution of the chance constrained program

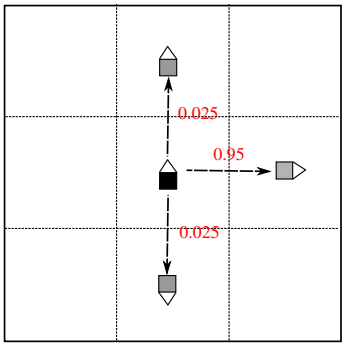
$$\begin{aligned} &\min \sum_j h'_j \\ &\text{s.t. } \Pr\{Hw \leq h'\} = 0.8. \end{aligned}$$

This program can be numerically solved by using the methods in [115], [116]. The resulting infinite-horizon ϵ-PCIS with $\epsilon = 0.80$ is the gray region shown in Figure 3.1. This region is obviously a superset of the RCIS in blue.

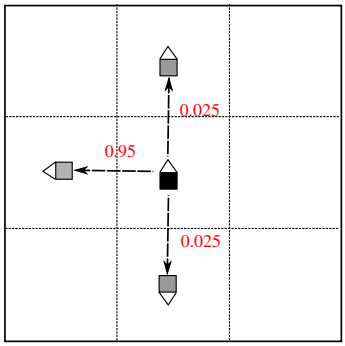


(a) Forward (FR)

(b) Backward (BK)



(c) Turn right and forward (TRFR)

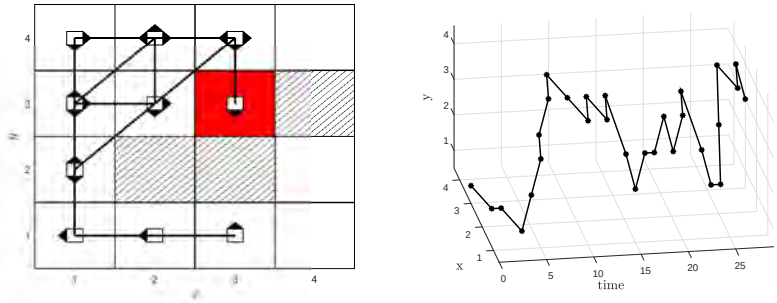


(d) Turn left and forward (TLFR)

Figure 3.5: Transition probability under actions for Example 2.

3.4.2 Example 2: Motion Planning

The motion planning example in [117] is adapted to seek an infinite-horizon PCIS within the workspace for a mobile robot. The state of the robot is abstracted by its cell coordinate, i.e., $(p_x, p_y) \in \{1, 2, 3, 4\}^2$, and its four possible orientations $\{\mathcal{E}, \mathcal{W}, \mathcal{S}, \mathcal{N}\}$. Due to the actuation noise and drifting, the robot motion is stochastic. Here, we restrict the action space to be $\{\mathrm{FR}, \mathrm{BK}, \mathrm{TRFR}, \mathrm{TLFR}\}$, under which the possible transitions are shown in Figure 3.5. Specifically, action "FR" means driving forward for 1 unit. As illustrated in the figure, the probability for that is 0.80. The probability of drifting forward to the left or the right by 1 unit is 0.10. Action "BK" can be similarly defined. Action "TRFR" means turning right $\pi/2$ and driving forward for 1 unit, of which the probability is 0.95. The probability of driving forward for 1 unit without turning right is 0.025 and the probability of turning right for π and driving forward for 1 unit is 0.025. Similarly, we can define the action "TLFR".

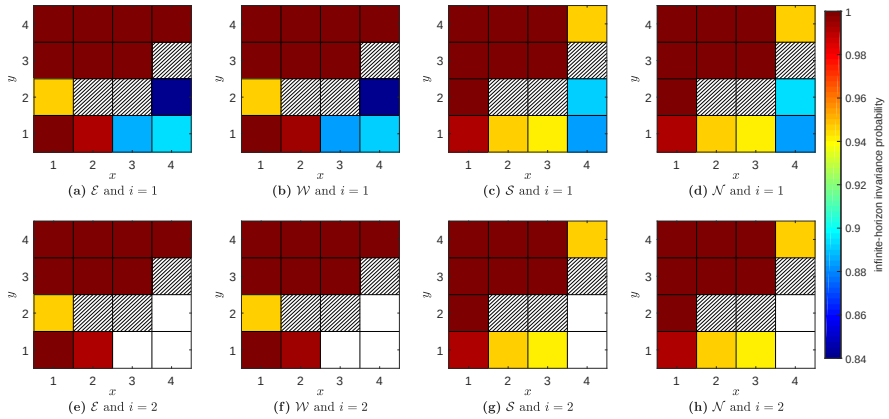


(a) (b)

Figure 3.6: One simulated trajectory of 30 time steps starting from $(3, 1, \mathcal{N})$ and ending at $(3, 4, \mathcal{S})$ in Example 2. (a) The state trajectory with indication of the robot orientation. (b) The trajectory of the position (p_x, p_y) evolving over time.

Consider the partitioned workspace shown in Figure 3.6(a), where the shadowed cells are occupied by obstacles and the red cell is an absorbing region, i.e., when the robot enters in this region it will stay there forever. We construct an MDP with 64 states and 4 actions. The transition relation and probability can be defined based on the above description. We compute the largest infinite-horizon ϵ-PCIS with $\epsilon = 0.90$ within the safe state space, i.e., the remaining of the state space by excluding the states associated with the obstacles.

By implementing Algorithm 3.3, the computed sets $\mathbb{P}_i$ and the corresponding infinite-horizon invariance probability $p^*_{\infty,\mathbb{P}_i}(x)$ are shown in Figure 3.7, of which each subfigure corresponds to one orientation in $\{\mathcal{E}, \mathcal{W}, \mathcal{S}, \mathcal{N}\}$. The first row of Figure 3.7 shows the results after the first iteration, where we can see that the infinite-horizon invariance probability $p^*_{\infty,\mathbb{P}_i}(x)$ at $x = (4, 2, \mathcal{E})$ and $x = (4, 2, \mathcal{W})$ is less than $\epsilon = 0.90$. Algorithm 3.3 converges in 2 steps and generates the largest infinite-horizon ϵ-PCIS $\tilde{\mathbb{Q}}$ with $\epsilon = 0.90$ shown in Figure 3.7(e)–3.7(h). This invariant set provides a region where the admissible action can drive the robot without colliding with the obstacles with probability 0.90. By implementing the optimal policy obtained in Lemma 3.7, we run a state trajectory starting from $(3, 1, \mathcal{N})$ as shown in Fig. 3.6(b). We can see that this trajectory is collision-free and finally ends at the absorbing region $(3, 3, \mathcal{S})$.



(a) $\mathcal{E}$ and $i = 1$ (b) $\mathcal{W}$ and $i = 1$ (c) $\mathcal{S}$ and $i = 1$ (d) $\mathcal{N}$ and $i = 1$

(e) $\mathcal{E}$ and $i = 2$ (f) $\mathcal{W}$ and $i = 2$ (g) $\mathcal{S}$ and $i = 2$ (h) $\mathcal{N}$ and $i = 2$

Figure 3.7: The sets $\mathbb{P}_i$ and the corresponding infinite-horizon invariance probability in Example 2 when computing the largest infinite-horizon ϵ-PCIS with $\epsilon = 0.90$ by Algorithm 3.3.

3.5 Summary

We investigated the extension of set invariance in a stochastic sense for control systems. We proposed finite- and infinite-horizon ϵ-PCISs, and provided some fundamental properties. We designed iterative algorithms to compute the PCIS within a given set. For systems with discrete state and control spaces, finite- and infinite-horizon ϵ-PCISs can be computed by solving an LP and an MILP at each iteration, respectively. We proved that the iterative algorithms were computationally tractable and can be terminated in a finite number of steps. For systems with continuous state and control spaces, we established the approximation of stochastic control systems and proved its convergence when computing finite-horizon ϵ-PCIS. In addition, thanks to the sufficient conditions for the existence of infinite-horizon ϵ-PCIS, we can compute an infinite-horizon ϵ-PCIS by the stochastic backward reachable set from the RCIS contained in it. Numerical examples were given to illustrate the theoretical results.

Chapter 4

Computation of Invariant Covers

Both stochastic invariance and robust invariance can express the safety property for control systems. In the previous chapter, we studied the stochastic invariance. This chapter will revisit the robust invariance for networked controlled systems. We consider some fundamental problems concerning existence and computation of an invariant cover for uncertain discrete-time linear control systems subject to state and control constraints. An invariant cover quantifies the information needed by a controller to enforce a robust invariance specification. We develop necessary and sufficient conditions on the existence of an invariant cover for a polytopic set of states. The conditions can be checked by solving a set of linear programs (LPs), one for each extreme point of the state set. Based on these conditions, we give upper and lower bounds on the minimal cardinality of the invariant cover, and design an iterative algorithm with finite-time convergence to compute an invariant cover. We further show in two examples how to use an invariant cover in the design of a coder–controller pair that ensures invariance of a given set for a networked control system with a finite data-rate communication.

The remainder of the chapter is organized as follows. Section 4.1 gives the introduction. The problem statement is given in Section 4.2. Section 4.3 addresses the existence of an invariant cover and Section 4.4 gives bounds on its minimal cardinality. Section 4.5 provides an algorithm for computing an invariant cover. The examples in Section 4.6 detail how to use the invariant cover to design coder–controllers for networked control systems. Section 4.7 concludes the chapter.

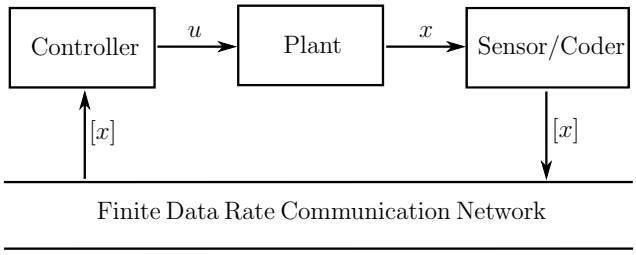


Figure 4.1: Coder-controller feedback loop, where x is the measured state, $[x]$ is the encoded state, and u is the control.

4.1 Introduction

In a networked control system a plant is connected with a controller through a communication network [29]–[31], as shown in Figure 4.1. Networked control systems are in widespread use in a variety of application areas, for example, smart buildings [118] and intelligent transportation [119]. Since the data-rate of a communication channel is usually limited, a central question is how much information is needed by the controller to enforce a given specification.

Feedback control under limited data-rate has been widely studied [120]–[122]. One well-known result is that the critical data-rate necessary for stabilization of linear systems depends on the unstable poles of their open-loop system [123]. In [124], the notion of topological feedback entropy, which is an extension of topological entropy [125], [126], has been used to quantify the information necessary for stabilization of nonlinear control systems.

Invariance is one of the most fundamental concepts in systems and control [46], [47]. In the context of networked control systems, the minimal data-rate necessary for set invariance under feedback control was studied in [124], [127]. It was shown in [124] that a finite topological feedback entropy is necessary to achieve invariance. Later, the notion of invariance entropy was proposed for continuous-time deterministic control systems based on spanning sets [127]. Equivalence between these two notions was established for discrete-time control systems under the assumption of strong invariance in [128].

The notion of invariance feedback entropy was first proposed in [32] for generalizing the notion of invariance entropy [127] to *uncertain* discrete-time control systems and was further explored in [33], [34]. It was shown in [34]

that the invariance feedback entropy of a given set of states is finite if and only if an invariant cover exists for this set. An invariant cover is a pair consisting of a finite cover of the given set and a map from this cover to the control set (see Definition 4.1). We remark that the invariant cover plays an important role in designing a coder–controller which achieves a finite data-rate and ensures invariance.

This chapter establishes fundamental results on invariant cover for uncertain discrete-time linear control systems. The main contributions are summarized as follows:

(C4.1) We develop two necessary and sufficient conditions for the existence of an invariant cover for a given polytopic set (Theorems 4.1 and 4.2). They suggest a computationally tractable method of determining whether an invariant cover exists through LPs.

(C4.2) Based on these conditions, we give upper and lower bounds on the minimal cardinality of an invariant cover (Theorem 4.3). As a complement to [34], this upper bound is valid for the invariance feedback entropy and the minimal data-rate necessary for invariance.

(C4.3) We provide an iterative algorithm to compute an invariant cover (Algorithm 4.1) and prove its finite-time convergence (Theorem 4.4). The performance of the algorithm is illustrated in two examples that use an invariant cover to design a static coder–controller pair for a networked control system with a finite data-rate to enforce the invariance of a given set.

4.2 Problem Formulation

Consider a discrete-time linear system in the form of

$$x_{k+1} = Ax_k + Bu_k + w_k, \tag{4.1}$$

where $x_k \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$ is the state, $u_k \in \mathbb{U} \subset \mathbb{R}^{n_u}$ the control input and $w_k \in \mathbb{W} \subset \mathbb{R}^{n_x}$ the disturbance input, and A, B are matrices with appropriate dimensions. The state and control sets $\mathbb{X}$ and $\mathbb{U}$ and the disturbance set $\mathbb{W}$ are each assumed to be convex polyhedral sets:

$$\mathbb{X} = \{x \in \mathbb{R}^{n_x} \mid F_x x \leq f_x\},$$

$$\mathbb{U} = \{u \in \mathbb{R}^{n_u} \mid F_u u \leq f_u\},$$
$$\mathbb{W} = \{w \in \mathbb{R}^{n_x} \mid F_w w \leq f_w\},$$

where F_x, F_u, F_w, and f_x, f_u, f_w are matrices and vectors with appropriate dimensions. We assume that $\mathbb{U}$ and $\mathbb{W}$ are compact sets and define $\mathbb{Q}$ as a compact subset of $\mathbb{X}$. We further assume that $\mathbb{Q}$ is full-dimensional. If this is not the case, i.e., if $\mathbb{Q}$ lies in an affine subspace of $\mathbb{R}^{n_x}$, then we assume that $\mathbb{Q} \subseteq \mathbb{S}_1 \times \mathbb{S}_2$ and $\mathbb{Q} \cap \mathbb{S}_1 = \{x^0\}$ for some $x^0 \in \mathbb{R}^{n_x}$, and we apply the following arguments to the subspace $\mathbb{S}_2$ and the projection of $\mathbb{Q}$ (assumed full-dimensional) onto $\mathbb{S}_2$.

A cover of a set $\mathbb{Q}$ is a collection of sets whose union includes $\mathbb{Q}$ as a subset. Next we define an invariant cover of $\mathbb{Q} \subseteq \mathbb{X}$, which is a pair consisting of a finite cover of $\mathbb{Q}$ and a map from this cover to the control set $\mathbb{U}$. Each state set from the finite cover can be driven to the set $\mathbb{Q}$ by means of a single control input generated by the map.

Definition 4.1. *[34] A cover $\mathcal{A}$ of a nonempty set $\mathbb{Q}$ and a function $G : \mathcal{A} \to \mathbb{U}$ is an* invariant cover $(\mathcal{A}, G)$ *of the system (4.1) and $\mathbb{Q}$ if $\mathcal{A}$ is finite and, for all $\mathbb{X}^{\mathrm{ic}} \in \mathcal{A}$, $A\mathbb{X}^{\mathrm{ic}} \oplus \{BG(\mathbb{X}^{\mathrm{ic}})\} \subseteq \mathbb{Q} \ominus \mathbb{W}$.*

Remark 4.1. *In the context of networked control systems, an invariant cover is used to define the invariance feedback entropy in [34]. Note that an invariant cover $(\mathcal{A}, G)$ immediately provides a static coder–controller: for any $x \in \mathbb{Q}$, the coder transmits one of the sets $\mathbb{X}^{\mathrm{ic}} \in \mathcal{A}$ that contains x to the controller and the controller implements $G(\mathbb{X}^{\mathrm{ic}})$ to guarantee invariance (Figure 4.1). It is shown in [34] that the data-rate of the static coder–controller under the invariant cover $(\mathcal{A}, G)$ is $\log_2 |\mathcal{A}|$ bits per time unit.*

Example 4.1. *Consider the linear scalar system*

$$x_{k+1} = 2x_k + u_k + w_k,$$

with $\mathbb{X} = \mathbb{R}$, $\mathbb{U} = [-1, 1]$, and $\mathbb{W} = [-0.4, 0.4]$. Let $\mathbb{Q} = [-0.6, 0.6]$. It is easy to verify that $\mathbb{Q}$ is an RCIS. Let $\mathbb{X}_1^{\mathrm{ic}} = [-0.6, -0.4]$, $\mathbb{X}_2^{\mathrm{ic}} = [-0.4, -0.2]$, $\mathbb{X}_3^{\mathrm{ic}} = [-0.2, 0]$, $\mathbb{X}_4^{\mathrm{ic}} = [0, 0.2]$, $\mathbb{X}_5^{\mathrm{ic}} = [0.2, 0.4]$, and $\mathbb{X}_6^{\mathrm{ic}} = [0.4, 0.6]$. Define $\mathcal{A} = \{\mathbb{X}_i^{\mathrm{ic}}\}_{i=1}^6$ and the map $G : \mathcal{A} \to \mathbb{U}$ with $G(\mathbb{X}_1^{\mathrm{ic}}) = 1$, $G(\mathbb{X}_2^{\mathrm{ic}}) = 0.6$, $G(\mathbb{X}_3^{\mathrm{ic}}) = 0.2$, $G(\mathbb{X}_4^{\mathrm{ic}}) = -0.2$, $G(\mathbb{X}_5^{\mathrm{ic}}) = -0.6$, and $G(\mathbb{X}_6^{\mathrm{ic}}) = -1$. We can verify that $(\mathcal{A}, G)$ is an invariant cover for this system and the set $\mathbb{Q}$. The data-rate of the static coder–controller defined by this invariant cover $(\mathcal{A}, G)$ is $\log_2 6$ bits per time unit.

Recall the robust controlled invariant set (RCIS) in Definition 2.21. An RCIS is a set that can be made invariant by a state feedback control law under any admissible disturbance. The computation of such sets is widely studied in the literature, e.g., [27]. From Definition 4.1 it is obvious that $\mathbb{Q}$ must be an RCIS in order that there exists an invariant cover $(\mathcal{A}, G)$ of the system (4.1) and $\mathbb{Q}$. In this chapter, we firstly consider the existence of an invariant cover.

Problem 4.1. *Consider the system (4.1) and a set $\mathbb{Q}$. Find necessary and sufficient conditions such that there exists an invariant cover $(\mathcal{A}, G)$ of the system (4.1) and the set $\mathbb{Q}$.*

From Remark 4.1 it follows that the data-rate of the (static) coder–controller decreases as the cardinality of the invariant cover decreases. We define the minimal cardinality of the invariant cover as follows:

$$|\mathcal{A}|^* = \inf |\mathcal{A}| \quad \text{s.t.} \quad \begin{array}{l}(\mathcal{A}, G) \text{ is an invariant} \\ \text{cover for (4.1) and } \mathbb{Q}\end{array}$$

If an invariant cover is known to exist, we consider the following problem.

Problem 4.2. *If an invariant cover $(\mathcal{A}, G)$ exists for the system (4.1) and a set $\mathbb{Q}$, provide upper and lower bounds on the minimal cardinality of the invariant cover.*

We further consider the computation problem.

Problem 4.3. *Design an algorithm to compute an invariant cover $(\mathcal{A}, G)$ for the system (4.1) and a set $\mathbb{Q}$ whenever such invariant cover exists.*

4.3 Existence Conditions

This section focuses on Problem 4.1. The sets $\mathbb{Q}$ and $\mathbb{Q} \ominus \mathbb{W}$ are assumed to have the H-representations

$$\begin{aligned}\mathbb{Q} &= \{x \in \mathbb{R}^{n_x} \mid Qx \leq q\}, \\ \mathbb{Q} \ominus \mathbb{W} &= \{x \in \mathbb{R}^{n_x} \mid Px \leq p\},\end{aligned}$$

where $q \in \mathbb{R}^{n_q}$, $p \in \mathbb{R}^{n_p}$ and Q, P are matrices with appropriate dimensions. We assume that the rows of Q are normalized so that $\|[Q]_i\| = 1$, $\forall i \in \mathbb{N}_{[1,n_q]}$, and that $q > 0$ so that the origin lies in the interior of $\mathbb{Q}$. For

any full-dimensional $\mathbb{Q}$, this can be ensured by redefining the state and disturbance input of (4.1) as $x_k - x^0$ and $w_k - x^0 + Ax^0$, respectively, for any x^0 in the interior of $\mathbb{Q}$.

For $x \in \mathbb{Q}$, we say that a control u is feasible for x if it drives x to $\mathbb{Q}$ for all $w \in \mathbb{W}$. We denote by $\Gamma \subseteq \mathbb{R}^{n_x+n_u}$ the set of all (x, u) such that u is feasible for x. The set Γ can be written as a compact polytopic set:

$$\Gamma = \Big\{(x,u) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \mid \underbrace{Qx \leq q}_{x\in\mathbb{Q}}, \underbrace{F_u u \leq f_u}_{u\in\mathbb{U}}, \underbrace{PAx + PBu \leq p}_{Ax+Bu\in\mathbb{Q}\ominus\mathbb{W}}\Big\}. \tag{4.2}$$

Define the map $\Pi : \mathbb{U} \to 2^{\mathbb{R}^{n_x}}$ as

$$\Pi(u) = \{x \in \mathbb{R}^{n_x} \mid (x,u) \in \Gamma\}. \tag{4.3}$$

For convenience we set $\Pi(u) = \emptyset$ if $u \notin \mathbb{U}$. For given $u \in \mathbb{U}$, the set $\Pi(u)$ has the property that any state $x \in \Pi(u)$ is steered into $\mathbb{Q}$ in a single time-step.

Lemma 4.1. *For any given $u \in \mathbb{U}$, if $\Pi(u) \neq \emptyset$, then*

$$A\Pi(u) \oplus \{Bu\} \subseteq \mathbb{Q} \ominus \mathbb{W}.$$

Proof. If $\Pi(u) \neq \emptyset$, then (4.2)–(4.3) imply that $Ax + Bu \subseteq \mathbb{Q} \ominus \mathbb{W}$ for all $x \in \Pi(u)$, i.e., $A\Pi(u) \oplus \{Bu\} \subseteq \mathbb{Q} \ominus \mathbb{W}$. □

The following lemma gives a necessary and sufficient condition for the existence of an invariant cover.

Lemma 4.2. *An invariant cover $(\mathcal{A}, G)$ of the system (4.1) and the set $\mathbb{Q}$ exists if and only if there exist a finite number $N \in \mathbb{N}$ and a set $\{u_i \in \mathbb{U}\}_{i=1}^{N}$ such that*

$$\bigcup_{i=1}^{N} \Pi(u_i) = \mathbb{Q}. \tag{4.4}$$

Proof. The sufficiency directly follows from Definition 4.1 and Lemma 4.1. We prove the necessity as follows. Let $(\mathcal{A}, G)$ be an invariant cover for (4.1) and $\mathbb{Q}$, where $\mathcal{A} = \{\mathbb{X}_i^{\mathrm{ic}}\}_{i=1}^{N_{\mathrm{ic}}}$ and for each $\mathbb{X}_i^{\mathrm{ic}}$, there exists $u_i^{\mathrm{ic}} = G(\mathbb{X}_i^{\mathrm{ic}}) \in \mathbb{U}$ such that $A\mathbb{X}_i^{\mathrm{ic}} \oplus Bu_i^{\mathrm{ic}} \subseteq \mathbb{Q} \ominus \mathbb{W}$. Here N_{ic} is the cardinality of $\mathcal{A}$.

From the definition of $\Pi(u)$ in (4.3), it follows that any set $\mathbb{Y} \subseteq \mathbb{Q}$ such that $A\mathbb{Y} \oplus \{Bu\} \subseteq \mathbb{Q} \ominus \mathbb{W}$ for some $u \in \mathbb{U}$ is a subset of $\Pi(u)$. This implies that, $\forall i \in \mathbb{N}_{[1,N_{\mathrm{ic}}]}$, $\mathbb{X}_i^{\mathrm{ic}} \subseteq \Pi(u_i^{\mathrm{ic}}) \subseteq \mathbb{Q}$, and since $\cup_{i=1}^{N_{\mathrm{ic}}} \mathbb{X}_i^{\mathrm{ic}} = \mathbb{Q}$, it follows that $\{u_i^{\mathrm{ic}}\}_{i=1}^{N_{\mathrm{ic}}}$ satisfies $\cup_{i=1}^{N_{\mathrm{ic}}} \Pi(u_i^{\mathrm{ic}}) = \mathbb{Q}$. □

Lemma 4.2 is important to prove the following result.

Theorem 4.1. *An invariant cover $(\mathcal{A}, G)$ of the system (4.1) and the set $\mathbb{Q}$ exists if and only if for all $x \in \mathbb{Q}$, there exists a control input $u \in \mathbb{U}$ such that*

$$\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \Pi(u) \text{ for some } r > 0.$$

Proof. We first show that the existence of an invariant cover for (4.1) and $\mathbb{Q}$ implies that for all $x \in \mathbb{Q}$ there exists $u \in \mathbb{U}$ and $r > 0$ such that $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \Pi(u)$. For given $x \in \mathbb{Q}$, $u \in \mathbb{U}$ and $r > 0$, let

$$\Theta(u) = \{z \mid PAz + PBu \leq p\},$$
$$\mathbb{P}_r(x) = \{z \mid PA(z - x) \leq r\mathbf{1}\}.$$

Furthermore, suppose that the rows of PA are normalised with $\|[PA]_i\| = 1$, $\forall i \in \mathbb{N}_{[1,n_p]}$ (this can be assumed without loss of generality by appropriately scaling P and p). Then the n_x-dimensional ball $\mathbb{B}_r(x)$ is a subset of $\Theta(u)$ if and only if $\mathbb{P}_r(x)$ is a subset of $\Theta(u)$ (since the polytopes $\Theta(u)$ and $\mathbb{P}_r(x)$ share the same set of face normals and since each face of $\mathbb{P}_r(x)$ is contained in a supporting hyperplane of $\mathbb{B}_r(x)$). Moreover, from $\Pi(u) = \Theta(u) \cap \mathbb{Q}$ it follows that $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \Pi(u)$ if $\mathbb{P}_r(x) \cap \mathbb{Q} \subseteq \Pi(u)$. But $\mathbb{P}_r(x) \cap \mathbb{Q} \subseteq \Pi(u)$ requires that

$$\left\{\begin{matrix} z \mid PA(z - x) \leq r\mathbf{1} \\ Qz \leq q \end{matrix}\right\} \subseteq \left\{\begin{matrix} z \mid PAz + PBu \leq p \\ Qz \leq q \end{matrix}\right\},$$

and by linear programming duality this is equivalent to the condition that there exists a matrix S with non-negative elements such that

$$S \begin{bmatrix} PA \\ Q \end{bmatrix} = \begin{bmatrix} PA \\ Q \end{bmatrix}, \tag{4.5}$$

$$S \begin{bmatrix} r\mathbf{1} + PAx \\ q \end{bmatrix} \leq \begin{bmatrix} p - PBu \\ q \end{bmatrix}. \tag{4.6}$$

Replacing q on the left side of (4.6) with $Qx + q - Qx$ and using (4.5), we re-write (4.6) as

$$\begin{bmatrix} PAx + PBu - p \\ Qx - q \end{bmatrix} \leq S \begin{bmatrix} -r\mathbf{1} \\ Qx - q \end{bmatrix}. \tag{4.7}$$

Suppose that an invariant cover exists for (1) and let

$$\epsilon = \max_{i \in \mathbb{N}_{[1,n_p]}} \max_{x \in \mathbb{Q}} \min_{u \in \mathbb{U}} [PA]_i x + [PB]_i u - [p]_i.$$

Then $\epsilon \leq 0$ by Lemma 4.2. If $\epsilon < 0$, then for all $x \in \mathbb{Q}$ there necessarily exists $u \in \mathbb{U}$ so that $S = I$ and $r = -\epsilon > 0$ are feasible for (4.5), (4.7) and $S \geq 0$. For the case in which $\epsilon = 0$, let $\mathcal{I}$ be the set of indices $i \in \mathbb{N}_{[1,n_p]}$ such that

$$[PA]_i x_i^* + [PB]_i u_i^* - [p]_i = 0 \text{ for some } x_i^* \in \mathbb{Q} \text{ and } u_i^* \in \mathbb{U}.$$

Then for all $x \in \mathbb{Q}$ there exists $u \in \mathbb{U}$ so that

$$[PA]_i x + [PB]_i u - [p]_i \leq \epsilon' \text{ for all } i \in \mathbb{N}_{[1,n_p]} \setminus \mathcal{I} \text{ and for some } \epsilon' < 0.$$

Furthermore, for each $i \in \mathcal{I}$, x_i^* and u_i^* are the solutions of the LPs

$$x_i^* = \arg\max_{x \in \mathbb{Q}} [PA]_i x, \quad u_i^* = \arg\min_{u \in \mathbb{U}} [PB]_i u,$$

and it follows from LP duality that there exists an index set $\mathcal{J}_i \subseteq \mathbb{N}_{[1,n_q]}$ and scalars $\lambda_j \geq 0$ such that

$$[PA]_i = \sum_{j \in \mathcal{J}_i} \lambda_j [Q]_j$$

and $[Q]_j x_i^* - [q]_j = 0$ for all $j \in \mathcal{J}_i$. In this case therefore $S = \left[\begin{smallmatrix} S_1 & S_2 \\ 0 & I \end{smallmatrix}\right]$, where S_1 is diagonal and

$$[S_1]_{ii} = \begin{cases} 0 & i \in \mathcal{I}, \\ 1 & i \notin \mathcal{I} \end{cases} \quad [S_2]_{ij} = \begin{cases} \lambda_j & i \in \mathcal{I} \text{ and } j \in \mathcal{J}_i, \\ 0 & i \notin \mathcal{I} \end{cases}$$

satisfies $S \geq 0$ and (4.5), and moreover (4.7) holds for all $x \in \mathbb{Q}$ with $r = -\epsilon' > 0$ and some $u \in \mathbb{U}$.

To complete the proof we show that an invariant cover necessarily exists for (4.1) and $\mathbb{Q}$ if, for all $x \in \mathbb{Q}$ there exists $u \in \mathbb{U}$ such that $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \Pi(u)$ for some $r > 0$. In this case it is possible to construct a set $\{x_i^*\}_{i=1}^N$ for some finite N (where $N = O(r^{-n_x})$) that satisfies:

(i) $\cup_{i=1}^N \mathbb{B}_r(x_i^*) \supseteq \mathbb{Q}$,

(ii) for all $i \in \mathbb{N}_{[1,N]}$, $x_i^* \in \mathbb{Q}$ and $\Pi(u_i^*) \supseteq \mathbb{B}_r(x_i^*) \cap \mathbb{Q}$ for some $u_i^* \in \mathbb{U}$.

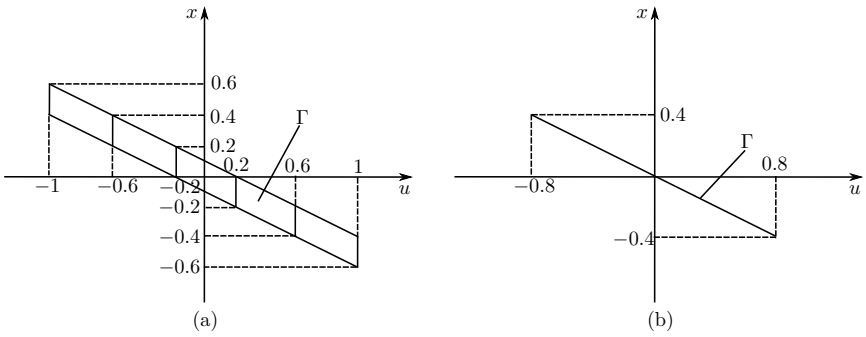


Figure 4.2: The set Γ for two different RCISs: (a) $\mathbb{Q} = [-0.6, 0.6]$; (b) $\mathbb{Q}' = [-0.4, 0.4]$.

Therefore an invariant cover exists by Lemma 4.2 and $|\mathcal{A}|^* \leq N$. □

We note that there is no obvious computationally tractable method of checking the necessary and sufficient conditions of Lemma 4.2 and Theorem 4.1.

4.3.1 Optimization-based Existence Condition

This subsection provides computationally tractable necessary and sufficient conditions for the existence of an invariant cover for a given set $\mathbb{Q}$. To avoid the computational difficulties of checking the conditions of Theorem 4.1 based on $\mathbb{B}_r(x) \cap \mathbb{Q}$, we consider instead the set $\bar{\mathcal{X}}(x, \alpha)$ defined for $x \in \mathbb{Q}, \alpha \in [0, 1]$ by

$$\bar{\mathcal{X}}(x, \alpha) = \big\{ z \in \mathbb{R}^{n_x} \mid Q\big(z - (1 - \alpha)x\big) \leq \alpha q \big\}.$$

This set can be equivalently expressed as

$$\bar{\mathcal{X}}(x, \alpha) = \{(1 - \alpha)x\} \oplus \alpha\mathbb{Q}, \tag{4.8}$$

so we therefore have $x \in \bar{\mathcal{X}}(x, \alpha) \subseteq \mathbb{Q}$, for all $x \in \mathbb{Q}$ and $\alpha \in [0, 1]$. It also follows from (4.8) that $\bar{\mathcal{X}}(x, \alpha)$ is monotonically non-decreasing with α, i.e.,

$$\bar{\mathcal{X}}(x, \alpha_1) \subseteq \bar{\mathcal{X}}(x, \alpha_2), \ \forall x \in \mathbb{Q} \text{ and } 0 \leq \alpha_1 \leq \alpha_2 \leq 1.$$

The results of this section rely on the following two lemmas, which are derived from the convexity and linearity of the conditions defining $\Pi(u)$ and $\bar{\mathcal{X}}(x, \alpha)$.

Lemma 4.3. *Let* $u = \sum_{i=1}^{N} \lambda_i u_i^*$, *where* $u_i^* \in \mathbb{U}, \Pi(u_i^*) \neq \emptyset$ *and* $\lambda_i \geq 0$ *for all* $i \in \mathbb{N}_{[1,N]}$ *with* $\sum_{i=1}^{N} \lambda_i = 1$. *Then*

$$\bigoplus_{i=1}^{N} \lambda_i \Pi(u_i^*) \subseteq \Pi(u). \tag{4.9}$$

Proof. The convexity of $\mathbb{U}$ implies that $u = \sum_{i=1}^{N} \lambda_i u_i^* \in \mathbb{U}$, while the convexity of $\mathbb{Q}$ and $\mathbb{Q} \ominus \mathbb{W}$ implies that $x \in \mathbb{Q}$ and $Ax + Bu \in \mathbb{Q} \ominus \mathbb{W}$ if $x = \sum_{i=1}^{N} \lambda_i x_i^*$, for any $\{x_i^*\}_{i=1}^{N}$ such that $x_i^* \in \Pi(u_i^*),\ \forall i \in \mathbb{N}_{[1,N]}$. We therefore have $x \in \Pi(u)$ and hence (4.9) holds. □

Lemma 4.4. *Let* $x = \sum_{i=1}^{N} \lambda_i x_i^*$, *where* $x_i^* \in \mathbb{Q}$ *and* $\lambda_i \geq 0$ *for all* $i \in \mathbb{N}_{[1,N]}$ *with* $\sum_{i=1}^{N} \lambda_i = 1$, *and let* $\alpha_i \in [0,1]$ *for all* $i \in \mathbb{N}_{[1,N]}$. *Then*

$$\bar{\mathcal{X}}\big(x, \min_{i \in \mathbb{N}_{[1,N]}} \alpha_i\big) \subseteq \bigoplus_{i=1}^{N} \lambda_i \bar{\mathcal{X}}(x_i^*, \alpha_i) \subseteq \bar{\mathcal{X}}\big(x, \max_{i \in \mathbb{N}_{[1,N]}} \alpha_i\big). \tag{4.10}$$

Proof. Given the assumptions on x_i^*, λ_i and α_i, we have

$$\bar{\mathcal{X}}(x_i^*, \alpha_i) \supseteq \bar{\mathcal{X}}(x_i^*, \min_{i \in \mathbb{N}_{[1,N]}} \alpha_i),\ \forall i \in \mathbb{N}_{[1,N]},$$

and hence

$$\begin{aligned}
\bigoplus_{i=1}^{N} \lambda_i \bar{\mathcal{X}}(x_i^*, \alpha_i) &\supseteq \bigoplus_{i=1}^{N} \lambda_i \bar{\mathcal{X}}\big(x_i^*, \min_{i \in \mathbb{N}_{[1,N]}} \alpha_i\big) \\
&= \{(1 - \min_{i \in \mathbb{N}_{[1,N]}} \alpha_i)x\} \oplus \min_{i \in \mathbb{N}_{[1,N]}} \alpha_i\, \mathbb{Q} \\
&= \bar{\mathcal{X}}(x, \min_{i \in \mathbb{N}_{[1,N]}} \alpha_i).
\end{aligned}$$

This proves the first subset relation in (4.10); the second can be proved using a similar argument. □

We define the critical vertices of Γ as follows.

Definition 4.2. *A vertex* (x, u) *of* Γ *is said to be a* critical vertex *of* Γ *if*

(i) $x \in \mathrm{vert}(\mathbb{Q})$,

(ii) $\bar{\mathcal{X}}(x,\alpha) \subseteq \Pi(u)$ *for some* $\alpha \in (0,1]$.

The main result of this section (Theorem 4.2) states that an invariant cover $(\mathcal{A}, G)$ exists for the system (4.1) and the set $\mathbb{Q}$ if and only if every $x \in \operatorname{vert}(\mathbb{Q})$ corresponds to a critical vertex of Γ. We prove this using Lemmas 4.3, 4.4, and the properties of critical vertices to define for each $x \in \mathbb{Q}$ a control u such that $\bar{\mathcal{X}}(x,\alpha) \subseteq \Pi(u)$ for some $\alpha > 0$. However, as x approaches the boundary of $\mathbb{Q}$, x also approaches the boundary of $\bar{\mathcal{X}}(x,\alpha)$. To ensure the existence of $r > 0$ such that $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \Pi(u)$ for all $x \in \mathbb{Q}$ and thus fulfil the conditions of Theorem 4.1, we therefore consider the set $\bar{\mathcal{X}}(y_\sigma(x),\alpha)$. For $\sigma \in (0,1)$ and $x \in \mathbb{Q} \setminus (1-\sigma)\mathbb{Q}$, $y_\sigma(x)$ is defined as a point in the boundary of $\mathbb{Q}$ given by the solution of an LP

$$\begin{aligned} y_\sigma(x) = \arg \max_{y \in \mathbb{Q}} \min_{i \notin J_\sigma(x)} \sigma[q]_i - [Q]_i\big(x - (1-\sigma)y\big) \\ \text{s.t.} \qquad [Q]_j y = [q]_j,\ \forall j \in J_\sigma(x) \end{aligned} \tag{4.11}$$

with

$$J_\sigma(x) = \{j \in \mathbb{N}_{[1,n_q]} \mid [Q]_j x > (1-\sigma)[q]_j\}.$$

For $x \in (1-\sigma)\mathbb{Q}$, we define $y_\sigma(x)$ by

$$y_\sigma(x) = x. \tag{4.12}$$

An upper limit on σ is provided by the following result.

Lemma 4.5. *Let* $\sigma \in (0, \bar{\sigma}]$, *where*

$$\bar{\sigma} = \frac{\min_{k \in \mathbb{N}_{[1,n_q]}} \min_{x \in \operatorname{vert}_k(\mathbb{Q})} [q]_k - [Q]_k x}{\max_{k \in \mathbb{N}_{[1,n_q]}} \max_{x \in \mathbb{Q}} [q]_k - [Q]_k x},$$

then

(i) $y_\sigma(x)$ *exists for all* $x \in \mathbb{Q}$;

(ii) $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}(y_\sigma(x),\sigma)$ *if* $r = \sigma^2 \min_{j \in \mathbb{N}_{[1,n_q]}} [q]_j$.

Proof. To prove the assertion in (i) we show by contradiction that (4.11) has a solution for all $x \in \mathbb{Q} \setminus (1-\sigma)\mathbb{Q}$ if $\sigma \leq \bar{\sigma}$. For $\sigma \in (0,1)$ and $x \in \mathbb{Q} \setminus (1-\sigma)\mathbb{Q}$, let

$$J_\sigma(x) = \{j \in \mathbb{N}_{[1,n_q]} : [Q]_j x > (1-\sigma)[q]_j\},$$

and define σ_j for each $j \in \mathbb{N}_{[1,n_q]}$

$$\sigma_j = \begin{cases} ([q]_j - [Q]_j x)/[q]_j & \text{if } j \in J_\sigma(x) \\ \sigma & \text{otherwise} \end{cases}$$

so that $\sigma_j \in [0, \sigma), \forall j \in J_\sigma(x)$. Also define $\mathbb{Q}_\sigma(x) \subseteq \sigma\mathbb{Q}$ as the set

$$\mathbb{Q}_\sigma(x) = \left\{x \in \mathbb{R}^{n_x} \mid [Q]_j x \leq \sigma_j [q]_j,\ \forall j \in \mathbb{N}_{[1,n_q]}\right\},$$

so that for each $j \in J_\sigma(x)$ we have $[Q]_j y = [q]_j$ where $y = x + z \in \mathbb{Q}$ and $z \in \operatorname{vert}(\mathbb{Q}_\sigma(x))$.

Suppose that (i) is false and (4.11) is primal infeasible. Then there exists a pair of indices $j_1, j_2 \in J_\sigma(x)$ such that the hyperplanes $\{y \mid [Q]_{j_1} y = [q]_{j_1}\}$ and $\{y \mid [Q]_{j_2} y = [q]_{j_2}\}$ have no point of intersection in $\mathbb{Q}$. But $[Q]_{j_i} y_i = [q]_{j_i}$ where $y_i = x + z_i$, $z_i \in \operatorname{vert}(\mathbb{Q}_\sigma(x))$, $i = 1, 2$, and hence

$$\begin{aligned} [q]_{j_2} - [Q]_{j_2} x &= [q]_{j_2} - [Q]_{j_2}(y_1 - z_1) \\ &\geq \min_{x \in \operatorname{vert}_{j_2}(\mathbb{Q})} \left\{[q]_{j_2} - [Q]_{j_2} x\right\} + \min_{z \in \mathbb{Q}_\sigma(x)} [Q]_{j_2} z \\ &\geq \min_k \min_{x \in \operatorname{vert}_k(\mathbb{Q})} \left\{[q]_{j_2} - [Q]_{j_2} x\right\} + \sigma \min_{x \in \mathbb{Q}} [Q]_{j_2} x, \end{aligned}$$

where the first inequality follows from $[Q]_{j_2} y_1 < [q]_{j_2}$ and $z_1 \in \mathbb{Q}_\sigma(x)$, and the second inequality from $\mathbb{Q}_\sigma(x) \subseteq \sigma\mathbb{Q}$. But $[q]_{j_2} - [Q]_{j_2} x = \sigma_2 [q]_{j_2} < \sigma [q]_{j_2}$, which implies

$$\sigma [q]_{j_2} > \min_k \min_{x \in \operatorname{vert}_k(\mathbb{Q})} [q]_{j_2} - [Q]_{j_2} x + \sigma \min_{x \in \mathbb{Q}} [Q]_{j_2} x,$$

and hence σ must be greater than $\bar{\sigma}$. Therefore, if $\sigma \leq \bar{\sigma}$, then $y \in \mathbb{Q}$ and $z \in \operatorname{vert}(\mathbb{Q}_\sigma(x))$ must exist such that $y = x + z$ and $[Q]_j y = [q]_j\ \forall j \in J_\sigma(x)$, and it follows that (4.11) has a solution for all $x \in \mathbb{Q} \setminus (1 - \sigma)\mathbb{Q}$.

To prove the assertion in (ii) we show that $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}(y_\sigma(x), \sigma)$ with $r = \sigma^2 \min_j [q]_j$. First consider the case in which $x \in (1 - \sigma)\mathbb{Q}$. Then $y_\sigma(x) = x$ so $\bar{\mathcal{X}}(y_\sigma(x), \sigma) = \bar{\mathcal{X}}(x, \sigma)$ and $\mathbb{B}_r(x) \subseteq \bar{\mathcal{X}}(x, \sigma)$ if and only if

$$\{z \mid Q(z - x) \leq r\mathbf{1}\} \subseteq \{z \mid Q(z - x) \leq \sigma(q - Qx)\}.$$

The condition holds whenever $r \leq \min_{j \in \mathbb{N}_{[1,n_q]}} \sigma([q]_j - [Q]_j x)$ but $x \in (1 - \sigma)\mathbb{Q}$ implies $q - Qx \geq \sigma q$, and it follows that $\mathbb{B}_r(x) \subseteq \bar{\mathcal{X}}(x, \sigma)$ if $r = \sigma^2 \min_{j \in \mathbb{N}_{[1,n_q]}} [q]_j$.

Next we determine r so that $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}(y_\sigma(x), \sigma)$ for the case that $x \in \mathbb{Q} \setminus (1-\sigma)\mathbb{Q}$. The definition of $y_\sigma(x)$ implies $[Q]_j y_\sigma(x) = [q]_j$ for all $j \in J_\sigma(x)$, and since

$$\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \big\{ z \mid Q(z - y_\sigma(x)) \leq \min\{r\mathbf{1} + Q(x - y_\sigma(x)), q - Qy_\sigma(x)\} \big\},$$

we have $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}(y_\sigma(x), \sigma)$ if

$$r + [Q]_j(x - y_\sigma(x)) \leq \sigma([q]_j - [Q]_j y_\sigma(x)) \text{ for all } j \notin J_\sigma(x),$$

or equivalently if

$$r \leq \min_{j \notin J_\sigma(x)} \sigma[q]_j - [Q]_j\big(x - (1-\sigma)y_\sigma(x)\big).$$

But (4.11) selects $y_\sigma(x)$ so that the right hand side of this expression is minimized for some $x \in \operatorname{vert}(\mathbb{Q})$ (since this implies that $y_\sigma(x) = x \in \operatorname{vert}(\mathbb{Q})$), and we therefore have

$$\begin{aligned} \min_{j \notin J_\sigma(x)} \sigma[q]_j - [Q]_j\big(x - (1-\sigma)y_\sigma(x)\big) &\geq \sigma \min_{j \notin J_\sigma(x)} [q]_j - [Q]_j x \\ &\geq \sigma^2 \min_k [q]_k, \end{aligned}$$

where the first inequality is obtained by setting $y_\sigma(x) = x$ and the second follows from $[Q]_j x \leq (1-\sigma)[q]_j$ for all $j \notin J_\sigma(x)$. Therefore $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}(y_\sigma(x), \sigma)$ if $r = \sigma^2 \min_{j \in \mathbb{N}_{[1,n_q]}} [q]_j$. □

Theorem 4.2. *There exists an invariant cover $(\mathcal{A}, G)$ of the system (4.1) and the set $\mathbb{Q}$ if and only if each vertex x of $\mathbb{Q}$ corresponds to a critical vertex (x, u) of Γ.*

Proof. Every vertex of $\mathbb{Q}$ corresponds to a critical vertex of Γ if and only if for each $x \in \operatorname{vert}(\mathbb{Q})$ there exists $\alpha \in (0, 1]$ and $u \in \mathbb{U}$ such that $\bar{\mathcal{X}}(x, \alpha) \subseteq \Pi(u)$. From (4.8) it is obvious that $\alpha > 0$ is necessary for $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}(x, \alpha)$ for some $r > 0$.

Therefore $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}(x, \alpha) \subseteq \Pi(u)$ for some $r > 0$ and $u \in \mathbb{U}$ only if every vertex of $\mathbb{Q}$ corresponds to a critical vertex of Γ, and it follows from Theorem 4.1 that this is a necessary condition for existence of an invariant cover.

To prove sufficiency, note that for all $x \in \mathbb{Q}$, $y_\sigma(x)$ can be expressed $y_\sigma(x) = \sum_{i=1}^N \lambda_i x_i^*$, with $\{x_i^*\}_{i=1}^N = \operatorname{vert}(\mathbb{Q})$, $\lambda_i \geq 0$ for all $i \in \mathbb{N}_{[1,N]}$ and

$\sum_{i=1}^{N} \lambda_i = 1$. If every vertex of $\mathbb{Q}$ corresponds to a critical vertex of Γ, then $\alpha_i \in (0, 1]$ and $u_i^* \in \mathbb{U}$ exist for all $i \in \mathbb{N}_{[1,N]}$ so that $\bar{\mathcal{X}}(x_i^*, \alpha_i) \subseteq \Pi(u_i^*)$. Using Lemmas 4.3 and 4.4 we therefore obtain

$$\bar{\mathcal{X}}\big(y_\sigma(x), \min_{i \in \mathbb{N}_{[1,N]}} \alpha_i\big) \subseteq \bigoplus_{i=1}^{N} \lambda_i \bar{\mathcal{X}}(x_i^*, \alpha_i) \subseteq \bigoplus_{i=1}^{N} \lambda_i \Pi(u_i^*) \subseteq \Pi(u)$$

where $u = \sum_{i=1}^{N} \lambda_i u_i^* \in \mathbb{U}$. Furthermore, Lemma 4.5 implies that $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}(y_\sigma(x), \alpha)$ with $r = \sigma^2 \min_k [q]_k$ if $\sigma = \min\{\alpha, \bar{\sigma}\}$. Therefore $\min_{i \in \mathbb{N}_{[1,N]}} \alpha_i > 0$ ensures that $r > 0$ and hence an invariant cover must exist by Theorem 4.1. □

Theorem 4.2 shows that it can be determined whether or not an invariant cover exists by checking if each vertex of $\mathbb{Q}$ has a corresponding critical vertex of Γ. This is the basis of the computational procedure described in Section 4.5.

Example 4.2. *The set Γ is shown in Figure 4.2 (a) for the system in Example 4.1 with $\mathbb{Q} = [-0.6, 0.6]$ being an RCIS. It can be seen that the vertices $x = -0.6$ and $x = 0.6$ of $\mathbb{Q}$ correspond respectively to critical vertices $(-0.6, 1)$ and $(0.6, -1)$ of Γ. In particular, it is easy to verify that $\alpha = 1/6$ gives $\bar{\mathcal{X}}(-0.6, 1/6) = \Pi(1) = [-0.6, -0.4]$ and $\bar{\mathcal{X}}(0.6, 1/6) = \Pi(-1) = [0.4, 0.6]$. Theorem 4.2 therefore implies that an invariant cover of $\mathbb{Q}$ exists, which was given in Example 4.1 and is shown in Figure 4.2 (a).*

Consider the set $\mathbb{Q}' = [-0.4, 0.4]$, which is also an RCIS. The corresponding set Γ for $\mathbb{Q}'$ is shown in Figure 4.2 (b). In this case the set Γ is a line segment. Therefore the vertices of $\mathbb{Q}'$ have no corresponding critical vertex of Γ. In particular, there is no finite set of control inputs such that (4.4) holds and thus there does not exist an invariant cover for this scalar system and the set $\mathbb{Q}'$.

It can be determined whether or not a given vertex of $\mathbb{Q}$ corresponds to a critical vertex of Γ by solving an LP, as we show next. Given the vertices of $\mathbb{Q}$, this suggests a computationally tractable method for checking whether or not an invariant cover exists: solve the LP for each vertex $x \in \mathrm{vert}(\mathbb{Q})$.

Lemma 4.6. *Let $\alpha^*(x)$ be the optimal value of the LP*

$$\begin{aligned} \alpha^*(x) = \max_{\alpha \in [0,1],\, S \geq 0,\, u} \quad & \alpha \\ \text{s.t.} \quad & (1-\alpha)PAx + Sq + PBu \leq p \\ & F_u u \leq f_u \\ & SQ = \alpha PA \end{aligned} \tag{4.13}$$

and $u^*(x)$ be the solution set for u. Then (x, u) is a critical vertex of Γ if and only if $\alpha^*(x) > 0$, $x \in \mathrm{vert}(\mathbb{Q})$, and $u \in u^*(x)$.

Proof. For given $x \in \mathbb{Q}$, (4.13) determines the maximum value of $\alpha \in [0, 1]$ such that $\bar{\mathcal{X}}(x, \alpha) \subseteq \Pi(u)$ for some $u \in \mathbb{U}$. Specifically, since $\bar{\mathcal{X}}(x, \alpha) \subseteq \mathbb{Q}$ for all $x \in \mathbb{Q}$ and $\alpha \in [0, 1]$, we have $\bar{\mathcal{X}}(x, \alpha) \subseteq \Pi(u)$ if and only if there exists $u \in \mathbb{U}$ such that $A\bar{\mathcal{X}}(x, \alpha) \oplus \{Bu\} \subseteq \mathbb{Q} \ominus \mathbb{W}$. By LP duality, this set inclusion condition holds if and only if a non-negative matrix R exists satisfying

$$\begin{cases} RQ = PA \\ (1-\alpha)PAx + \alpha Rq + PBu \leq p. \end{cases}$$

The variable transformation $S = \alpha R$ results in a set of constraints that are linear in u, α and S. The problem of maximizing α subject to $\bar{\mathcal{X}}(x, \alpha) \subseteq \Pi(u)$ and $u \in \mathbb{U}$ can therefore be expressed as the LP (4.13). □

4.4 Cardinality Bounds

This section develops an approach based on the existence condition of Theorem 4.2 to address Problem 4.2. We derive upper and lower bounds on $|\mathcal{A}|^*$, the minimal cardinality of an invariant cover $(\mathcal{A}, G)$ of the system (4.1) and the set $\mathbb{Q}$. Define

$$v^* = \max_{u \in \Phi} \mathrm{vol}\big(\Pi(u)\big),$$

where $\Phi = \{u \in \mathbb{U} \mid \exists x,\ (x, u) \in \Gamma\}$ and $\mathrm{vol}(\cdot)$ denotes the volume. Let $\underline{\alpha}^*$ and $\bar{\alpha}^*$ denote the optimal values

$$\begin{aligned} \underline{\alpha}^* = \min_{x \in \mathrm{vert}(\mathbb{Q})} \ \max_{\alpha \in [0,1],\, S \geq 0,\, u} \ & \alpha \\ \text{s.t.} \quad & (1-\alpha)PAx + Sq + PBu \leq p \\ & F_u u \leq f_u \\ & SQ = \alpha PA \end{aligned} \tag{4.14}$$

and

$$\begin{aligned} \bar{\alpha}^* = \max_{\alpha \in [0,1], S \geq 0, y, u} \ & \alpha \\ \text{s.t.} \quad & Qy \leq (1-\alpha)q \\ & PAy + Sq + PBu \leq p \\ & F_u u \leq f_u \\ & SQ = \alpha PA. \end{aligned} \tag{4.15}$$

Remark 4.2. *In (4.15) $\bar{\alpha}^*$ is the solution of a single LP, whereas $\underline{\alpha}^*$ in (4.14) is computed by solving one LP for each vertex of $\mathbb{Q}$. In particular,*

$$\underline{\alpha}^* = \min_{x \in \mathrm{vert}(\mathbb{Q})} \alpha^*(x),$$

where $\alpha^*(x)$ is given by (4.13).

Before giving the bounds on $|\mathcal{A}|^*$, we need the following lemma.

Lemma 4.7. *Let $\delta \subseteq \mathbb{Q}$ and $\sigma = \min\{\underline{\alpha}^*, \bar{\sigma}\}$. Then,*

(i) $x \oplus \sigma\delta \subseteq \bar{\mathcal{X}}\big((1-\sigma)^{-1}x, \sigma\big)$ *for* $x \in (1-\sigma)\mathbb{Q}$;

(ii) $\big(\{x\} \oplus \sigma^2\delta\big) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}(y_\sigma(x), \sigma)$ *for* $x \in \mathbb{Q} \setminus (1-\sigma)\mathbb{Q}$,

where $y_\sigma(x)$ is defined in (4.11).

Proof. The assertion in (i) directly follows from the definition of the set $\bar{\mathcal{X}}(x, \sigma)$. For given $x \in (1-\sigma)\mathbb{Q}$, $x \oplus \sigma\delta \subseteq x \oplus \sigma\mathbb{Q} = \bar{\mathcal{X}}\big((1-\sigma)^{-1}x, \sigma\big)$.

The assertion in (ii) can be demonstrated by showing that $\big(\{x\} \oplus \sigma^2\mathbb{Q}\big) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}\big(y_\sigma(x), \sigma\big)$ using an argument similar to the proof of Lemma 4.5, assertion (ii). For $\hat{\sigma} > 0$ we have

$$\big(\{x\} \oplus \hat{\sigma}\mathbb{Q}\big) \cap \mathbb{Q} = \Big\{z \mid [Q]_j z \le \min\{[q]_j, \hat{\sigma}[q]_j + [Q]_j x\}, \forall j \in \mathbb{N}_{[1,n_q]}\Big\}.$$

Therefore

$$\big(\{x\} \oplus \hat{\sigma}\mathbb{Q}\big) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}\big(y_\sigma(x), \sigma\big) = \big\{z \mid Qz \le \sigma q + (1-\sigma)Qy_\sigma(x)\big\}$$

if and only if for all $j \in \mathbb{N}_{[1,n_q]}$,

$$\sigma[q]_j + (1-\sigma)[Q]_j y_\sigma(x) \ge \min\{[q]_j, \hat{\sigma}[q]_j + [Q]_j x\}.$$

Recall that $J_\sigma(x) = \{j \in \mathbb{N}_{[1,n_q]} \mid [Q]_j x > (1-\sigma)[q]_j\}$ and $[Q]_j y_\sigma(x) = [q]_j$ for $j \in J_\sigma(x)$, from which it follows that for all $j \in J_\sigma(x)$,

$$\sigma[q]_j + (1-\sigma)[Q]_j y_\sigma(x) = [q]_j \ge \min\{[q]_j, \hat{\sigma}[q]_j + [Q]_j x\}.$$

For $j \notin J_\sigma(x)$, $\sigma[q]_j - [Q]_j\big(x - (1-\sigma)y_\sigma(x)\big)$ is minimized over $x \in \mathbb{Q} \setminus (1-\sigma)\mathbb{Q}$ when $x = y_\sigma(x)$, and we therefore have

$$\sigma[q]_j - [Q]_j\big(x - (1-\sigma)y_\sigma(x)\big) = \sigma([q]_j - [Q]_j x) \ge \sigma^2[q]_j$$

and

$$\begin{aligned}\sigma[q]_j + (1-\sigma)[Q]_j y_\sigma(x) &\geq \sigma^2[q]_j + [Q]_j x \\ &\geq \min\{[q]_j, \sigma^2[q]_j + [Q]_j x\}, \ \ \forall j \notin J_\sigma(x).\end{aligned}$$

It follows that $\big(\{x\}\oplus\hat{\sigma}\delta\big)\cap\mathbb{Q} \subseteq \big(\{x\}\oplus\hat{\sigma}\mathbb{Q}\big)\cap\mathbb{Q} \subseteq \bar{\mathcal{X}}\big(y_\sigma(x),\sigma\big)$ if $\hat{\sigma}=\sigma^2$. □

Theorem 4.3. *Let $\delta = [\underline{d}_1, \overline{d}_1] \times \cdots \times [\underline{d}_{n_x}, \overline{d}_{n_x}]$ and $\Delta = [\underline{D}_1, \overline{D}_1] \times \cdots \times [\underline{D}_{n_x}, \overline{D}_{n_x}]$ be inner- and outer-bounding hyperrectangles such that $\delta \subseteq \mathbb{Q} \subseteq \Delta$, and let $\sigma = \min\{\underline{\alpha}^*, \bar{\sigma}\}$. Then $|\mathcal{A}|^*$ satisfies*

$$(a).\ \left\lceil \frac{\mathrm{vol}(\mathbb{Q})}{v^*} \right\rceil \leq |\mathcal{A}|^* \leq \prod_{i=1}^{n_x} \left\lceil \frac{\overline{D}_i - \underline{D}_i}{(\overline{d}_i - \underline{d}_i)\sigma^2} \right\rceil \tag{4.16}$$

$$(b).\ |\mathcal{A}|^* = 1 \ \textit{if and only if}\ \bar{\alpha}^* = 1.$$

Proof. We first consider the statement in (a). The lower bound on $|\mathcal{A}|^*$ in (4.16) follows directly from the volumetric scaling of the maximal set $\Pi(u)$ relative to $\mathbb{Q}$ and the definition of v^* as $\max_{u\in\Phi} \mathrm{vol}(\Pi(u))$. To prove the upper bound on $|\mathcal{A}|^*$ in (4.16), we note that by the proof of Theorem 4.2 and Lemma 4.7, for all $x \in \mathbb{Q}$, $\big(\{x\} \oplus \sigma^2\delta\big) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}(y_\sigma(x),\sigma) \subseteq \Pi(u)$ for some $u \in \mathbb{U}$, where $y_\sigma(x)$ is defined in (4.11)–(4.12). The upper bound on $|\mathcal{A}^*|$ in (4.16) then follows from an upper bound on the cardinality of a cover of $\mathbb{Q}$ of the form $\cup_{i=1}^N \big(\{x_i\}\oplus\sigma^2\delta\big)$, which can be obtained from the ratios of the corresponding sides of the hyperrectangles Δ (which contains $\mathbb{Q}$) and $\sigma^2\delta$.

The upper bound on $|\mathcal{A}|^*$ in (4.16) then follows from the ratios of the corresponding sides of the hyperrectangles Δ (which contains $\mathbb{Q}$) and $\{(1-\underline{\alpha}^*)x\}\oplus \underline{\alpha}^*\delta$ (which is contained in $\bar{\mathcal{X}}(x,\underline{\alpha}^*)$).

To prove the statement in (b), we note that the value of $\bar{\alpha}^*$ in (4.15) is the maximum, as x varies over $\mathbb{Q}$, of $\alpha \in [0,1]$ such that $\bar{\mathcal{X}}(x,\alpha) \subseteq \Pi(u)$ for some $u \in \mathbb{U}$. This follows from Lemma 4.6, which implies that $\bar{\alpha}^* = \max_{x\in\mathbb{Q}} \alpha^*(x)$. Furthermore from (4.8) we have $\bar{\mathcal{X}}(x,\alpha) = \mathbb{Q}$ if and only if $\alpha = 1$, and it follows that $\mathcal{A} = \mathbb{Q}$ (and hence $|\mathcal{A}|^* = 1$) if and only if $\bar{\alpha}^* = 1$. In this case $G(\mathbb{Q})$ is simply equal to u^*, the optimal value of u in (4.14). □

Remark 4.3. *We note that [34] only provides a lower bound on the invariance feedback entropy and the minimal data-rate necessary for invariance. As a complement, the upper bound in (4.16) on the minimal cardinality of an invariant cover also provides a bound on these quantities in the light of Lemma 3 of [34].*

Note that this upper bound is likely to be loose due to the appearance of σ^2 in the denominator, and this is observed in numerical examples. In addition, the lower bound in (4.16) is valid for the minimal data-rate achieved over all admissible static coder-controllers, but differs from its lower bound in Theorem 8 of [34].

The lower bound on $|\mathcal{A}|^*$ in (4.16) can be computed in principle by determining $\mathrm{vol}(\Pi(u))$ over a finite set of points. Specifically, the vertices of $\Pi(u)$ are obtained (as functions of $u \in \Phi$) as the solutions of a set of right-hand-side multiparametric linear programs, and they are therefore piecewise affine functions of u with the pieces defined by a polyhedral complex, $\mathcal{K}$, of subsets of Φ [129]. As a result, the volume of $\Pi(u)$ can be expressed (by triangulating $\Pi(u)$ into a collection of simplexes [130]) as a sum of non-negative determinants of matrices whose elements are piecewise affine functions of u. It follows that $\mathrm{vol}(\Pi(u))$ is piecewise quasi-convex in u and the maximum, v^*, over $u \in \Phi$, is therefore achieved at a vertex of the complex $\mathcal{K}$.

Determining the volume of an arbitrary polytopic set is computationally hard, see, [131]. Since $\bar{\alpha}^*$ in (4.15) is the maximum value of α such that $\bar{\mathcal{X}}(x, \alpha) \subseteq \Pi(u)$ for some $x \in \mathbb{Q}$ and $u \in \mathbb{U}$, a convenient approximation is

$$v^* \approx (\bar{\alpha}^*)^{n_x} \mathrm{vol}(\mathbb{Q}).$$

This implies that the lower bound on $|\mathcal{A}|^*$ in (4.16) is approximately equal to $(1/\bar{\alpha}^*)^{n_x}$. We note however that $(1/\bar{\alpha}^*)^{n_x}$ does not necessarily lower bound $|\mathcal{A}|^*$ since $v^* \geq (\bar{\alpha}^*)^{n_x} \mathrm{vol}(\mathbb{Q})$.

4.4.1 Tightness of Cardinality Bounds

This subsection shows that the bounds on the minimal cardinality are tight for scalar systems under some mild conditions. Consider a scalar system:

$$x_{k+1} = ax_k + u_k + w_k,$$

with $\mathbb{X} = \mathbb{R}$, $\mathbb{U} = [\underline{u}, \bar{u}]$, and $\mathbb{W} = [\underline{w}, \bar{w}]$. Let $\mathbb{Q} = [\underline{q}, \bar{q}]$. Assume that $\underline{q} < \underline{w} < \bar{w} < \bar{q}$. Then, we have that $\mathrm{vert}(\mathbb{Q}) = \{\underline{q}, \bar{q}\}$ and $\mathbb{Q} \ominus \mathbb{W} = [\underline{q} - \underline{w}, \bar{q} - \bar{w}]$. Recall $\underline{\alpha}^*$ and $\bar{\alpha}^*$ defined in Eqs. (4.14)–(4.15). Since $v^* = (\bar{\alpha}^*)^{n_x} \mathrm{vol}(\mathbb{Q})$, one can choose $\delta = \mathbb{Q} = \Delta$, and $\sigma = \underline{\alpha}^*$ for $n_x = 1$, the bounds in (4.16) become

$$\lceil \frac{1}{\bar{\alpha}^*} \rceil \leq |\mathcal{A}|^* \leq \lceil \frac{1}{\underline{\alpha}^*} \rceil.$$

Denote by $\alpha^*(\underline{q})$ and $\alpha^*(\bar{q})$ the optimal value of LP (4.13) for the vertices of $\mathbb{Q}$, respectively. Since the optimal value occurs at a vertex of the feasible region for the LP, we have $\underline{\alpha}^* \in \{\alpha^*(\underline{q}), \alpha^*(\bar{q})\}$ and $\bar{\alpha}^* \in \{\alpha^*(\underline{q}), \alpha^*(\bar{q})\}$. Therefore the bound in (4.16) is tight if and only if $\alpha^*(\underline{q}) = \alpha^*(\bar{q})$.

In the case of a symmetric scalar system, we set $0 < -\underline{u} = \bar{u}$, $0 < -\underline{w} = \bar{w}$, and $0 < -\underline{q} = \bar{q}$. Without loss of generality, we assume that $a > 0$. In this case, $\alpha^*(\underline{q}) = \alpha^*(\bar{q})$ holds. The explicit minimal cardinality of an invariant cover is then

$$|\mathcal{A}|^* = \begin{cases} \lceil \frac{a\bar{q}}{\bar{q}-\bar{w}} \rceil & \text{if } (a-1)\bar{q} + \bar{w} + \bar{u} \geq 0 \\ +\infty & \text{otherwise.} \end{cases}$$

This can be validated by Example 4.2. Note that, in contrast to [34], the control limits are taken into account in this analysis.

4.5 Computational Algorithm

This section uses the existence conditions discussed in Section 4.3 to solve Problem 4.3. Algorithm 4.1 describes a procedure for computing an invariant cover of system (4.1) and a given polytopic set $\mathbb{Q}$. The algorithm makes use of the following result, which is proved in [132, Theorem 3].

Lemma 4.8. *Let $\bar{\mathbb{Y}}$ be a polytope and let $\mathbb{Y}_0 = \{z \in \bar{\mathbb{Y}} \mid Yz \leq y\}$ be a nonempty polytopic subset of $\bar{\mathbb{Y}}$, where $y \in \mathbb{R}^{n_y}$. For each $i \in \mathbb{N}_{[1,n_y]}$, define $\mathbb{Y}_i = \{z \in \bar{\mathbb{Y}} \mid [Y]_i z \geq [y]_i,\ [Y]_j z \leq [y]_j\ \forall j < i\}$. Then, $\{\mathbb{Y}_i\}_{i=1}^{n_y}$ is a partition of $\bar{\mathbb{Y}}$ with respect to $\mathbb{Y}_0$, i.e.,*

(i) $\bigcup_{i=0}^{n_y} \mathbb{Y}_i = \bar{\mathbb{Y}}$,

(ii) $\operatorname{int}(Y_i) \cap \operatorname{int}(Y_0) = \emptyset, \forall i$,

(iii) $\operatorname{int}(Y_i) \cap \operatorname{int}(Y_j) = \emptyset, \forall i \neq j$.

Algorithm 4.1 first uses the condition in Theorem 4.2 to check the existence of an invariant cover (lines 1–9). If no invariant cover exists, the algorithm stops and returns Null. Otherwise, the algorithm continues by repeatedly partitioning subsets of $\mathbb{Q}$ using Lemma 4.8, with the initial subset $\mathbb{Y}_0$ defined in terms of a set $\Pi(u)$ which is constructed using a convex combination of the vertices of $\mathbb{Q}$. The procedure PARTITION($\bar{\mathbb{Y}}$) (lines 13-23) constructs

Algorithm 4.1 Invariant Cover Computation

Require: System (4.1), and sets $\mathbb{Q}$, $\mathbb{U}$, and $\mathbb{Q} \ominus \mathbb{W}$.
Ensure: An invariant cover $(\mathcal{A}, G)$ for (4.1) and $\mathbb{Q}$.

$\{x_i^v\}_{i=1}^N \leftarrow \mathrm{vert}(\mathbb{Q})$;
for all $i \in \mathbb{N}_{[1,N]}$ **do**
    Solve (4.13) for $\alpha^*(x_i^v)$;
    $u_i^v \leftarrow u^*(x_i^v)$;
end for
$\underline{\alpha}^* \leftarrow \min_i \alpha^*(x_i^v)$;
if $\underline{\alpha}^* = 0$ **then**
    Stop and return Null;
end if
$\sigma \leftarrow \min\{\underline{\alpha}^*, \bar{\sigma}\}$
$\mathcal{A} \leftarrow \emptyset$;
Execute PARTITION($\mathbb{Q}$);
procedure PARTITION($\bar{\mathbb{Y}}$)
    Compute $x \in \mathrm{vert}(\bar{\mathbb{Y}})$;
    Compute $y_\sigma(x)$ using (4.11);
    Compute $\{\lambda_i\}_{i=1}^N$ satisfying $y_\sigma(x) = \sum_{i=1}^N \lambda_i x_i^v$, $\sum_{i=1}^N \lambda_i = 1$ and $\lambda_i \geq 0,\ \forall i \in \mathbb{N}_{[1,N]}$;
    $u \leftarrow \sum_{i=1}^N \lambda_i u_i^v$;
    $\mathcal{A} \leftarrow \{\mathcal{A}, \Pi(u)\}$ and $G(\Pi(u)) \leftarrow u$;
    $\mathbb{Y}_0 \leftarrow \bar{\mathbb{Y}} \cap \Pi(u)$;
    Partition $\bar{\mathbb{Y}} \setminus \mathbb{Y}_0$ into $\{\mathbb{Y}_i\}_{i=1}^{n_y}$ using Lemma 4.8;
    for each nonempty sub-region $\mathbb{Y}_i$ **do**
        Execute PARTITION($\mathbb{Y}_i$);
    end for
end procedure
Return: $(\mathcal{A}, G)$.

a cover of the set $\bar{\mathbb{Y}}$ and stores this cover and the corresponding control inputs in $(\mathcal{A}, G)$. This process continues until the elements of $\mathcal{A}$ cover the entire set $\mathbb{Q}$.

A vertex of $\bar{\mathbb{Y}}$ in line 14 can be found by checking whether any vertex of $\mathbb{Q}$ lies in $\bar{\mathbb{Y}}$ and then solving a LP in n_x variables if this check fails. The set $\{\lambda_i\}_{i=1}^N$ in line 16 can be defined uniquely as the minimizing argument of

$$\begin{aligned} \min_{\lambda_1,\dots,\lambda_N} \ & \| \sum_{i=1}^{N} \lambda_i u_i^v \|_\infty \\ \text{s.t. } & y = \sum_{i=1}^{N} \lambda_i x_i^v, \ \sum_{i=1}^{N} \lambda_i = 1, \ \lambda_i \geq 0, \forall i \in \mathbb{N}_{[1,N]} \end{aligned}$$

which requires the solution of an LP in N variables. The main computational effort of Algorithm 4.1 is spent on solving the LPs in lines 14–16.

Theorem 4.4. *Algorithm 4.1 finds an invariant cover $(\mathcal{A}, G)$ of the system (4.1) and the set $\mathbb{Q}$ in finite time, if it exists.*

Proof. We demonstrate that Algorithm 4.1 terminates in finite time whenever an invariant cover exists for the system (4.1) and set $\mathbb{Q}$ by showing that PARTITION$(\bar{\mathbb{Y}})$ in line 22 can only be invoked a finite number of times if $\underline{\alpha}^* > 0$. Since σ is defined in line 10 as $\min\{\bar{\sigma}, \underline{\alpha}^*\}$, Lemma 4.5 implies that each pair (x, u) computed in lines 14 and 17 satisfies $\mathbb{B}_r(x) \cap \mathbb{Q} \subseteq \bar{\mathcal{X}}(y_\sigma(x), \sigma) \subseteq \Pi(u)$ with $r = \sigma^2 \min_j [q]_j > 0$ if $\underline{\alpha}^* > 0$. Therefore $\mathbb{Y}_0$ computed in line 18 satisfies $\mathbb{B}_r(x) \cap \bar{\mathbb{Y}} = \mathbb{B}_r(x) \cap \mathbb{Q} \cap \bar{\mathbb{Y}} \subseteq \Pi(u) \cap \bar{\mathbb{Y}} = \mathbb{Y}_0$, so that each vertex of $\mathbb{Y}_0$ that is not a vertex of $\bar{\mathbb{Y}}$ is necessarily of a distance of at least r from x. Since x is defined as a vertex of $\bar{\mathbb{Y}}$ in line 13, it follows that PARTITION$(\bar{\mathbb{Y}})$ can be called only a finite number of times before $\mathcal{A}$ covers $\mathbb{Q}$. □

Remark 4.4. *In contrast to methods for determining (robust) invariant sets by iteratively computing (robust) backward reachable sets until convergence (e.g. [27]), Algorithm 4.1 computes an invariant cover by repeatedly partitioning an unexplored region.*

Remark 4.5. *An advantageous property of the invariant cover computed by Algorithm 4.1 is that each subset is represented in the form of $\Pi(u)$ for some $u \in \mathbb{U}$. By the Upper Bound Theorem in [133], the maximum number of vertices of the polytopes of the family is upper bounded by $O((n_q + n_p)^{\lfloor \frac{n_x}{2} \rfloor})$ since the*

maximum number of delimiting planes in $\Pi(u)$ *is* $n_q + n_p$*. Note that this property may significantly reduce the computation of the coder-controller for online implementation.*

4.6 Examples

This section provides two examples to illustrate the proposed algorithm and explore the dependence of computation time on the system dimension. The following numerical experiments were coded using Matlab R2018b with the lrs library [134] and run on a 2.9 GHz Intel Core i7 CPU with 16 GB RAM.

4.6.1 Example 1

Consider system (4.1) with the system parameters (A and B) and set parameters (F_u, f_u, F_w, f_w, Q, q) defined in the following:

$$A = \begin{bmatrix} 0.9225 & 1.0476 \\ 1.0476 & 0.9320 \end{bmatrix}, \quad B = \begin{bmatrix} 1.1518 & 0 \\ 2.4188 & 0.4991 \end{bmatrix}$$

$$F_u = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad f_u = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$F_w = \begin{bmatrix} -0.9847 & 0.1745 \\ -0.4028 & -0.9153 \\ 0.2153 & -0.9766 \\ 0.6809 & -0.7324 \\ 0.4028 & 0.9153 \end{bmatrix}, \quad f_w = \begin{bmatrix} 0.1000 \\ 0.1000 \\ 0.1000 \\ 0.1000 \\ 0.1000 \end{bmatrix}$$

$$Q = \begin{bmatrix} -0.4040 & 0.9148 \\ 0.3521 & 0.9360 \\ -0.7061 & 0.7081 \\ 0.1622 & 0.9868 \\ 0.7061 & -0.7081 \\ -0.1622 & -0.9868 \end{bmatrix}, \quad q = \begin{bmatrix} 1.0572 \\ 1.0744 \\ 1.0704 \\ 1.0000 \\ 1.0000 \\ 1.0000 \end{bmatrix}.$$

We first check the existence of an invariant cover. By solving the optimization problem (4.14), we obtain $\underline{\alpha}^* = 0.2813 > 0$, which from Theorem 4.2 implies the existence of invariant cover.

First, we compare the bounds on the minimal cardinality of the invariant cover in Theorem 4.3 with that in [34]. The lower bound on $|\mathcal{A}|^*$ obtained by Theorem 8 of [34] is 1, while the lower bound from Theorem 4.3 is 5. We can see that the lower bound in our chapter is tighter than that in [34] in this example. One explanation is that the control set is taken into account when deriving the bounds in (4.16) by solving the LPs, while the lower bound in Theorem 8 of [34] is independent of the control set.

By implementing Algorithm 4.1, we compute the invariant cover $(\mathcal{A}, G)$ with cardinality $|\mathcal{A}| = 5$ for the set $\mathbb{Q}$ shown in Figure 4.3. We use this invariant cover to design a static coder–controller as in [34] and compute the state trajectory starting from a vertex of $\mathbb{Q}$, see Figure 4.4 (a). The state remains at all times inside the set $\mathbb{Q}$. The corresponding control input trajectory is shown in Figure 4.4(b). The maximal data-rate needed to guarantee invariance is no greater than $\log_2 5$ bits per sampling interval. Note that the system in this example is open-loop unstable. From Section 1.2.2 of [123], the critical data-rate necessary for the stabilization of this system without disturbances is $\log_2 1.9748$ bits per sampling interval, where 1.9748 is the unstable eigenvalue of the matrix A. This critical data-rate is lower than $\log_2 5$, which is required for the robust invariance specification considered here.

4.6.2 Example 2

We consider a quadrotor which is controlled by a remote computer via a communication network. Following [135], the quadrotor can be modeled as a 6-DOF system. For each axis $j \in \{x, y, z\}$, the dynamics can be expressed as a 2-DOF discrete-time double integrator:

$$\boldsymbol{x}_{j,k+1} = A\boldsymbol{x}_{j,k} + Bu_{j,k} + w_{j,k},$$

where

$$A = \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} \tau^2/2 \\ \tau \end{bmatrix},$$

the state $\boldsymbol{x}_{j,k} = [p_{j,k}\ v_{j,k}]^T \in \mathbb{R}^2$ consists of the position and velocity, the control input $u_{j,k} \in \mathbb{R}$ is the acceleration, $w_{j,k}$ is the external disturbance, and τ is the sampling period. The velocity and control input are subject to the constraints: $v_{j,k} \in [v_{j,\min}, v_{j,\max}]$ and $u_{j,k} \in [u_{j,\min} u_{j,\max}]$. The disturbance $w_{j,k}$ is bounded according to $\|w_{j,k}\|_\infty \leq \bar{w}_j$.

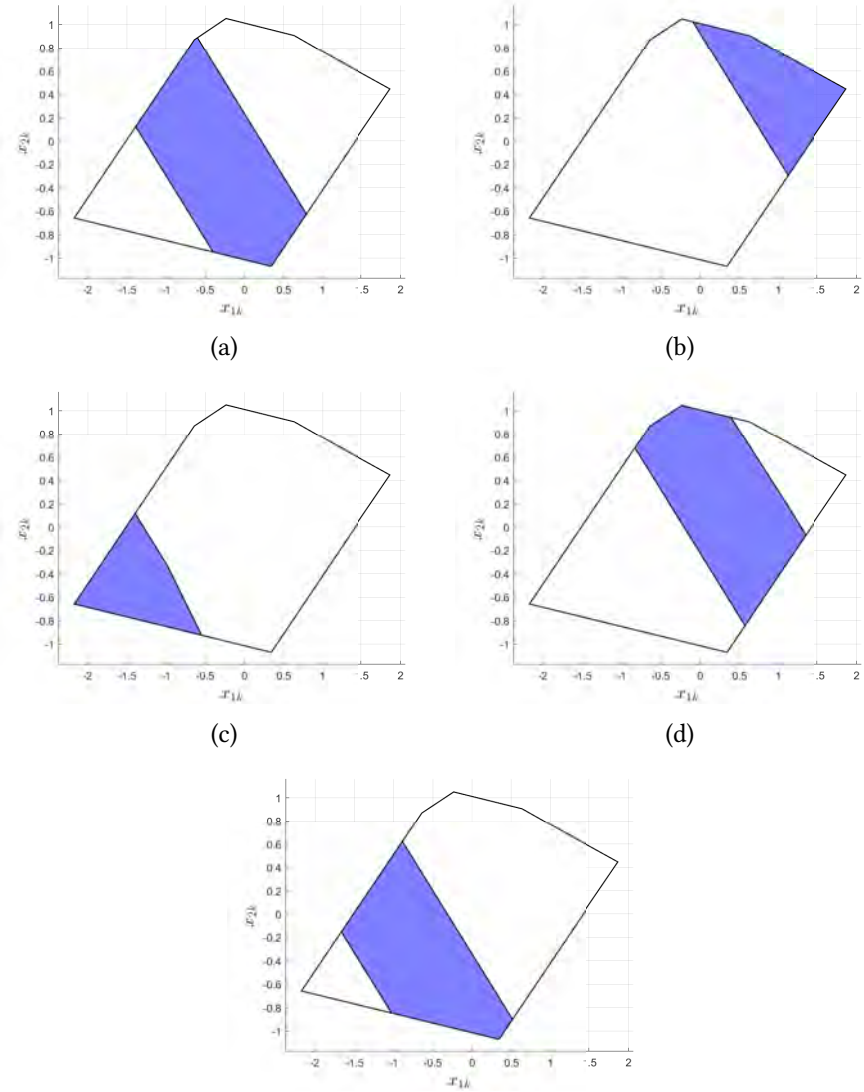
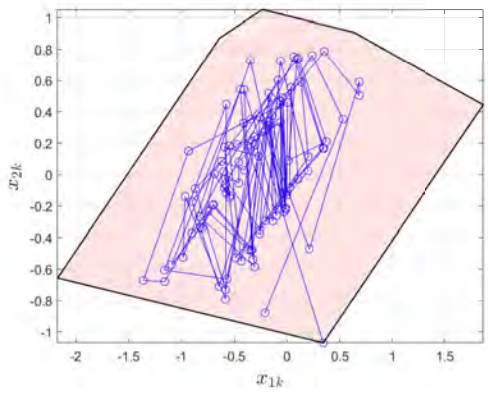
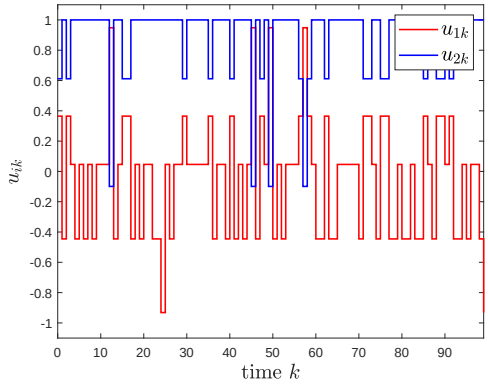


Figure 4.3: Invariant cover $(\mathcal{A}, G)$ used in Example 1: (a) $\mathbb{X}_1^{\text{inv}}$ and $G(\mathbb{X}_1^{\text{inv}}) = [0.0461\ 1]^T$; (b) $\mathbb{X}_2^{\text{inv}}$ and $G(\mathbb{X}_2^{\text{inv}}) = [-0.9317\ 1]^T$; (c) $\mathbb{X}_3^{\text{inv}}$ and $G(\mathbb{X}_3^{\text{inv}}) = [0.9485\ \ -0.0991]^T$; (d) $\mathbb{X}_4^{\text{inv}}$ and $G(\mathbb{X}_4^{\text{inv}}) = [-0.4455\ 1]^T$; (e) $\mathbb{X}_5^{\text{inv}}$ and $G(\mathbb{X}_5^{\text{inv}}) = [0.3649\ 0.6117]^T$.

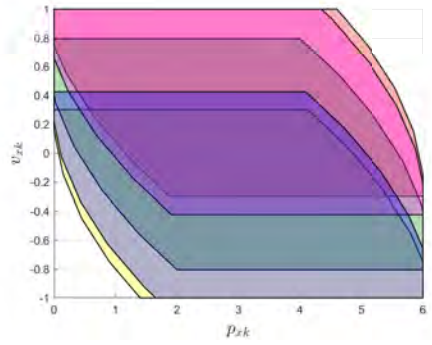


(a)

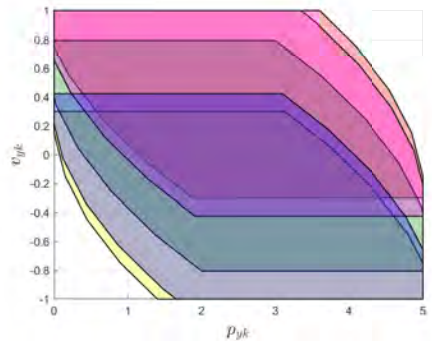


(b)

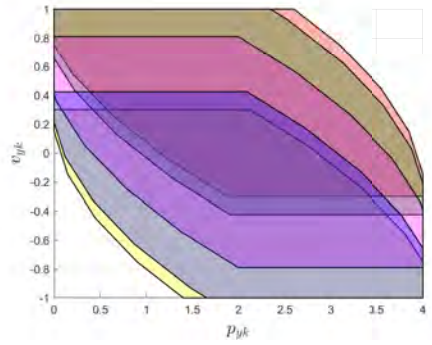
Figure 4.4: One simulated trajectory for the system in Example 1: (a) state trajectory x_k; (b) control input u_k.



(a)

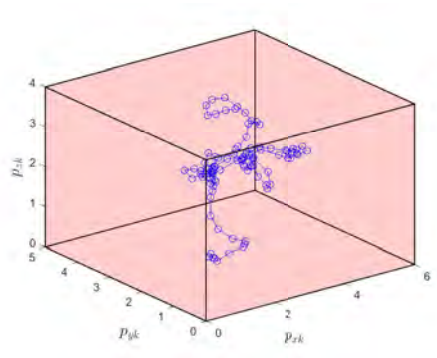


(b)

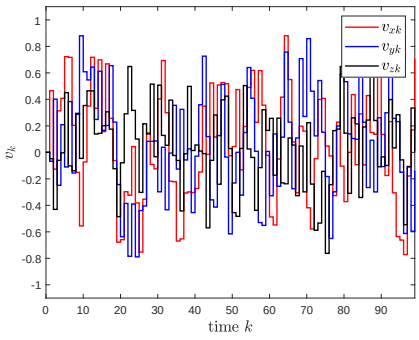


(c)

Figure 4.5: The RCIS $\mathbb{Q}_j$ and the corresponding invariant cover for each axis j in Example 2: (a) x axis; (b) y axis; (c) z axis. Here, the elements in the invariant covers are in different colors.



(a)



(b)

Figure 4.6: One simulated trajectory for the system in Example 2: (a) position trajectory $[p_{xk}\ p_{yk}\ p_{zk}]^T$; (b) velocity trajectory $[v_{xk}\ v_{yk}\ v_{zk}]^T$.

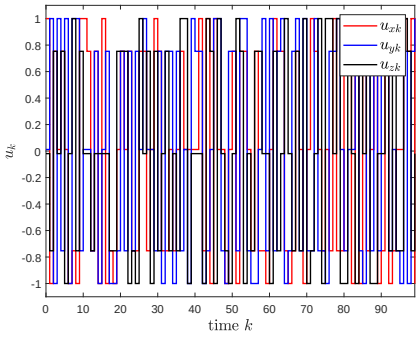


Figure 4.7: Control input u_k in Example 2.

The objective is to design a coder–controller with a finite data-rate such that the quadrotor keeps moving in a safe region as shown in Figure 4.6(a). We first compute the maximal RCIS $\mathbb{Q}_j$ with respect to the safe region for each axis j. By following the results in this chapter, we then compute an invariant cover through Algorithm 4.1 for each $\mathbb{Q}_j$. If the invariant cover exists for each axis, we can design a coder–controller similar to Example 1.

The constraints are defined by the following parameters: $v_{j,\min} = -1\,\mathrm{ms}^{-1}$, $v_{j,\max} = 1\,\mathrm{ms}^{-1}$, $u_{j,\min} = -1\,\mathrm{ms}^{-2}$, $u_{j,\max} = 1\,\mathrm{ms}^{-2}$, $\bar{w}_j = 0.2$, and the sampling interval is $\tau = 0.5\,\mathrm{s}$. The computed RCIS for each axis is shown in Figure 4.5. The lower bound for each axis from Theorem 4.3 is $|\mathcal{A}_j|^* \geq 3$. Note that the origin is not in the interior of the sets $\mathbb{Q}_j$. We need to transform the system such that the sets $\mathbb{Q}_j$ contain the origin before using Algorithm 4.1. We compute an invariant cover $(\mathcal{A}_j, G_j)$ for each $\mathbb{Q}_j$ with cardinality $|\mathcal{A}_j| = 5$, as shown in Figure 4.5. The corresponding static coder–controller ensures that the quadrotor position remains in the safe region at all times and the velocity and acceleration satisfy their constraints. See Figure 4.6 for the position and velocity trajectories and Figure 4.7 for the corresponding control inputs. Note that the control constraints are satisfied. The data-rate needed to enforce the invariance is at most $6 \log_2 5 \approx 13.9316$ bits/s.

4.6.3 Computation Time and Invariant Cover Cardinality

To investigate how the computation required by Algorithm 4.1 depends on the state dimension n_x and control dimension n_u, we consider a range of values for (n_x, n_u) and for each case we generate 100 random systems in the form of (4.1) with linear state and control constraints, and with spectral radii no greater than 1.3. For each realization we find a set $\mathbb{Q}$ that admits an invariant cover and use Algorithm 4.1 to compute an invariant cover. The dependence of computation time on (n_x, n_u) is shown in the boxplot of Figure 4.8. As expected, the computation time increases as the state and control input dimensions increase.

Fig. 4.9 compares $|\mathcal{A}|$, the cardinality of $\mathcal{A}$ computed by Algorithm 4.1, with the upper and lower bounds derived in Section 4 on the minimal cardinality $|\mathcal{A}|^*$ for the same randomly generated systems. The results shown (with $1 \leq |\mathcal{A}| \leq 50$) represent 83% of system realizations. Although Algorithm 4.1 is not guaranteed to find an invariant cover with minimal cardinality, $|\mathcal{A}|$ (shown by the dashed line in Fig. 4.9) does not exceed the upper bound on $|\mathcal{A}|^*$ in all cases. Although Algorithm 4.1 is not guaranteed to find an invariant

cover with minimal cardinality, $|\mathcal{A}|$ does not exceed the upper bound on $|\mathcal{A}|^*$ given by (4.16) in all cases.

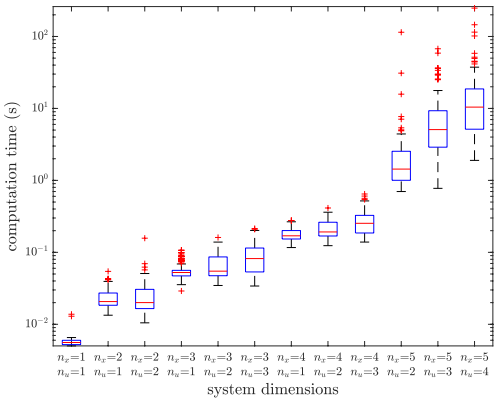
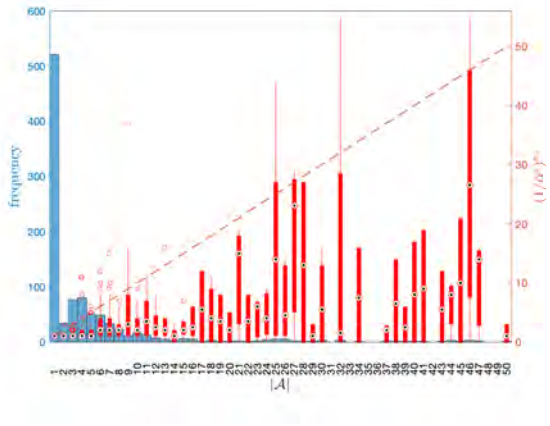


Figure 4.8: Computation time of invariant cover with respect to the state dimension n_x and the control dimension n_u.



(a)

Figure 4.9: In red: box plot of the approximate lower bound $(1/\bar{\alpha}^*)^{n_x}$ on $|\mathcal{A}|^*$. In blue: the observed distribution of $|\mathcal{A}|$.

4.7 Summary

This chapter considered some fundamental problems concerning the invariant cover for uncertain discrete-time linear control systems. We provided computationally tractable necessary and sufficient conditions on the existence of an invariant cover, as well as upper and lower bounds on the minimal cardinality of the invariant cover. In addition, we gave an algorithm to compute an invariant cover in finite time, whenever it exists. Numerical examples were given to illustrate the effectiveness of the results.

Chapter 5

Verification and Control based on Temporal Logic Trees

In the previous two chapters, we developed algorithms to compute probabilistic controlled invariant sets and invariant covers, both of which respect the invariance property. In this chapter, we study the problems of model checking and control synthesis for more complex specification than invariance. We consider discrete-time uncertain systems under linear temporal logic (LTL) specifications. We construct temporal logic trees (TLT) from LTL formulae via reachability analysis. For a given transition system and an LTL formula, we prove that there exist both a universal TLT and an existential TLT via minimal and maximal reachability analysis, respectively. We show that the universal TLT is an underapproximation for the LTL formula and the existential TLT is an overapproximation. We provide sufficient conditions and necessary conditions to verify whether a transition system satisfies an LTL formula by using the TLT approximations. As a major contribution of this chapter, for a controlled transition system and an LTL formula, we prove that a controlled TLT can be constructed from the LTL formula via control-dependent reachability analysis. Based on the controlled TLT, we design an online control synthesis algorithm, under which a set of feasible control inputs can be generated at each time step. We also prove that this algorithm is recursively feasible.

The remainder of this chapter is organized as follows. In Section 5.1, we provide the background and motivations. In Section 5.2, we introduce TLT structures and show how to construct a TLT from a given LTL formula. In Section 5.3, we solve the LTL model checking problem through the constructed TLT. Section 5.4 solves the LTL control synthesis problem. In Section 5.5, we illustrate the effectiveness of our approaches with two numerical examples. In Section 5.6, we conclude the chapter.

5.1 Introduction

In the recent past the integration of computer science and control theory has promoted the development of new areas such as embedded systems design [136], hybrid systems theory [68], and, more recently, cyber-physical systems [69]. Model checking and control synthesis are two fundamental problems in formal verification and control. Both problems are of great interest in disparate and diverse applications, such as robotics, transportation systems, and safety-critical embedded system design. However, they are challenging problems when considering dynamical systems affected by uncertainty, and in particular uncertain infinite (uncountable) systems under complex, temporal logic specifications. In this chapter, we provide solutions to the model checking and formal control synthesis problems, for discrete-time uncertain systems under linear temporal logic (LTL) specification.

5.1.1 Related Work

In general, LTL formulae are expressive enough to capture many important properties, e.g., safety (nothing bad will ever happen), liveness (something good will eventually happen), and more complex combinations of Boolean and temporal statements [36].

In the area of formal verification, a dynamical process is by and large modeled as a finite transition system. A typical approach to both model checking and control synthesis for a finite transition system and an LTL formula leverages automata theory. It is known that each LTL formula can be transformed to an equivalent automaton [137]. The model checking problem can be solved by verifying whether the intersection of the trace set of the transition system and the set of accepted languages of the automaton expressing the negation of the LTL formula is empty, or not [36]. The control synthesis problem can be solved by the following steps: (1) translate the LTL formula into a deterministic automaton; (2) build a "product automaton" between the transition system and the obtained automaton; (3) transform the product automaton into a game [138]; (4) solve the game [139]–[141]; and (5) map the solution into a control strategy.

In recent years, the study of model checking and control synthesis for dynamical systems with continuous (uncountable) spaces, which extends the standard setup in formal verification, has attracted significant attention within the control community. This has enabled the formal control synthesis for in-

teresting properties, which are more complex than the usual control objectives such as stability and set invariance. In order to adapt automaton-based methods to infinite systems, *abstraction* plays a central role in both model checking and control synthesis, which entails: (1) to abstract an infinite system to a finite transition system; (2) to conduct automaton-based model checking or control synthesis for the finite transition system; (3) if a solution is found, to map it back to the infinite system; otherwise, to refine the finite transition system and repeat the steps above.

In order to show the correctness of the solution obtained from the abstracted finite system over the infinite system, an equivalence or inclusion relation between the abstracted finite system and the infinite system needs to be established [65]. Relevant notions include (approximate) bisimulations and simulations. These relations and their variants have been explored for systems that are incrementally (input-to-state) stable [142], [143], or systems with similar properties [66]. Recent work [67] shows that the condition of approximate simulation can be relaxed to controlled globally asymptotic or practical stability with respect to a given set for nonlinear systems. We remark that such condition holds for only a small class of systems in practice.

Based on abstractions, the problem of model checking for infinite systems has been studied in [144], [145]. In [144], it is shown that model checking for discrete-time, controllable, linear systems from LTL formulae is decidable through an equivalent finite abstraction. In [145], the authors study the problem of verifying whether a linear system with additive uncertainty from some initial states satisfies a fragment of LTL formulae, which can be transformed to a deterministic Büchi automaton. The key idea is to use a formula-guided method to construct and refine a finite system abstracted from the linear system and guarantee their equivalence. Along the same line, the problem of control synthesis has also been widely studied for linear systems [146], nonlinear systems [147], stochastic systems [148], hybrid systems [149], and stochastic hybrid systems [150]. The applications of control synthesis under LTL specifications include single-robot control in dynamic environments [70], multi-robot control [71], and transportation control [72].

Beyond automata-based methods, alternative attempts have been made for specific model classes. Receding horizon methods are used to design controllers under LTL for deterministic linear systems [73] and uncertain linear systems [151]. The control of Markov decision processes under LTL is considered in [73] and further applied to multi-robot coordination in [74]. Control

synthesis for dynamical systems has been extended also to other specifications like signal temporal logic (STL) [75], and probabilistic computational tree logic [76]. Interested readers may refer to the tutorial paper [152] and the book [35] for detailed discussions.

5.1.2 Motivations

Although the last two decades have witnessed a great progress on model checking and control synthesis for infinite systems from both theoretical and practical perspectives, there are some inherent restrictions in the dominant automaton-based methods.

First, abstraction from infinite systems to finite systems suffers from the curse of dimensionality: abstraction techniques usually partition the state space, and transitions are constructed via reachability analysis. The computational complexity increases exponentially with the system dimension. Many works are dedicated to improving the computational efficiency by using over-approximation for (mixed) monotone systems [72], or by exploiting the structure of the uncertainty [150]. However, another issue with abstraction techniques is that only systems with "good properties" (e.g., incremental stability, or smooth dynamics) might admit finite abstractions with guarantees, which limits their generality.

Second, there are few results for handling general LTL formulae when an infinite system comes with uncertainty (e.g., bounded disturbance, or additive noise). In most contributions on control synthesis of uncertain systems, fragments of LTL formulae (e.g., bounded LTL or *co-safe* LTL) are usually taken into account [153], [154]. As mentioned before, the LTL formulae are defined over infinite trajectories and it is difficult to control uncertainties propagating along such trajectories. This restriction results from conservative over-approximation in the computation of forward reachable sets, which is widely used for abstraction, and which leads to information loss when used with automaton-based methods.

Third, current methods usually lack online scalability. In many applications, full *a priori* knowledge of a specification cannot be obtained. For example, consider an automated vehicle required to move from some initial position to some destination without colliding into any obstacle (e.g., other vehicles and pedestrians). Since the trajectories of other vehicles and pedestrians cannot be accurately predicted, we cannot in advance define a specification that captures all the possibilities during the navigation process. Thus, offline design

of automaton-based methods is significantly restricted.

Finally, the controller obtained from automaton-based methods usually only contains a single control policy. In some applications, e.g., human-in-the-loop control [155], [156], a set of feasible control inputs are needed to provide more degrees of freedom in the actual implementation. For example, [155] studies a control problem where humans are given a higher priority than the automated system in the decision making process. A controller is designed to provide a set of admissible control inputs with enough degrees of freedom to allow the human operator to easily complete the task.

5.1.3 Contributions

Motivated by the above, this chapter studies LTL model checking and control synthesis for discrete-time uncertain systems. There are many results for reachability analysis on infinite systems [47], [157] and the computation of both forward and backward reachable sets has been widely studied [158]–[160]. The connection between STL and reachability analysis is studied in [161], which inspires our work. The main contributions of this chapter are three-fold:

(C5.1) We construct tree structures from LTL formulae via reachability analysis over dynamical models. We denote the tree structure as a TLT. The connection between TLT and LTL is shown to hold for both uncertain finite and infinite models. The construction of the TLT is abstraction-free for infinite systems and admits online implementation, as demonstrated in Section 5.5. More specifically, given a system and an LTL formula, we prove that both a universal TLT and an existential TLT can be constructed for the LTL formula via minimal and maximal reachability analysis, respectively (Theorems 5.1 and 5.2). We also show that the universal TLT is an underapproximation for the LTL formula and the existential TLT is an overapproximation for the LTL formula. Our formulation does not restrict the generality of LTL formulae.

(C5.2) We provide a method for model checking of discrete-time dynamical systems using TLT. We provide sufficient conditions to verify whether a transition system satisfies an LTL formula by using universal TLT for under-approximating the satisfaction set, or alternatively using existential TLT for over-approximating the violation set (Theorem 5.3). Du-

ally, we provide necessary conditions by using existential TLT for over-approximating the satisfaction set, or alternatively using universal TLT for under-approximating the violation set (Theorem 5.4).

(C5.3) As a core and novel contribution of this chapter, we detail an approach for online control synthesis for a controlled transition system to guarantee that the controlled system will satisfy the specified LTL formula. Given a control system and an LTL formula, we construct a controlled TLT (Theorem 5.5). Based on the obtained TLT, we design an online control synthesis algorithm, under which a set of feasible control inputs is generated at each time step (Algorithm 5.3). We prove that this algorithm is recursively feasible (Theorem 5.6). We provide applications to show the scalability of our methods.

5.2 Temporal Logic Trees

This section will introduce the notion of TLT and establish a satisfaction relation between a trajectory and a TLT. Then, we construct TLT from LTL formulae and discuss the approximation relation between them.

Recall the transition system $\mathsf{TS} = (\mathbb{S}, \to, \mathbb{S}_0, \mathcal{AP}, L)$ defined in Section 2.1.1 and the LTL formulae defined in Section 2.3. Consider a transition system $\mathsf{TS} = (\mathbb{S}, \to, \mathbb{S}_0, \mathcal{AP}, L)$ and an LTL formula φ.

5.2.1 Definitions

Definition 5.1. *A TLT is a tree for which the following statements hold:*

- *each node is either a* set *node,* a subset of $\mathbb{S}$*, or an* operator *node,* from $\{\wedge, \vee, \bigcirc, \mathsf{U}, \Box\}$*;*
- *the root node and the leaf nodes are set nodes;*
- *if a set node is not a leaf node, its unique child is an operator node;*
- *the children of any operator node are set nodes.*

Next we define the complete path and the minimal Boolean fragment for a TLT. Minimal Boolean fragments play an important role when simplifying the TLT for model checking and control synthesis in the following.

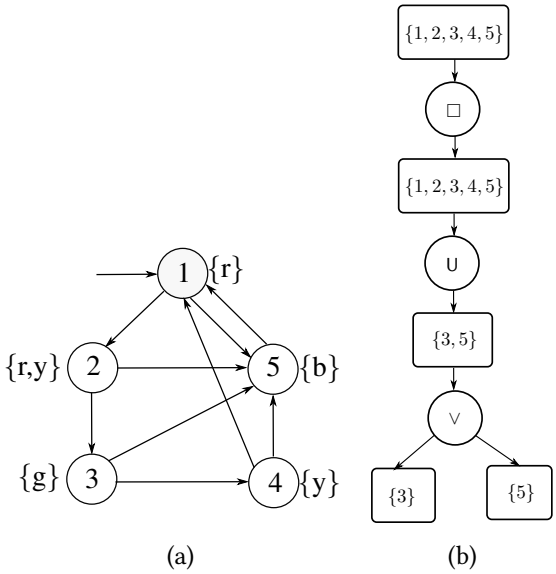


Figure 5.1: (a) A transition system illustrating a traffic light example. Labels are shown aside the states. The initial state is denoted by an incoming edge. (b) A TLT corresponding to an LTL formula $\varphi = \Box\Diamond(g \vee b)$ for this system. Note that $\Diamond\varphi = \text{true} \cup \varphi$.

Definition 5.2. *A complete path of a TLT is a sequence of nodes and edges from the root node to a leaf node. Any subsequence of a complete path is called a fragment of the complete path.*

Definition 5.3. *A minimal Boolean fragment of a complete path is one of the following fragments:*

(i) a fragment from the root node to the first Boolean operator node ($\wedge$ or $\vee$) in the complete path;

(ii) a segment from one Boolean operator node to the next Boolean operator node in the complete path;

(iii) a fragment from the last Boolean operator node of the complete path to the leaf node;

Example 5.1. *Consider the traffic light in Example 2.1, as shown in Figure 5.1(a) and the TLT in Figure 5.1(b), which corresponds to the LTL formula* $\varphi = \Box\Diamond(g \vee b)$ *(the formal construction of a TLT from an LTL formula will be detailed in next subsection). We encode one of the complete paths of this TLT in the form of* $\mathbb{X}_0\Box\mathbb{X}_1\mathsf{U}\mathbb{X}_2 \vee \mathbb{X}_3$*, where* $\mathbb{X}_0 = \mathbb{X}_1 = \{1,2,3,4,5\}$*,* $\mathbb{X}_2 = \{3,5\}$*, and* $\mathbb{X}_3 = \{3\}$*. For this complete path, the minimal Boolean fragments consist of* $\mathbb{X}_0\Box\mathbb{X}_1\Diamond\mathbb{X}_2\vee$ *and* $\vee\mathbb{X}_3$*, which correspond to cases (i) and (iii) in Definition 5.3, respectively.*

We now define the satisfaction relation between a given trajectory and a complete path of a TLT.

Definition 5.4. *Consider a trajectory* $\boldsymbol{p} = x_0x_1\ldots x_k\ldots$ *and encode a complete path of a TLT in the form of* $\mathbb{X}_0 \odot_1 \mathbb{X}_1 \odot_2 \ldots \odot_{N_f} \mathbb{X}_{N_f}$ *where* N_f *is the number of operators in the complete path,* $\mathbb{X}_i \subseteq \mathbb{S}$ *for all* $i \in \mathbb{N}_{[0,N_f]}$ *and* $\odot_i \in \{\wedge, \vee, \bigcirc, \mathsf{U}, \Box\}$ *for all* $i \in \mathbb{N}_{[1,N_f]}$*. The trajectory* $\boldsymbol{p}$ *is said to satisfy this complete path if* $x_0 \in \mathbb{X}_0$ *and there exists a sequence of time steps* $k_0k_1,\ldots,k_{N_f}$ *with* $k_i \in \mathbb{N}$ *for all* $i \in \mathbb{N}_{[0,N_f]}$ *and* $0 \triangleq k_0 \leq k_1 \leq k_2 \leq \ldots \leq k_{N_f}$ *such that for all* $i \in \mathbb{N}_{[0,N_f]}$*,*

(i) if $\odot_i = \wedge$ *or* $\odot_i = \vee$*,* $x_{k_i} \in \mathbb{X}_{i-1}$ *and* $x_{k_i} \in \mathbb{X}_i$*;*

(ii) if $\odot_i = \bigcirc$*,* $x_{k_i-1} \in \mathbb{X}_{i-1}$ *and* $x_{k_i} \in \mathbb{X}_i$*;*

(iii) if $\odot_i = \mathsf{U}$*,* $x_j \in \mathbb{X}_{i-1}, \forall j \geq \mathbb{N}_{[k_{i-1},k_i-1]}$*, and* $x_{k_i} \in \mathbb{X}_i$*;*

(iv) if $\odot_i = \Box$*,* $x_j \in \mathbb{X}_i, \forall j \geq k_i$*.*

Consider a k*-th prefix* $\boldsymbol{p}[..k] = x_0x_1\ldots x_k$ *from* $\boldsymbol{p}$ *and a fragment from the complete path in the form of* $\mathbb{X}_0 \odot_1 \mathbb{X}_1 \odot_2 \ldots \odot_{N'_f} \mathbb{X}_{N'_f}$ *where* $N'_f \leq N_f$*. The prefix* $\boldsymbol{p}[..k]$ *is said to satisfy this fragment if* $x_0 \in \mathbb{X}_0$*,* $x_k \in \mathbb{X}_{N'_f}$*, and there exists a sequence of time steps* $k_0k_1,\ldots,k_{N'_f}$ *with* $k_i \in \mathbb{N}$ *for all* $i \in \mathbb{N}_{[0,N'_f]}$ *and* $0 \triangleq k_0 \leq k_1 \leq k_2 \leq \ldots \leq k_{N'_f} \leq k$*, such that for all* $i \in \mathbb{N}_{[0,N'_f]}$*, (i)–(iii) holds and furthermore*

(iv') if $\odot_i = \Box$*,* $x_j \in \mathbb{X}_i, \forall j \in \mathbb{N}_{[k_i,k]}$*.*

Definition 5.5. *A time coding of a TLT is an assignment of each operator node in the tree to a nonnegative integer.*

Definition 5.6. *Consider a trajectory $\boldsymbol{p} = x_0 x_1 \dots x_k \dots$ and a TLT. The trajectory $\boldsymbol{p}$ is said to satisfy the TLT if there exists a time coding such that the output of Algorithm 5.1 is* true.

The time coding indicates when the operators in the TLT are activated along a given trajectory. Algorithm 5.1 provides a procedure to test if a trajectory satisfies a TLT under a given time coding. The TLT is first transformed into a compressed tree, which is analogous to a binary decision diagram (lines 1–2), through Algorithm 5.2. Then, we check if the trajectory satisfies each complete path of the TLT under the time coding (lines 3–9). Finally, we backtrack the tree with output true or false. If the output is true, the trajectory satisfies the TLT; otherwise, the trajectory does not satisfy the TLT under the given time coding.

Algorithm 5.2 aims to obtain a tree in a compact form. Each minimal Boolean fragment is encoded according to Definition 5.3. The notation $\odot_i$ denotes the operator node and N_f denotes the number of set nodes in the corresponding minimal Boolean fragment. We compress the sets in the minimal Boolean fragment to be a single set. The simplified tree consists of set nodes and Boolean operator nodes.

Example 5.2. *From Definition 5.4, we can verify that the trajectory $\boldsymbol{p} = (1234)^\omega$ satisfies the complete path given in Example 5.1 by choosing $k_0 = k_1 = 0$ and $k_2 = k_3 = 2$. It follows from Definition 5.6 that this trajectory satisfies the corresponding TLT.*

5.2.2 Construction and Approximation of TLT

We define the approximation relations between TLT and LTL formulae as follows.

Definition 5.7. *A TLT is said to be an under-approximation of an LTL formula φ if all the trajectories that satisfy the TLT also satisfy φ.*

Definition 5.8. *A TLT is said to be an over-approximation of an LTL formula φ, if all the trajectories that satisfy φ also satisfy the TLT.*

The following two theorems show how to construct TLT via reachability analysis for the LTL formulae, and discuss their approximation relations. Recall the minimal reachability in Section 2.2.1, and more specifically, the definitions of the reachability operators $\mathcal{R}^{\mathrm{m}}$ and $\mathcal{RI}$.

Algorithm 5.1 TLT Satisfaction

Input: a trajectory $\boldsymbol{p} = x_0 x_1 \ldots x_k \ldots$, a TLT and a time coding
Output: true or false;

construct a compressed tree via Algorithm 5.2 with input of the TLT;
replace all set nodes of the compressed tree with false;
for each complete path of the TLT **do**
  if $\boldsymbol{p}$ satisfies the complete path under the time coding **then**
    set the corresponding leaf node in the compressed tree with true;
  else
    set the corresponding leaf node in the compressed tree with false;
  end if
end for
backtrack the tree;
return the root node of the tree.

Theorem 5.1. *For any transition system* TS *and any LTL formula* φ*,*

(i) a TLT can be constructed from the formula $\forall\varphi$ *through the reachability operators* $\mathcal{R}^{\mathrm{m}}$ *and* $\mathcal{RI}$*;*

(ii) this TLT is an under-approximation of φ*.*

Proof sketch. See Appendix for a detailed proof in the end of this chapter.

We prove the constructability by the following three steps: (1) we transform the given LTL formula φ into an equivalent LTL formula in the weak-until positive normal form; (2) for each atomic proposition $a \in \mathcal{AP}$, we show that a TLT can be constructed from $\forall a$ (or $\forall\neg a$); (3) we leverage induction to show the following: given LTL formulae φ, φ_1, and φ_2 in weak-until positive normal form, if TLT can be constructed from $\forall\varphi$, $\forall\varphi_1$, and $\forall\varphi_2$, respectively, then TLT can also be constructed through reachability operators $\mathcal{R}^{\mathrm{m}}$ and $\mathcal{RI}$ from the formulae $\forall(\varphi_1 \wedge \varphi_2)$, $\forall(\varphi_1 \vee \varphi_2)$, $\forall\bigcirc\varphi$, $\forall(\varphi_1 \mathsf{U} \varphi_2)$, and $\forall(\varphi_1 \mathsf{W} \varphi_2)$, respectively.

We follow a similar approach to prove an under-approximation relation between the constructed TLT and the LTL formula. The under-approximation occurs due to the conjunction operator and the presence of branching in the transition system. □

Further recall the maximal reachability in Section 2.2.1, and more specifically, the definitions of the reachability operators $\mathcal{R}^{\mathrm{M}}$ and $\mathcal{I}$.

Algorithm 5.2 Tree Compression

Input: a tree
Output: a compressed tree

for each complete path of the tree **do**
  for each minimal Boolean fragment **do**
    switch minimal Boolean fragment **do**
      case (i) in Definition 5.3
        encode the fragment in the form of $\mathbb{Y}_1 \odot_1 \ldots \odot_i \ldots \mathbb{Y}_{N_f} \odot_{N_f}$ with $\odot_{N_f} \in \{\wedge, \vee\}$;
        replace the fragment with $\cup_{i=1}^{N_f} \mathbb{Y}_i \odot_{N_f}$;
      case (ii) in Definition 5.3
        encode the fragment in the form of $\odot_1 \mathbb{Y}_1 \odot_2 \ldots \odot_{N_f} \mathbb{Y}_{N_f} \odot_{N_f+1}$ with $\odot_1, \odot_{N_f+1} \in \{\wedge, \vee\}$;
        replace the fragment with $\odot_1 \cup_{i=1}^{N_f} \mathbb{Y}_i \odot_{N_f+1}$;
      case (iii) in Definition 5.3
        encode the fragment in the form of $\odot_1 \mathbb{Y}_1 \odot_2 \ldots \odot_{N_f} \mathbb{Y}_{N_f}$ with $\odot_1 \in \{\wedge, \vee\}$;
        replace the fragment with $\odot_1 \cup_{i=1}^{N_f} \mathbb{Y}_i$;
    $\triangleright$ *$\odot_i$ denotes the operator node and N_f denotes the number of set nodes in the minimal Boolean fragment;*
  end for
end for
return the updated tree.

Theorem 5.2. *For any transition system* TS *and any LTL formula* φ*,*

(i) a TLT can be constructed from the formula $\exists\varphi$ *through the reachability operators* $\mathcal{R}^{\mathrm{M}}$ *and* $\mathcal{I}$*;*

(ii) this TLT is an over-approximation of φ*.*

Proof. The proof of the first part is similar to that of Theorem 5.1 by replacing the universal quantifier $\forall$ and the reachability operators $\mathcal{R}^{\mathrm{m}}$ and $\mathcal{RI}$ with the existential quantifier $\exists$ and the operators $\mathcal{R}^{\mathrm{M}}$ and $\mathcal{I}$, respectively. Also, the proof of the second part is similar to that of Theorem 5.1 by following the definition of the maximal reachability analysis. □

We call the constructed TLT of $\forall\varphi$ the *universal TLT* of φ and the TLT of $\exists\varphi$ the *existential TLT* of φ. We remark that the constructed TLT is not unique: this is because an LTL formula can have different equivalent expressions (e.g., normal forms). Despite this, the approximation relations between an LTL formula and the corresponding TLT still hold.

The following corollary shows that the approximation relation between TLT and LTL formulae are tight for deterministic transition systems.

Corollary 5.1. *For any deterministic transition system* TS *and any LTL formula* φ*, the universal TLT and the existential TLT of* φ *are identical.*

Proof. If the system is deterministic, it follows from Lemmas 2.1–2.2 and Lemmas 2.3–2.4 that for any $\Omega_1, \Omega_2 \subseteq \mathbb{S}$ and $\Omega \subseteq \mathbb{S}$, $\mathcal{R}^{\mathrm{m}}(\Omega_1, \Omega_2) = \mathcal{R}^{\mathrm{M}}(\Omega_1, \Omega_2)$ and $\mathcal{RI}(\Omega) = \mathcal{I}(\Omega)$. Then, by the same construction procedure, we have that the constructed universal TLT is the same as the constructed existential TLT. □

Remark 5.1. *Computation of reachable sets plays a central role in the construction of the TLT. The computation of reachable sets is not the focus of this chapter. Interested readers are referred to relevant results [158]–[160] and associated computational tools, e.g., the multi-parametric toolbox [61] and the Hamilton-Jacobi toolbox [62].*

Example 5.3. *Consider the traffic light in Example 2.1 and the LTL formula* $\varphi = \Box\Diamond(g \vee b)$ *in Example 5.2 again. We follow the proof of Theorem 5.1 to show the correspondence between* $\forall\varphi$ *and the TLT in Figure 5.1:*

(1) the universal TLT of g is a single set node, i.e., $\{3\}$ and the universal TLT of b is also a single set node, i.e., $\{5\}$;

(2) the root node of the universal TLT of $g \vee b$ is the union of $\{3\}$ and $\{5\}$, i.e., $\{3, 5\}$;

(3) the root node of the universal TLT of $\Diamond(g \vee b)$ is $\mathcal{R}^{\mathrm{m}}(\mathbb{S}, \{3, 5\}) = \{1, 2, 3, 4, 5\}$;

(4) the root node of the universal TLT of $\Box\Diamond(g \vee b)$ is $\mathcal{RI}(\{1, 2, 3, 4, 5\}) = \{1, 2, 3, 4, 5\}$.

We can follow the same steps in the proof of Theorem 5.2 to construct the existential TLT of φ, which is the same as the universal TLT of φ for the system in Example 2.1.

5.3 Model Checking

This section focuses on the model checking problem.

Problem 5.1. *Consider a transition system* TS *and an LTL formula φ. Verify whether* $\mathsf{TS} \vDash \varphi$*, i.e., $\forall x_0 \in \mathbb{S}_0$, $x_0 \vDash \forall\varphi$.*

Thanks to the approximation relations between the TLT and the LTL formulae, we obtain the following lemma.

Lemma 5.1. *For any transition system* TS *and any LTL formula φ,*

(i) $x_0 \vDash \forall\varphi$ if x_0 belongs to the root node of the universal TLT of φ;

(ii) $x_0 \vDash \exists\varphi$ only if x_0 belongs to the root node of the existential TLT of φ.

Proof. The first result directly follows from that the root node of the universal TLT is an under-approximation of the satisfaction set of φ, as shown in Theorem 5.1. Dually, the second result follows from that the root node of the universal TLT is an over-approximation of the satisfaction set of φ, shown in Theorem 5.2. □

The next theorem provides two sufficient conditions for solving Problem 5.1.

Theorem 5.3. *For a transition system* TS *and an LTL formula* φ, $\mathsf{TS} \vDash \varphi$ if *one of the following conditions holds:*

(i) the initial state set $\mathbb{S}_0$ *is a subset of the root node of the universal TLT for* φ*;*

(ii) no initial state from $\mathbb{S}_0$ *belongs to the root node of the existential TLT for* $\neg\varphi$*.*

Proof. Condition (i) directly follows from the first result of Lemma 5.1. Let us next prove condition (ii). It follows that

$$\mathsf{TS} \vDash \varphi \Leftrightarrow \forall \boldsymbol{p} \in \mathsf{Trajs}(\mathsf{TS}), \boldsymbol{p} \vDash \varphi \Leftrightarrow \forall \boldsymbol{p} \in \mathsf{Trajs}(\mathsf{TS}), \boldsymbol{p} \nvDash \neg\varphi.$$

From the second result of Lemma 5.1, if x_0 does not belong to the root node of the existential TLT of $\neg\varphi$, we have $\boldsymbol{p} \nvDash \neg\varphi$, $\forall \boldsymbol{p} \in \mathsf{Trajs}(x_0)$. Thus, the condition (ii) is sufficient for verifying $\mathsf{TS} \vDash \varphi$. □

Similarly, we derive two necessary conditions for solving the model checking problem.

Theorem 5.4. *For a transition system* TS *and an LTL formula* φ, $\mathsf{TS} \vDash \varphi$ only if *one of the following conditions holds:*

(i) the initial state set $\mathbb{S}_0$ *is a subset of the root node of the existential TLT for* φ*;*

(ii) no initial state from $\mathbb{S}_0$ *belongs to the root node of the universal TLT for* $\neg\varphi$*.*

Proof. Similar to Theorem 5.3. □

Notice that the approximation relations between the TLT and the LTL formula are tight for deterministic transition systems, as shown in Corollary 5.1. In this case, the model checking problem can be tackled as follows.

Corollary 5.2. *For a deterministic transition system* TS *and an LTL formula* φ, $\mathsf{TS} \vDash \varphi$ *if and only if the initial state set* $\mathbb{S}_0$ *is a subset of the root node of the universal (or existential) TLT for* φ*.*

Proof. Follows from Corollary 5.1. □

The conditions in Theorems 5.3–5.4 imply that one can directly do model checking by using TLT, as shown in the following example.

Example 5.4. *Let us continue to consider the traffic light and the LTL formula* $\varphi = \Box\Diamond(g \vee b)$*. Let us verify whether* $\mathsf{TS} \models \varphi$ *by using the above method. Since the unique initial state* x_0 *belongs to the root node of the universal TLT of* φ *shown in Figure 5.1, it follows from condition (i) in Theorem 5.3 that* $\mathsf{TS} \models \varphi$*. Next, we show how to use condition (ii) to verify that* $\mathsf{TS} \models \varphi$*.*

First of all, we have $\neg\varphi = \Diamond\Box(\neg g \wedge \neg b)$*. Following the proof of Theorem 5.2, we construct the existential TLT of* $\neg\varphi$*:*

(1) the existential TLT of $\neg g$ *is a single set node, i.e.,* $\{1, 2, 4, 5\}$ *and the existential TLT of* $\neg b$ *is also a single set node, i.e,,* $\{1, 2, 3, 4\}$*;*

(2) the root node of the existential TLT of $\neg g \wedge \neg b$ *is the intersection of* $\{1, 2, 4, 5\}$ *and* $\{1, 2, 3, 4\}$*, i.e.,* $\{1, 2, 4\}$*;*

(3) the root node of the existential TLT of $\Box(\neg g \wedge \neg b)$ *is* $\mathcal{I}(\{2, 3, 4, 5\}) = \emptyset$*.*

As the existential TLT of $\neg\varphi$ *is the empty set* $\emptyset$*, this implies that condition (ii) in Theorem 5.3 holds and thus* $\mathsf{TS} \models \varphi$*.*

5.4 Control Synthesis

Recall the controlled transition system defined in Section 2.1.2 and the controlled reachability analysis in Section 2.2.2. More specifically, recall the definitions of the reachability operators $\mathcal{R}^{\mathrm{c}}$ and $\mathcal{RCI}$. Consider a controlled transition system $\mathsf{CTS} = (\mathbb{S}, \mathbb{U} \to, \mathbb{S}_0, \mathcal{AP}, L)$ and an LTL formula φ. This section will first show the construction of TLT from an LTL formula for a controlled transition system and then provide a control synthesis algorithm based the TLT.

5.4.1 Construction and Approximation of TLT

The next theorem shows how to construct a TLT from an LTL formula for a controlled transition system and discusses its approximation relation.

Theorem 5.5. *For a controlled transition system* CTS *and any LTL formula* φ*, the following holds:*

(i) a TLT can be constructed from the formula φ *through the reachability operators* $\mathcal{R}^{\mathrm{c}}$ *and* $\mathcal{RCI}$*;*

(ii) given an initial state x_0, if there exists a policy $\boldsymbol{\mu}$ such that $\boldsymbol{p}$ satisfies the constructed TLT, $\forall \boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu})$, then $\boldsymbol{p} \vDash \varphi$, $\forall \boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu})$.

Proof. The proof of the first part is similar to that of Theorem 5.1 by replacing the reachability operators $\mathcal{R}^{\mathrm{m}}(\cdot)$ and $\mathcal{RI}(\cdot)$ with $\mathcal{R}^{\mathrm{c}}(\cdot)$ and $\mathcal{RCI}(\cdot)$, respectively.

Similar to the under-approximation of the universal TLT in Theorem 5.1, we can show that the path satisfying the constructed TLT in the first part also satisfies the LTL formula. Then, we can directly prove the second result. □

Let us call the constructed TLT of φ in Theorem 5.5 the controlled TLT of φ.

Remark 5.2. *Checking whether there exists a policy, such that all the resulting trajectories satisfy* the obtained controlled TLT*, is in general a hard problem. A straightforward necessary condition is that x_0 belongs to the root node of the controlled TLT: however, this is neither a necessary nor a sufficient condition on the existence of a policy such that all the resulting trajectories satisfy* the given LTL formula. *A (rather conservative) necessary condition for the latter case can be obtained by regarding the controlled TS as a non-deterministic transition system, and then applying Thereom 5.4.*

Next we will show how to construct the controlled TLT through an example.

Example 5.5. *Consider the controlled transition system in Example 2.2, shown in Figure 5.2(a). For an LTL formula $\varphi = \Diamond\Box o_2$, we can follow the steps in the proof of Theorem 5.5 to construct the controlled TLT of φ, as shown in Figure 5.2(b).*

5.4.2 Control Synthesis Algorithm

In this subsection, we solve the following control synthesis problem.

Problem 5.2. *Consider a controlled transition system* CTS *and an LTL formula φ. For an initial state $x_0 \in \mathbb{S}_0$, find, whenever existing, a sequence of feedback control inputs $\boldsymbol{u} = u_0(x_0)u_1(x_1)\dots u_k(x_k)\dots$ such that the resulting trajectory $\boldsymbol{p} = x_0x_1\dots x_k\dots$ satisfies φ.*

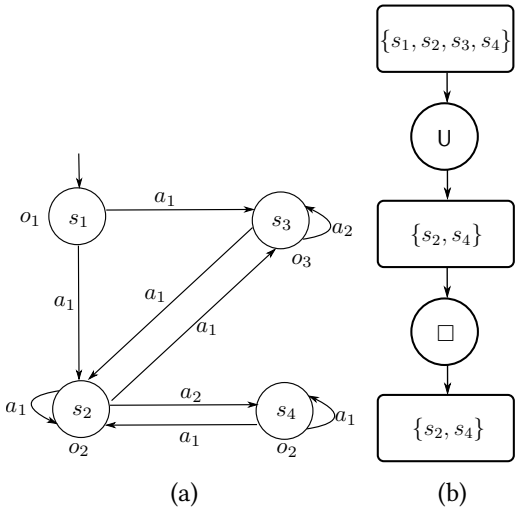


Figure 5.2: (a) Graph description of a controlled transition system; (b) The controlled TLT of $\varphi = \Diamond\Box o_2$ for the system.

Remark 5.3. *Note that the objective of the above problem is not to find a policy $\boldsymbol{\mu}$, but a sequence of control inputs that depend on the measured state. In general, synthesizing a policy $\boldsymbol{\mu}$ such that each trajectory $\boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu})$ satisfies φ is computationally intractable for infinite systems. Instead, here we seek to find online a feasible control input at each time step, in a similar spirit to constrained control or receding horizon control.*

Instead of directly solving Problem 5.2, we consider the following related task, whose solution is also a solution to Problem 5.2, thanks to Theorem 5.5.

Problem 5.3. *Consider a controlled transition system* CTS *and an LTL formula φ. For an initial state $x_0 \in \mathbb{S}_0$, find, whenever existing, a sequence of control inputs $\boldsymbol{u} = u_0(x_0)u_1(x_1)\dots u_k(x_k)\dots$ such that the resulting trajectory $\boldsymbol{p} = x_0x_1\dots x_k\dots$ satisfies the controlled TLT constructed from φ.*

Algorithm 5.3 provides a solution to Problem 5.3. In particular, Algorithm 5.3 presents an online feedback control synthesis procedure, which consists of three steps: (1) control tree: replace each set node of the TLT with a corresponding control set candidate (Algorithm 5.4); (2) compressed control tree: compress the control tree as a new tree whose set nodes are control sets

and whose operator nodes are conjunctions and disjunctions (Algorithm 5.2); (3) backtrack on the control sets through a bottom-up traversal over the compressed control tree (Algorithm 5.5). If the output of Algorithm 5.5 is NExis, there does not exist a feasible solution to Problem 5.3. We remark that Algorithm 5.3 is implemented in a similar way to receding horizon control with the prediction horizon being one.

Algorithm 5.3 Online Feedback Control Synthesis via TLT

Input: an initial state $x_0 \in \mathbb{S}_0$ and the controlled TLT of an LTL formula φ

Output: NExis or $(\boldsymbol{u}, \boldsymbol{p})$ with $\boldsymbol{u} = u_0 u_1 \dots u_k \dots$ and $\boldsymbol{p} = x_0 x_1 \dots x_k \dots$

1: initialize $k = 0$;
2: construct a control tree via Algorithm 5.4, with inputs $\boldsymbol{p}[..k] = x_0 \dots x_k$ and the TLT;
3: construct a compressed tree via Algorithm 5.2, with input the control tree;
4: synthesize a control set $\mathbb{U}_k^{\varphi}(x_k)$ via Algorithm 5.5, with input the compressed tree;
5: **if** $\mathbb{U}_k^{\varphi}(x_k) = \emptyset$ **then**
6: stop and **return** NExis;
7: **else**
8: choose $u_k \in \mathbb{U}_k^{\varphi}(x_k)$;
9: implement u_k and measure $x_{k+1} \in \mathsf{Post}(x_k, u_k)$;
10: update $k = k + 1$ and go to Step 2;
11: **end if**

Algorithm 5.4 aims to construct a control tree that enjoys the same tree structure as the input TLT. The difference is that all the set nodes are replaced with the corresponding control set nodes. The correspondence is established as follows: (1) whether the initial state x_0 belongs to the root node or not (lines 1–3); (2) whether the prefix $\boldsymbol{p}[..k]$ satisfy the fragment from the root node to the set node (lines 5–7); (4) whether or not the set node is a leaf node (lines 9–14); (5) which operator the child of the set node is (lines 16–24).

Algorithm 5.5 aims to backtrack a set by using the compressed tree. We update the parent of each Boolean operator node through a bottom-up traversal.

Note that the construction of a control tree in Algorithm 5.4 is closely related to the controlled reachability analysis in Section 2.2.2. In lines 12–13,

the computation of control set follows from the definition of robust invariant set (RCIS). In lines 22–23, the definition of one-step controlled reachable set is used to compute the control set. In lines 24–26, the control set is synthesized from the definition of a controlled reachable set.

The following theorem shows that Algorithm 5.3 is recursively feasible. This means that initial feasibility implies future feasibility. This is an important property, particularly used in receding horizon control.

Theorem 5.6. *Consider a controlled transition system* CTS*, an LTL formula φ, and an initial state $x_0 \in \mathbb{S}_0$. Let x_0 and the controlled TLT of φ be the inputs of Algorithm 5.3. If there exists a policy $\boldsymbol{\mu}$ such that $\boldsymbol{p}$ satisfies the controlled TLT of φ, $\forall \boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu})$, then*

(i) the control set $\mathbb{U}_k^{\varphi}(x_k)$ (line 8 of Algorithm 5.3) is nonempty for all $k \in \mathbb{N}$;

(ii) at each time step k, there exists at least one trajectory $\boldsymbol{p}$ with prefix $\boldsymbol{p}[..k+1] = x_0 \dots x_k x_{k+1}$ under some policy such that $\boldsymbol{p}$ satisfies the controlled TLT of φ, $\forall u_k \in \mathbb{U}_k^{\varphi}(x_k)$ and $\forall x_{k+1} \in \mathsf{Post}(x_k, u_k)$.

Proof. The proof follows from the construction of the set $\mathbb{U}_k^{\varphi}(x_k)$ in Algorithm 5.4 and the operations in Algorithms 5.2 and 5.5, and the definitions of controlled reachable sets and RCISs. If there exists a policy $\boldsymbol{\mu}$ such that $\boldsymbol{p}$ satisfies the controlled TLT of φ, $\forall \boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu})$, we have that Algorithm 5.3 is feasible at each time step k, which implies that $\mathbb{U}_k^{\varphi}(x_k) \neq \emptyset$. Furthermore, from Algorithm 5.4, each element in $\mathbb{U}_k^{\varphi}(x_k)$ guarantees the one-step ahead feasibility for all realizations of $x_{k+1} \in \mathsf{Post}(x_k, u_k)$, which implies the result (ii). □

Theorem 5.6 implies that if there exists a policy such that all the resulting trajectories satisfy the controlled TLT built from φ, then Algorithm 5.3 is always feasible at each time step in two senses: (1) the control set $\mathbb{U}_k^{\varphi}(x_k)$ is nonempty; and (2) there always exists a feasible policy such that the trajectories with the realized prefix satisfy the controlled TLT.

Remark 5.4. *In Algorithm 5.3, the integration of Algorithms 5.2, 5.4, and 5.5 can be interpreted as a feedback control law. This control law is a set-valued map $\mathbb{S}^{k+1} \to 2^{\mathbb{U}}$ at time step k. Given the prefix $\boldsymbol{p}[..k] = x_0 \dots x_k$, this map collects all the feasible control inputs such that the state can move along the TLT from $\boldsymbol{p}[..k]$.*

Algorithm 5.4 Control Tree

Input: $\boldsymbol{p}[..k] = x_0 \dots x_k$ and a TLT
Output: a control tree

if $k = 0$ and $x_0 \notin$ the root node of TLT **then**
  return $\emptyset$
else
  for each set node $\mathbb{X}$ of TLT through a bottom-up traversal **do**
    if $\boldsymbol{p}[..k]$ does not satisfy the fragment from the root node to $\mathbb{X}$ **then**
      $\triangleright$ *see Definition 5.4*;
      replace $\mathbb{X}$ with $\emptyset$;
    else
      if $\mathbb{X}$ is leaf node **then**
        if the parent of $\mathbb{X}$ is $\Box$ **then**
          replace $\mathbb{X}$ with $\mathbb{U}_{\mathbb{X}} = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathcal{RCI}(\mathbb{X})\}$;
        else
          replace $\mathbb{X}$ with $\mathbb{U}$;
        end if
      else
        switch the child of $\mathbb{X}$ **do**
          case $\wedge$ (or $\vee$)
            replace $\mathbb{X}$ with $\mathbb{U}_{\mathbb{X}} = \cap_{i \in \mathrm{CH}} \mathbb{U}_{\mathrm{CH},i}$ (or $\mathbb{U}_{\mathbb{X}} = \cup_{i \in \mathrm{CH}} \mathbb{U}_{\mathrm{CH},i}$)
            $\triangleright$ *for each Boolean operator node,* CH *collects its children and* $\mathbb{U}_{\mathrm{CH},i}$ *is the corresponding control set for each child;*
          case $\bigcirc$
            replace $\mathbb{X}$ with $\mathbb{U}_{\mathbb{X}} = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathbb{Y}\}$
            $\triangleright$ $\mathbb{Y}$ *is the child of* $\bigcirc$*;*
          case U or $\Box$
            replace $\mathbb{X}$ with $\mathbb{U}_{\mathbb{X}} = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathbb{X}\}$;
        end if
      end if
  end for
  return the updated tree as the control tree.
end if

Algorithm 5.5 Set Backtracking

Input: a compressed tree
Output: a set $\mathbb{U}_k^{\varphi}$

1: **for** each Boolean operator node of the compressed tree through a bottom-up traversal **do**
2: **switch** Boolean operator **do**
3: **case** $\wedge$
4: replace its parent with $\mathbb{Y}_{\mathrm{P}} \cup (\cap_{i\in\mathrm{CH}}\mathbb{Y}_{\mathrm{CH},i})$;
5: **case** $\vee$
6: replace its parent with $\mathbb{Y}_{\mathrm{P}} \cup (\cup_{i\in\mathrm{CH}}\mathbb{Y}_{\mathrm{CH},i})$;
7: **end for**
8: $\triangleright$ *for each Boolean operator node,* $\mathbb{Y}_{\mathrm{P}}$ *denotes its parent,* CH *collects its children, and* $\mathbb{Y}_{\mathrm{CH},i}$ *is the corresponding control set for each child;*
9: **return** the root node.

Remark 5.5. *Note that to implement Algorithm 5.3, we do not need to first check for the existence of a policy for the controlled TLT. The fact that a nonempty control set is synthesized by Algorithm 5.3 at each time step is necessary for the existence of the policy for the controlled TLT. We use the existence of the policy as a-priori condition for proving the recursive feasibility of Algorithm 5.3 in Theorem 5.5.*

Example 5.6. *Let us continue to consider the controlled transition system in Example 2.2 and the LTL formula* $\varphi = \diamondsuit\square o_2$ *in Example 5.5. Implementing Algorithm 5.3, we obtain Table 5.1. We can see that at each time step, we can synthesize a nonempty feedback control set. One realization is* $s_1 \xrightarrow{a_1} s_3 \xrightarrow{a_2} s_3 \xrightarrow{a_1} s_2 \xrightarrow{a_1} s_3 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_4 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_4 \cdots$, *of which the trajectory* $\boldsymbol{p} = s_1s_3s_3s_2s_3s_2(s_4s_2)^{\omega}$ *satisfies both the controlled TLT and the formula* φ.

In this example, Algorithm 5.3 is recursively feasible since we can verify that the condition in Theorem 5.6 holds. That is, there exists a policy such that all the resulting trajectories satisfy the controlled TLT: a feasible stationary policy is $\boldsymbol{\mu} = \bar{\mu}\bar{\mu}\cdots$, *where* $\bar{\mu} : \mathbb{S} \to \mathbb{U}$ *with* $\bar{\mu}(s_1) = a_1, \bar{\mu}(s_2) = a_2, \bar{\mu}(s_3) = a_1$, *and* $\bar{\mu}(s_4) = a_1$. *Under this policy, there are two possible trajectories,* $\boldsymbol{p} = s_1s_3(s_2s_4)^{\omega}$ *and* $\boldsymbol{p} = s_1(s_2s_4)^{\omega}$, *both of which satisfy the controlled TLT and the LTL formula* φ.

Table 5.1: Online implementation under Algorithm 5.3

Time k	State x_k	Control set $\mathbb{U}_k^{\varphi}(x_k)$	Control input u_k
0	s_1	$\{a_1\}$	a_1
1	s_3	$\{a_1, a_2\}$	a_2
2	s_3	$\{a_1, a_2\}$	a_1
3	s_2	$\{a_1, a_2\}$	a_1
4	s_3	$\{a_1, a_2\}$	a_1
5	s_2	$\{a_1, a_2\}$	a_2
6	s_4	$\{a_1\}$	a_1
7	s_2	$\{a_1, a_2\}$	a_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$

5.5 Numerical Evaluations

5.5.1 Mobile Robot Example

Following the same scenario as in [162], as shown in Figure 5.3, we consider a mobile robot modeled as a double integrator:

$$x_{k+1} = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} u_k + w_k. \tag{5.1}$$

Different from [162], we choose the smaller sampling time of 0.2 second and take into account the disturbance w_k. The working space is $\mathbb{X} = \{z \in \mathbb{R}^2 \mid [-10, -10]^T \leq z \leq [2, 2]^T\}$, the control constraint set is $\mathbb{U} = \{z \in \mathbb{R} \mid -2 \leq z \leq 2\}$, and the disturbance set is $\mathbb{W} = \{z \in \mathbb{R}^2 \mid [-0.05, -0.05]^T \leq z \leq [0.05, 0.05]^T\}$. In Figure 5.3(a), the obstacle regions are $\mathbb{O}_1 = \{z \in \mathbb{R}^2 \mid [-10, -10]^T \leq z \leq [-5, -5]^T\}$ and $\mathbb{O}_2 = \{z \in \mathbb{R}^2 \mid [-5, -4]^T \leq z \leq [2, -3]^T\}$, the target region is $\mathbb{T} = \{z \in \mathbb{R}^2 \mid [-0.5, -0.5]^T \leq z \leq [-0.5, -0.5]^T\}$, and two visiting regions are $\mathbb{A} = \{z \in \mathbb{R}^2 \mid [-6, 1]^T \leq z \leq [-5, 2]^T\}$ and $\mathbb{B} = \{z \in \mathbb{R}^2 \mid [-5, -3]^T \leq z \leq [-4, -2]^T\}$.

Recall the system CS (2.1). Let the set of the observations be

$$\mathcal{O} = \{a_1, a_2, a_3, a_4, a_5, a_6\}$$

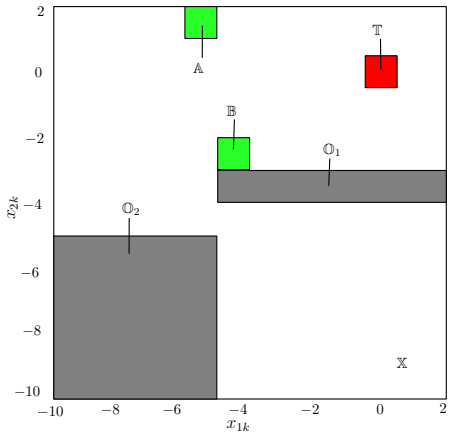


Figure 5.3: Scenario of mobile robot example.

and, if $x \in \mathbb{X}$, we define the observation function as

$$g(x) = \begin{cases} \{a_1, a_2\}, & \text{if } x \in \mathbb{X} \cap \mathbb{O}_1, \\ \{a_1, a_3\}, & \text{if } x \in \mathbb{X} \cap \mathbb{O}_2, \\ \{a_1, a_4\}, & \text{if } x \in \mathbb{X} \cap \mathbb{A}, \\ \{a_1, a_5\}, & \text{if } x \in \mathbb{X} \cap \mathbb{B}, \\ \{a_1, a_6\}, & \text{if } x \in \mathbb{X} \cap \mathbb{T} \\ \{a_1\}, & \text{otherwise.} \end{cases} \tag{5.2}$$

As shown in Remark 2.2, we can rewrite the system (5.1) with the observation function (5.2) as a controlled transition system with the set of atomic propositions $\mathcal{AP} = \mathcal{O}$ and the labelling function $L = g$.

In [162], the specification is to visit the region $\mathbb{A}$ or region $\mathbb{B}$, and then the target region $\mathbb{T}$, while always avoiding obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$, and staying inside the working space $\mathbb{X}$. This specification can be expressed as a co-safe LTL formula $\varphi' = ((a_1 \wedge \neg a_2 \wedge \neg a_3)\mathsf{U}a_6) \wedge (\neg a_6 \mathsf{U}(a_4 \vee a_5))$. Here, we extend the specification to be to visit region $\mathbb{A}$ or region $\mathbb{B}$, and then visit and *always* stay inside the target region $\mathbb{T}$, while always avoiding obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$, and staying inside working space $\mathbb{X}$. Obviously, this specification cannot be expressed as a co-safe LTL formula, and thus cannot be handled by the approach in [162]. We instead express this specification as the LTL formula $\varphi = ((a_1 \wedge \neg a_2 \wedge \neg a_3)\mathsf{U}\Box a_6) \wedge (\neg a_6 \mathsf{U}(a_4 \vee a_5))$. We will show that our

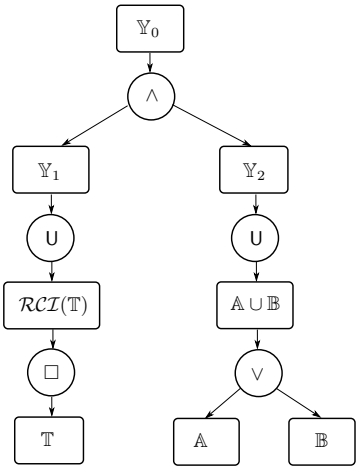


Figure 5.4: The controlled TLT for the LTL formula $\varphi = ((a_1 \wedge \neg a_2 \wedge \neg a_3)\mathsf{U}\Box a_6) \wedge (\neg a_6 \mathsf{U}(a_4 \vee a_5))$ in the mobile robot example, where $\mathbb{Y}_1 = \mathbb{Y}_1 \cap \mathbb{Y}_2$, $\mathbb{Y}_1 = \mathcal{R}^{\mathrm{c}}(\mathbb{X} \setminus (\mathbb{O}_1 \cup \mathbb{O}_2), \mathcal{RCI}(\mathbb{T}))$, and $\mathbb{Y}_2 = \mathcal{R}^{\mathrm{c}}(\mathbb{X} \setminus \mathbb{T}, \mathbb{A} \cup \mathbb{B})$.

approach can handle such non-co-safe LTL formula. By computing inner approximations of the controlled reachable sets, we can construct the controlled TLT of φ and then use Algorithm 5.3 to synthesize controllers online. The constructed controlled TLT for φ is shown in Figure 5.4. Similar to [162], we choose three different initial states, for each of which the state trajectories and the control trajectories are shown in Figures 5.5–5.7. We can see that in Figure 5.5(a), Figure 5.6(a), and Figure 5.7(a), all state trajectories x_k satisfy the required specification φ. The black dots are the initial state. In this example, the target region $\mathbb{T}$ is a RCIS. After entering $\mathbb{T}$, the states stay there by using the controllers that ensure robust invariance. In Figure 5.5(b), Figure 5.6(b), and Figure 5.7(b)„ the dashed lines denote the control bounds, the cyan regions represent the synthesized control sets $\mathbb{U}_k^{\varphi}$ in Algorithm 5.3, and the blue lines are the implemented control inputs x_k.

5.5.2 Automated Car Example

This example will show how the specification can be updated online by using our approach. As shown in Figure 5.8, we consider a scenario where an automated car plans to move to a target set $\mathbb{T}$ but with some unknown obstacles on the road. The sensing region of the car is limited. We use a single integrator

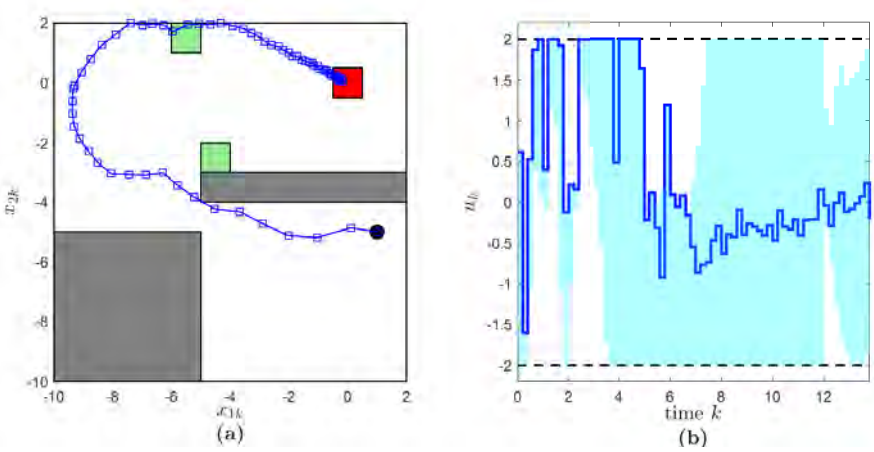


Figure 5.5: One simulated trajectory starting from the initial states $[1, -5]^T$ in the mobile robot example: (a) state x_k; (b) control input u_k and control set $\mathbb{U}_k^{\varphi}$.

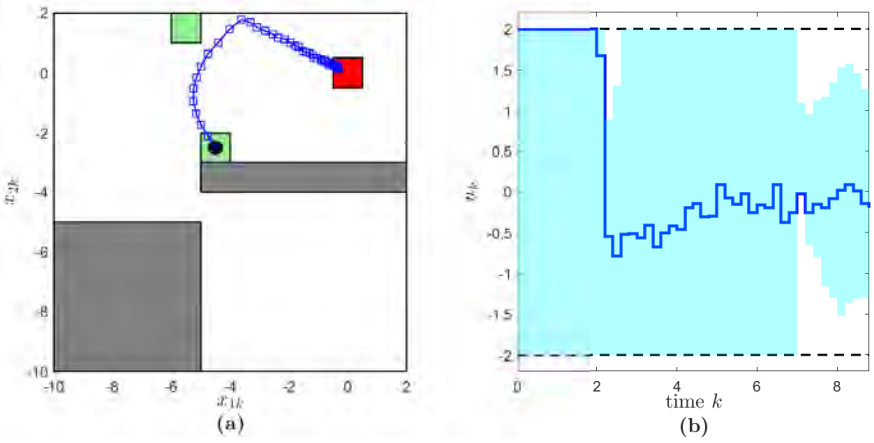


Figure 5.6: One simulated trajectory starting from the initial states $[-4.5, -2.5]^T$ in the mobile robot example: (a) state x_k; (b) control input u_k and control set $\mathbb{U}_k^{\varphi}$.

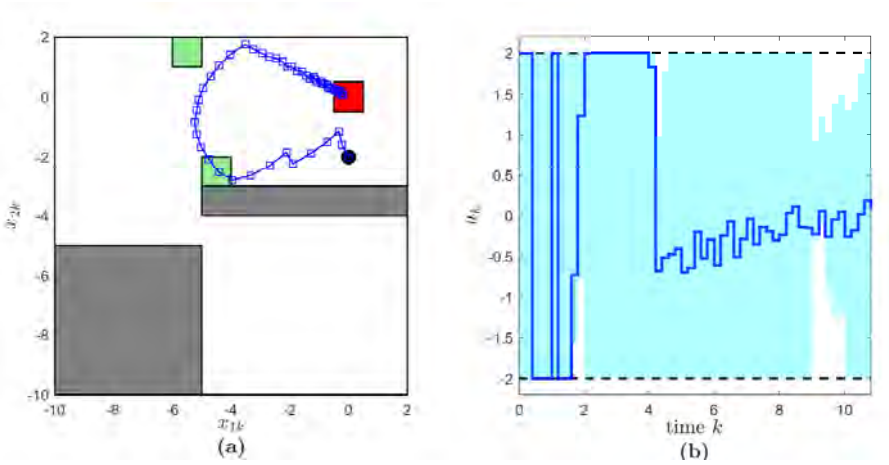


Figure 5.7: One simulated trajectory starting from the initial states $[0, -2]^T$ in the mobile robot example: (a) state x_k; (b) control input u_k and control set $\mathbb{U}_k^{\varphi}$.

model with a sample period of 1 second to model the dynamics of the car:

$$x_{k+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u_k + w_k, \tag{5.3}$$

where $x_k = [p_k^x \; p_k^y]^T$ is the position and $u_k = [v_k^x \; v_k^y]^T$ velocity. The working space is $\mathbb{X} = \{z \in \mathbb{R}^2 \mid [0, -5]^T \leq z \leq [150, 5]^T\}$, the control constraint set is $\mathbb{U} = \{z \in \mathbb{R}^2 \mid [-2, -0.5]^T \leq z \leq [2, 0.5]^T\}$, the disturbance set is $\mathbb{W} = \{z \in \mathbb{R}^2 \mid [-0.1, -0.1]^T \leq z \leq [0.1, 0.1]^T\}$, and the target region is $\mathbb{T} = \{z \in \mathbb{R}^2 \mid [145, -5]^T \leq z \leq [150, 0]^T\}$. We assume that $\mathbb{X}$, $\mathbb{U}$, and $\mathbb{W}$ are known *a priori* to the car and the car should move along the lane with the right direction unless lane change is necessary. In Figure 5.8, there are two broken cars in the sets $\mathbb{O}_1 = \{z \in \mathbb{R}^2 \mid [40, -5]^T \leq z \leq [45, 0]^T\}$ and $\mathbb{O}_2 = \{z \in \mathbb{R}^2 \mid [100, -5]^T \leq z \leq [105, 0]^T\}$. We assume that $\mathbb{O}_1$ and $\mathbb{O}_2$ are unknown to the car at the beginning. As long as the car can sense them, they are known to the car.

Let the initial state be $x_0 = [0.5, -2.5]^T$ and the sensing limitation is 15. At time step $k = 0$, the set of observations is $\mathcal{O} = \{a_1, a_2\}$ and if $x \in \mathbb{X}$, we define the observation function as

$$g(x) = \begin{cases} \{a_1, a_2\}, & \text{if } x \in \mathbb{X} \cap \mathbb{T}, \\ \{a_1\}, & \text{otherwise.} \end{cases}$$

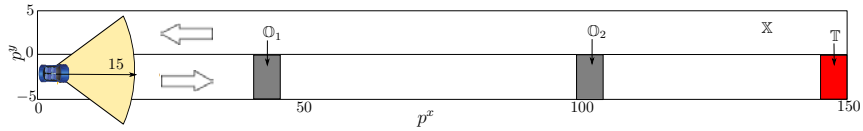


Figure 5.8: Scenario illustration of automated car example: an automated car plans to reach a target set $\mathbb{T}$ but with some unknown obstacles on the road.

The initial specification can be expressed as an LTL $\varphi = a_1 \mathsf{U} a_2$. By constructing the controlled TLT of φ shown in Figure 5.9 and implementing Algorithm 5.3, we obtain one realization as shown in Figure 5.10. We can see that the car keeps moving straightforward until it senses the obstacle $\mathbb{O}_1$ at $[25.5, -2.4]^T$.

When the car can sense $\mathbb{O}_1$, a new observation a_3 with $a_3 \neq a_1$ and $a_3 \neq a_2$ is added to the set $\mathcal{O}$, which becomes $\mathcal{O} = \{a_1, a_2, a_3\}$. If $x \in \mathbb{X}$, we update the observation function as

$$g(x) = \begin{cases} \{a_1, a_2\}, & \text{if } x \in \mathbb{X} \cap \mathbb{T}, \\ \{a_1, a_3\}, & \text{if } x \in \mathbb{X} \cap \mathbb{O}_1, \\ \{a_1\}, & \text{otherwise.} \end{cases}$$

To avoid $\mathbb{O}_1$, the new specification is changed to be $\varphi' = \varphi \wedge (\square \neg a_3)$. We can construct the TLT of φ' based on that of φ, which is shown in Figure 5.9, and then continue to implement Algorithm 5.3. We can see that the car changes lane from $[25.5, -2.4]^T$ and quickly merges back after overtaking $\mathbb{O}_1$. The trajectories are shown in Figure 5.10. The vehicle is under the control with respect to φ' until it can sense $\mathbb{O}_2$ at $[86.3, -2.5]^T$.

Similarly, when the car can sense $\mathbb{O}_2$, we update $\mathcal{O} = \{a_1, a_2, a_3, a_4\}$ and the observation function as if $x \in \mathbb{X}$,

$$g(x) = \begin{cases} \{a_1, a_2\}, & \text{if } x \in \mathbb{X} \cap \mathbb{T}, \\ \{a_1, a_3\}, & \text{if } x \in \mathbb{X} \cap \mathbb{O}_1, \\ \{a_1, a_4\}, & \text{if } x \in \mathbb{X} \cap \mathbb{O}_2, \\ \{a_1\}, & \text{otherwise.} \end{cases}$$

To avoid $\mathbb{O}_2$, the new specification is changed to be $\varphi'' = \varphi' \wedge (\square \neg a_4)$. We can construct the TLT of φ'' based on that of φ', which is shown in Figure 5.9, and then continue to implement Algorithm 5.3. We can see that the car changes

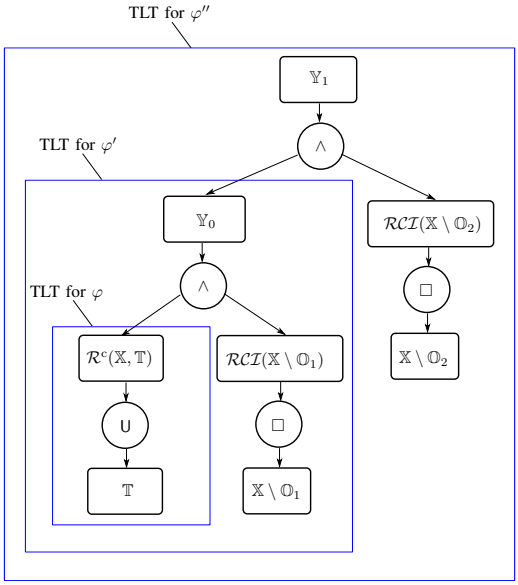


Figure 5.9: The controlled TLT for the LTL formulae in the automated car example, where $\varphi = a_1 \mathsf{U} a_2$, $\varphi' = \varphi \wedge (\square \neg a_3)$, $\varphi'' = \varphi' \wedge (\square \neg a_4)$, $\mathbb{Y}_0 = \mathcal{R}^c(\mathbb{X}, \mathbb{T}) \cap \mathcal{RCI}(\mathbb{X} \setminus \mathbb{O}_1)$, and $\mathbb{Y}_1 = \mathcal{R}^c(\mathbb{X}, \mathbb{T}) \cap \mathcal{RCI}(\mathbb{X} \setminus \mathbb{O}_1) \cap \mathcal{RCI}(\mathbb{X} \setminus \mathbb{O}_2)$.

lane from $[86.3, -2.5]^T$ and quickly merges back after overtaking $\mathbb{O}_2$. Under the control with respect to φ'', the car finally reaches the target set $\mathbb{T}$.

Figure 5.10 (a) shows the state trajectories, from which we can see that the whole specification is completed. Figures 5.10 (b)–(c) show the corresponding control inputs, where the dashed lines denote the control bounds. The cyan regions represent the synthesized control sets and the blue lines are the control trajectories. Furthermore, we repeat the above process for 100 realizations of the disturbance trajectories. The state trajectories for such 100 realizations are shown in Figure 5.11.

We remark that in this example, the control inputs are chosen to push the state to move down along the TLT as fast as possible. More specifically, if the state x_k is the i-step reachable set in the set node $\mathcal{R}^c(\mathbb{X}, \mathbb{T})$, we can generate a smaller control set from which the control input can push the state to the $(i-1)$-step reachable set. That is what we can see from Figure 5.10, where almost all control inputs in the synthesized control sets along x-axis are positive.

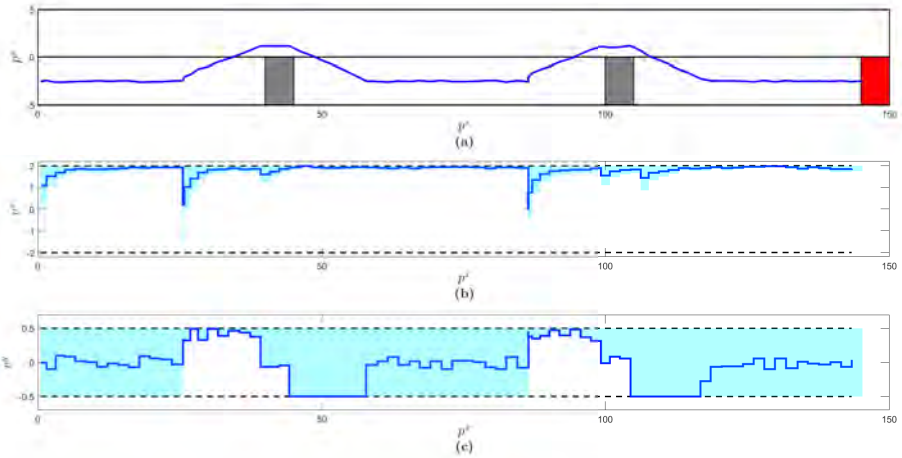


Figure 5.10: One simulated trajectory for 1 realization of disturbance in the automated car example: (a) position $x = [p^x \; p^y]^T$; (b) control input of x-axis, i.e., the velocity v^x; (c) control input of y-axis, i.e., the velocity v^y.

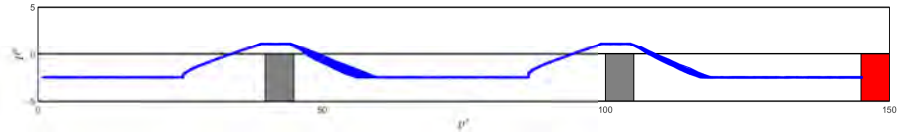


Figure 5.11: Position trajectories for 100 realizations of disturbance in the automated car example.

5.6 Summary

In this chapter, we studied LTL model checking and control synthesis for discrete-time uncertain systems. Quite unlike automaton-based methods, our solutions build on the connection between LTL formulae and TLT structures via reachability analysis. For a transition system and an LTL formula, we proved that the TLT provide an underapproximation (or overapproximation) for the LTL via minimal (or maximal) reachability analysis. We provided sufficient conditions and necessary conditions to the model checking problem. For a controlled transition system and an LTL formula, we showed that the TLT is an underapproximation for the LTL formula and thereby proposed an online control synthesis algorithm, under which a set of feasible control inputs is generated at each time step. We proved that this algorithm is recursively feasible. We also illustrated the effectiveness of the proposed methods through several examples.

Appendix. Proof of Theorem 5.1

The whole proof is divided into two parts: the first part shows how to construct a TLT from the formula $\forall\varphi$ by means of the reachability operators $\mathcal{R}^{\mathrm{m}}$ and $\mathcal{RI}$, while the second part shows that such TLT is an underapproximation for φ.

Construction: We follow three steps to construct a TLT.

Step 1: rewrite the given LTL in the weak-until positive normal form. From [36], each LTL formula has an equivalent LTL formula in the weak-until positive normal form, which can be inductively defined as

$$\begin{aligned}\varphi ::= &\ \mathrm{true} \mid \mathrm{false} \mid a \mid \neg a \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \\ &\mid \varphi_1 \mathsf{U} \varphi_2 \mid \varphi_1 \mathsf{W} \varphi_2. \end{aligned} \tag{5.4}$$

Step 2: for each atomic proposition $a \in \mathcal{AP}$, construct the TLT with only a single set node from $\forall a$ or $\forall\neg a$. More specifically, the set node for $\forall a$ is $L^{-1}(a) = \{x \in \mathbb{S} \mid a \in L(x)\}$ while the set node for $\forall\neg a$ is $\mathbb{S} \setminus L^{-1}(a)$. In addition, the TLT for $\forall\mathrm{true}$ (or $\forall\mathrm{false}$) also has a single set node, which is $\mathbb{S}$ (or $\emptyset$).

Step 3: based on Step 2, follow the induction rule to construct the TLT for any LTL formula in the weak-until positive normal form. More specifically, we will

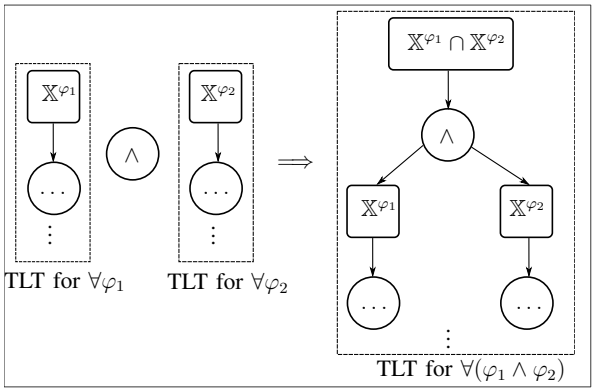


Figure 5.12: The TLT construction of $\forall(\varphi_1 \wedge \varphi_2)$. Here, the circles denote the operator nodes and the rectangles denote the set nodes. Similar for Figures 5.13–5.16.

show that given the LTL formulae φ, φ_1, and φ_2 in the weak-until positive normal form, if the TLT can be constructed from $\forall\varphi$, $\forall\varphi_1$, and $\forall\varphi_2$, respectively, then the TLT can be thereby constructed from the formulae $\forall(\varphi_1 \wedge \varphi_2)$, $\forall(\varphi_1 \vee \varphi_2)$, $\forall \bigcirc \varphi$, $\forall(\varphi_1 \mathsf{U} \varphi_2)$, and $\forall(\varphi_1 \mathsf{W} \varphi_2)$, respectively.

For $\forall(\varphi_1 \wedge \varphi_2)$ (or $\forall(\varphi_1 \vee \varphi_2)$), we construct the TLT by connecting the root nodes of the TLT for $\forall\varphi_1$ and $\forall\varphi_2$ through the operator $\wedge$ (or $\vee$) and taking the intersection (or union) of two root nodes, as shown in Figures 5.12-5.13. For $\forall \bigcirc \varphi$, we denote by $\mathbb{X}^{\varphi}$ the root node of the TLT for $\forall\varphi$ and then construct the TLT by adding a new set node $\mathcal{R}^{\mathrm{m}}(\mathbb{S}, \mathbb{X}^{\varphi}, 1)$ to be the parent of $\mathbb{X}^{\varphi}$ and connecting them through the operator $\bigcirc$, as shown in Figure 5.14.

For $\forall(\varphi_1 \mathsf{U} \varphi_2)$, the TLT construction is as follows. Denote by $\{(\mathbb{Y}_i^{\varphi_1}, \mathcal{O}_i^{\varphi_1})\}_{i=1}^{N^{\varphi_1}}$ all the pairs comprising a leaf node and its corresponding parent in the TLT of $\forall\varphi_1$, where N^{φ_1} is the number of the leaf nodes. Here, $\mathbb{Y}_i^{\varphi_1}$ is the ith leaf node and $\mathcal{O}_i^{\varphi_1}$ is its parent. Denote by $\mathbb{X}^{\varphi_2}$ the root node of TLT for $\forall\varphi_2$. We first change each leaf node $\mathbb{Y}_i^{\varphi_1}$ to $\mathcal{R}^{\mathrm{m}}(\mathbb{Y}_i^{\varphi_1}, \mathbb{X}^{\varphi_2}) \setminus \mathbb{X}^{\varphi_2}$. We then update the new tree for $\forall\varphi_1$ from the leaf node to the root node according to the definition of the operators. After that, we take N^{φ_1} copies of the TLT of φ_2. We set the root node of each copy as the child of each new leaf node, respectively, and connect them trough the operator U. Finally, we have one more copy of the TLT of $\forall\varphi_2$ and connect this copy and the new tree trough the disjunction $\vee$. An illustrative diagram is given in Figure 5.15.

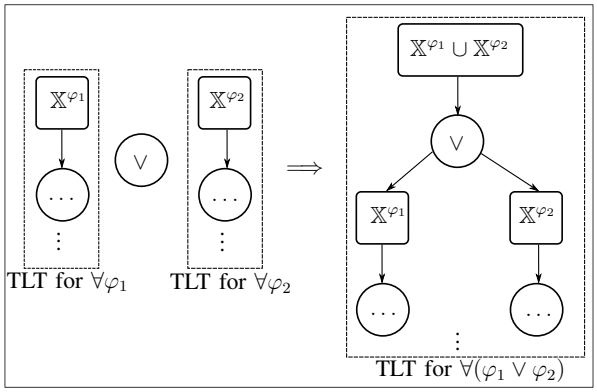


Figure 5.13: The TLT construction of $\forall(\varphi_1 \vee \varphi_2)$.

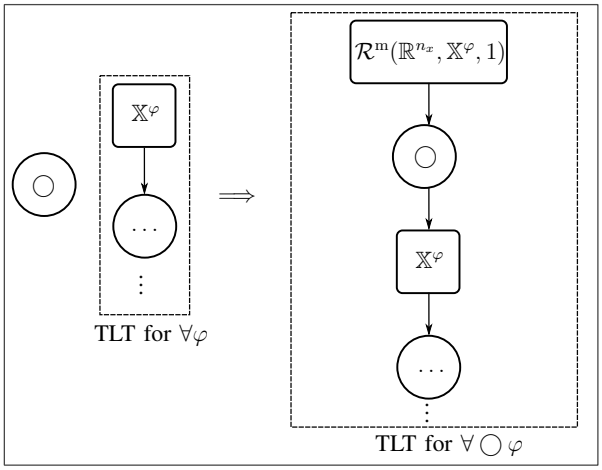


Figure 5.14: The TLT construction of $\forall \bigcirc \varphi$.

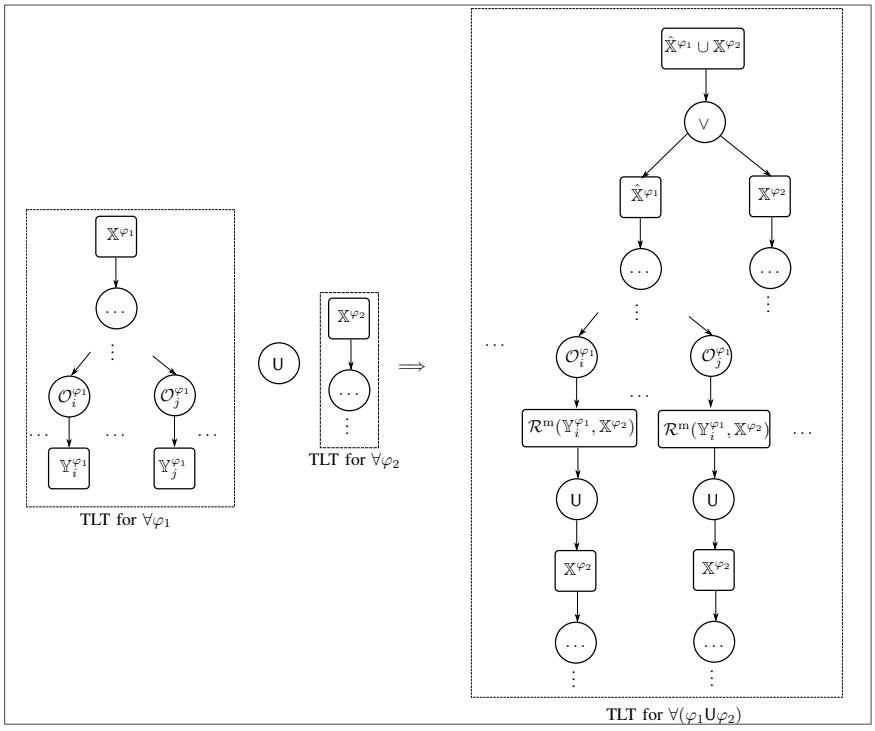


Figure 5.15: The TLT construction for $\forall(\varphi_1 \mathsf{U} \varphi_2)$.

For the fragment $\forall(\varphi_1 \mathsf{W} \varphi_2)$, we first recall that $\varphi_1 \mathsf{W} \varphi_2 = \varphi_1 \mathsf{U} \varphi_2 \vee \Box \varphi_1$. Let $\varphi' = \varphi_1 \mathsf{U} \varphi_2$ and $\varphi'' = \Box \varphi_1$. Denote by $\mathbb{X}^{\varphi_1}$ the root node of the TLT for $\forall\varphi_1$. We first construct the TLT of $\forall\varphi'$ as described above. Second, we further construct the TLT of $\forall\varphi''$ with by adding a new node $\mathcal{RI}(\mathbb{X}^{\varphi_1})$ as the parent of $\mathbb{X}^{\varphi_1}$ and connecting them through $\Box$. Then, we construct the TLT of $\forall(\varphi' \vee \varphi'')$. An illustrative diagram is given in Figure 5.16.

Underapproximation: First, it is very easy to verify that the constructed TLT above with a single set node $L^{-1}(a)$ (or $\mathbb{S} \setminus L^{-1}(a)$ or) for $\forall a$ (or $\forall \neg a$ or $\mathbb{S}$ or $\emptyset$) is an underapproximation for $a \in \mathcal{AP}$ (or $\neg a$ or $\forall \mathrm{true}$ or $\forall \mathrm{false}$) and the underapproximation relation in these cases is also tight.

Next we also follow the induction rule to show that the constructed TLT from $\forall\varphi$ is an underapproximation for φ. Consider LTL formulae φ, φ_1, and φ_2. We will show that if the constructed TLT of $\forall\varphi$, $\forall\varphi_1$, and $\forall\varphi_2$ are the underapproximations of φ, φ_1, and φ_2, respectively, then the TLT constructed

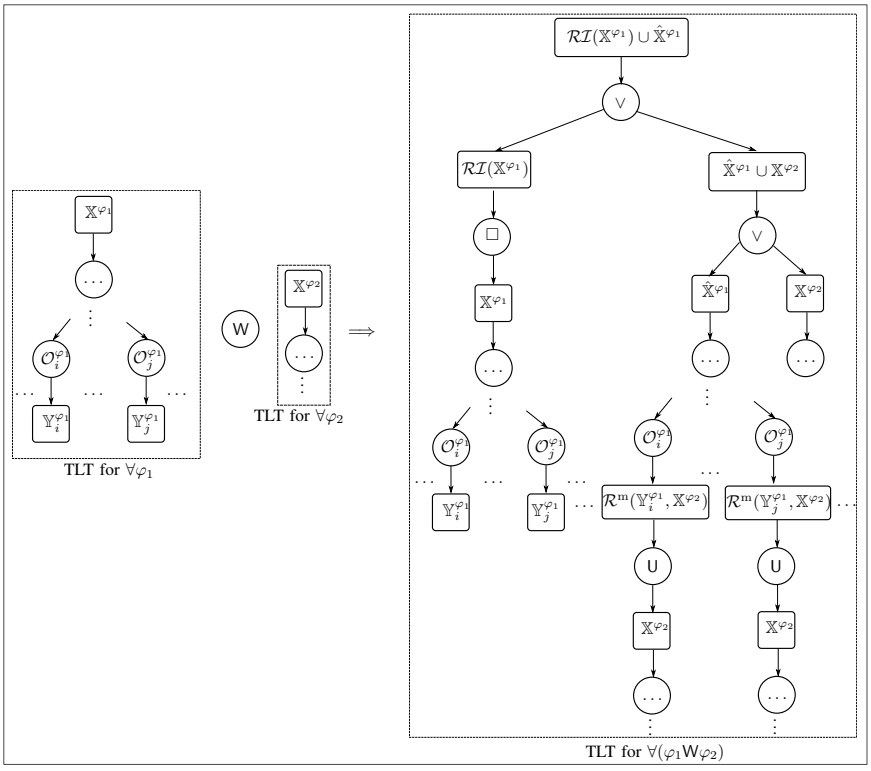


Figure 5.16: The TLT construction for $\forall(\varphi_1 \mathsf{W} \varphi_2)$.

above for the formulae $\forall(\varphi_1 \wedge \varphi_2)$, $\forall(\varphi_1 \vee \varphi_2)$, $\forall \bigcirc \varphi$, $\forall(\varphi_1 \mathsf{U} \varphi_2)$, and $\forall(\varphi_1 \mathsf{W} \varphi_2)$ are the underapproximations of $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\bigcirc\varphi$, $\varphi_1 \mathsf{U} \varphi_2$, and $\varphi_1 \mathsf{W} \varphi_2$, respectively.

According to the set operation (intersection or union) or the definition of one-step minimal reachable set, it is easy to verify that the constructed TLT for $\forall(\varphi_1 \wedge \varphi_2)$ (or $\forall(\varphi_1 \vee \varphi_2)$) or $\forall \bigcirc \varphi$ is an underapproximation for $\varphi_1 \wedge \varphi_2$ (or $\varphi_1 \vee \varphi_2$) or $\bigcirc\varphi$ if the TLT of $\forall\varphi_1$ and $\forall\varphi_2$, and $\forall\varphi$ are underapproximations φ, φ_1, and φ_2, respectively.

Let us consider $\varphi_1 \mathsf{U} \varphi_2$. Assume that a trajectory $\boldsymbol{p}$ satisfies the TLT of $\forall(\varphi_1 \mathsf{U} \varphi_2)$. Recall the construction of the TLT of $\forall(\varphi_1 \mathsf{U} \varphi_2)$ from $\forall\varphi_1$ and $\forall\varphi_2$. According to the definition of minimal reachable set, we have (1) $\boldsymbol{p}$ satisfies the TLT of $\forall\varphi_2$; *or* (2) there exists that $j \in \mathbb{N}$ such that $\boldsymbol{p}[j..]$ satisfies the TLT of $\forall\varphi_2$ and for all $i \in \mathbb{N}_{[0,j-1]}$, the trajectory $p[i..]$ satisfies the the TLT of $\forall\varphi_1$. Under the assumption that the TLT of $\forall\varphi_1$ and $\forall\varphi_2$ are the underapproximations of φ_1 and φ_2, respectively, we have that there exists $j \in \mathbb{N}$ such that $\boldsymbol{p}[j..] \vDash \varphi_2$ and for all $i \in \mathbb{N}_{[0,j-1]}$, $p[i..] \vDash \varphi_1$, which implies that $\boldsymbol{p} \vDash \varphi_1 \mathsf{U} \varphi_2$. Thus, the TLT of $\forall(\varphi_1 \mathsf{U} \varphi_2)$ is an approximation of $\varphi_1 \mathsf{U} \varphi_2$.

Recall that $\varphi_1 \mathsf{W} \varphi_2 = \varphi_1 \mathsf{U} \varphi_2 \vee \Box\varphi_1$. Following the proofs for until operator U and the disjunction $\vee$ and the definition of the robust invariant set, it yields that the constructed TLT of $\forall(\varphi_1 \mathsf{W} \varphi_2)$ is an underapproximation of $\varphi_1 \mathsf{W} \varphi_2$.

The proof is completed. □

Chapter 6

Car Parking Application

In the previous chapter, we proposed the notion of temporal logic trees (TLT) and developed TLT-based approaches for performing model checking and control synthesis for uncertain systems. In this chapter, we will show that such approaches have great potential in shared-autonomy systems where the human actions are used proactively. Specifically, we propose a guiding controller solution to a car parking problem, where a human operator needs to remotely drive a vehicle into an empty parking spot. We specify the parking task as a set of linear temporal logic (LTL) formulae. Then, using the TLT-based approach, we synthesize a set of controllers for assisting the human operator to complete the mission, while guaranteeing that the system maintains specified spatial and temporal properties. We assume the human operator's exact preference of how to complete the mission is unknown. Instead, we use a data-driven approach to infer and update the human operator's preference over parking spots in real-time. If, while the human is operating the vehicle, she provides inputs that violate any of the invariances prescribed by the LTL formula, our verification-based controller will use its internal belief of the human operator's intended objective to guide the operator back on track. Moreover, we show that as long as the specifications are initially feasible, our controller will stay feasible and can guide the human to park the vehicle in the empty spot safely despite unpredicted human actions. We demonstrate the results on the Small Vehicles for Autonomy (SVEA) platform.

The remainder of this chapter is organized as follows. Section 6.1 gives the background. In Section 6.2, we outline our plant model and provide the problem statement. In Section 6.3, we formulate the guiding controller. In Section 6.4, we illustrate the effectiveness of our approach with an experiment. In Section 6.5, we conclude the chapter.

6.1 Introduction

Vehicle parking is one of the most time-consuming and safety-critical tasks that drivers perform daily. According to statistics reported in [163], drivers in the U.S., U.K. and Germany wasted 17, 44, and 41 hours a year, respectively; in total costing 72.7 billion dollars, 23.3 billion pounds, and 40.4 billion euros a year. In addition to being an unsustainable (due to wasted fuel) and economically wasteful task, vehicle parking is also often a dangerous and straining task for drivers that involves complex maneuvering into tight spaces with many chaotic safety constraints. Motivated by these issues, engineers and researchers have recently spent significant effort in advancing parking guidance and management technology.

Vehicle parking-related industries have already made a variety of technological advances to parking guidance and management systems. For example, nowadays, many private vehicles have parking assistance systems that are capable of assisting humans in maneuvering into nearby parking spots. In the development of better parking management systems, Hikvision, a supplier of video surveillance products, demonstrated that vision-based guidance methods can improve parking efficiency [164]. Furthermore, there are now even deployments of commercial automated parking products, e.g., Remote Parking Assist from Mercedes [165] and Remote Control Parking from BMW [166].

In the literature, researchers devote most of their effort into solving the problem of generating obstacle-free trajectories for vehicles in tight parking environments. Nonlinear vehicle dynamics and non-convex environments make this problem difficult to solve in real time. In [167], authors propose a hierarchical framework that integrates a Hybrid A* planner with an optimization-based collision avoidance planner. The Hybrid A* planner provides a coarse warm-start solution for the collision avoidance planner, speeding up the real-time implementation. A distributed model predictive control formulation improves parking efficiency and helps ensure collision avoidance by taking into account human behavior prediction and vehicle coordination in [168]. In [169], authors decouple an automated parking problem into a centralized parking spot allocation and path generation problem, and a decentralized collision avoidance control problem. Although these approaches yield important results, the formulations are not suitable for checking whether the parking task is feasible in the first place, since checking the feasibility of a non-convex optimization problem is challenging.

Additionally, to the extent of the authors' knowledge, shared autonomy-based parking receives little attention in the research community, despite the fact that many of the current parking technologies rely on the presence of a human supervisor to perform the parking maneuver [165], [166]. The requirement of a human supervisor partially stems from the unpredictable and chaotic nature of parking lots and other human-driven vehicles. For similar reasons, many have proposed the use of remote human operators as a layer of operations management and exception-handling for connected vehicles in general [170].

6.1.1 Contribution

The main contribution of this chapter is to propose a guiding controller that supports a remote human operator to safely drive a vehicle to an empty parking spot. The parking task is specified as a set of LTL formula. The TLT-based approach in Chapter 5 is used to do control set synthesis using reachability analysis, giving us guarantees that the system will follow the LTL specifications. These control sets tells us what a human operator is allowed and not allowed to do. Furthermore, we develop a data-driven approach to infer and update the human operator's preference over parking spots in real-time. Then, with the verified control sets, we improve the approach in [171] by allowing the human to freely make decisions as long as they do not violate invariances specified by the LTL formula.

6.1.2 Related Work

As shown by [172] and [173], using LTL formula allows us to conveniently express time-related tasks, e.g., car parking, for automated systems. Furthermore, the work presented in [171] exemplifies the advantage of using temporal tasks for human-in-the-loop mixed-initiative control. However, with LTL specified missions, [35], [68], [174], [175] show that synthesizing controls that guarantee that a specification will be met is nontrivial. [161] details a correspondence between reachable sets and signal temporal logic (STL) that allows for control synthesis directly from STL specifications with guarantees that the controller will satisfy the invariances given by the STL formula. Inspired from this, we propose a TLT-based approach for synthesizing control sets from LTL specifications.

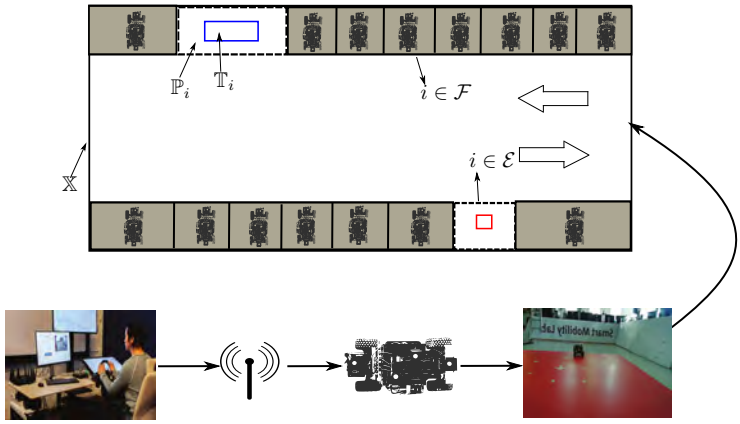


Figure 6.1: Illustration of our parking scenario.

There are several proposals for how to design guiding controllers. In [176], the authors propose an approach to learn optimal policies via reinforcement learning while enforcing LTL specifications. They utilize a shield, a similar notion to the guiding controller in our chapter, to monitor the actions from the learner. The shield is used to correct them only if the chosen action causes a violation of the specification. We remark that the systems studied in [176] are finite-transition systems, whereas in our work we consider discrete-time dynamical systems, leading to different control synthesis approaches. Another notable approach is given in [155], which proposes one of the first frameworks where humans are given a higher priority than the automated system in the decision making process whereas the human's direct control of the automated system is "weakened". The designed controller provides a set of admissible control inputs with enough degrees of freedom to allow the human operator to easily complete her task. We take inspiration from this approach for the design of our guiding controller.

6.2 Problem Formulation

In this section, we formulate the remote car parking problem. Before that, let us first detail the remote parking scenario, as shown in Figure 6.1, where a remote operator is safely driving a vehicle to some empty parking spot, while avoiding collision with the walls of the parking lot and the parked vehicles.

6.2.1 Vehicle Model

For simplicity, the vehicle is modeled as a two-dimensional single-integrator affected by a bounded disturbance. After discretizing the model with a sampling period of δ second, it follows that

$$x_{k+1} = Ax_k + Bu_k + w_k, \tag{6.1}$$

where $x_k = \left[p_k^x, p_k^y\right]^T$, $u_k = \left[v_k^x, v_k^y\right]^T$, p_k^x and p_k^y, v_k^x and v_k^y denote the longitudinal and lateral position and velocity, respectively. The control input u_k is bounded by $\mathbb{U} \subset \mathbb{R}^2$ and the disturbance w_k is bounded by $\mathbb{W} \subset \mathbb{R}^2$.

6.2.2 LTL-Specified Parking Task

To define the parking task, the whole space of the parking lot is denoted by the set $\mathbb{X} \subseteq \mathbb{R}^2$. In the parking lot, each parking spot is either full or empty. Denote the sets of indices corresponding to full and empty parking spots as $\mathcal{F}$ and $\mathcal{E}$, respectively. As in Figure 6.1, the state set of the parking spot i is denoted by $\mathbb{P}_i \subseteq \mathbb{R}^2$. We further introduce the set of states within every $\mathbb{P}_i$ that correspond to an accurate and correct parking job as $\mathbb{T}_i$. In our example, we define accurate parking as parking in a precise location. Then, we can describe the parking task as: if there is one or more empty parking spots, then park in one of them while staying safe.

To specify the parking task using LTL formulae, we denote by $[\mathbb{Y}]$ the corresponding atomic proposition of the set $\mathbb{Y}$. Then, the parking task described above for an empty parking spot i can be written as

$$\varphi_i = \left([\mathbb{X}] \wedge (\wedge_{j\in\mathcal{F}} \neg[\mathbb{P}_j])\right) \mathsf{U} [\mathbb{P}_i] \mathsf{U} \Box [\mathbb{T}_i]. \tag{6.2}$$

Then, we use (6.2) to write the set of specified tasks as $\{\varphi_i\}_{i\in\mathcal{E}}$.

The objective of this chapter is to solve the following problem.

Problem 6.1. *Given a vehicle and a multi-objective parking task as a set of LTL specifications, design a controller that guides the remote operator to safely park the vehicle into an empty parking spot.*

6.3 Guiding Controller

This section provides a guiding controller solution to the remote parking problem. Recall that the parking task is given as a set of specifications $\{\varphi_i\}_{i\in\mathcal{E}}$. We

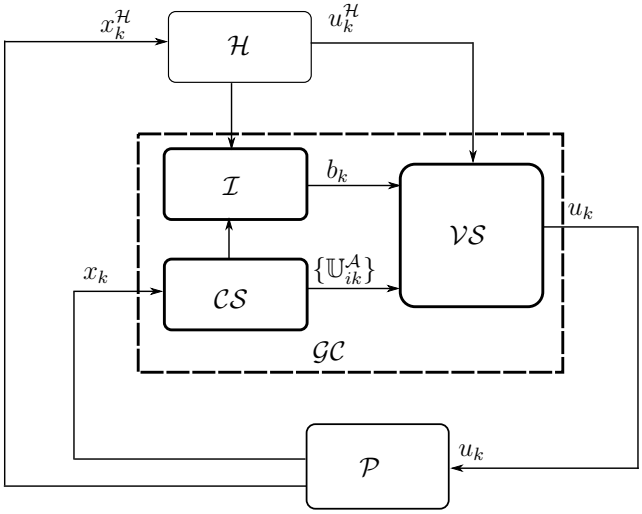


Figure 6.2: Guiding control framework. $\mathcal{H}$: human decision-maker; $\mathcal{P}$: plant; $\mathcal{GC}$: guiding controller; $\mathcal{I}$: inferring; $\mathcal{CS}$: control set synthesis; $\mathcal{VS}$: verification-based synthesis.

assume that the human's preference over the specification group is uncertain, e.g., time-varying or random. In other words, the empty parking spot the human operator intends to park in is unknown.

Let us illustrate the solution by the block diagram in Figure 6.2, where the plant $\mathcal{P}$ corresponds to the vehicle and is described by the dynamics (6.1). A human operator ($\mathcal{H}$) is allowed to make decisions and provide inputs to a verification system corresponding to a guiding controller ($\mathcal{GC}$) that filters human decisions for ensuring mission completion and safety. In Figure 6.2, we distinguish the state x_k that is measured by the sensor and transmitted to the guiding controller with the state $x_k^{\mathcal{H}}$ that the human operator perceives by herself. According to the state $x_k^{\mathcal{H}}$, the human operator $\mathcal{H}$ can make decisions and provide inputs $u_k^{\mathcal{H}}$ to a guiding controller, denoted by $\mathcal{GC}$. This guiding controller filters the human's decision $u_k^{\mathcal{H}}$ to a verified control command u_k and sends it for implementation at the plant $\mathcal{P}$.

According to Problem 6.1, the main objective of this chapter is to design the guiding controller $\mathcal{GC}$ that consists of three submodules, as shown in Figure 6.2: (1) a control set synthesis module $\mathcal{CS}$ which provides a group of control sets, i.e., $\{\mathbb{U}_{ik}^{\mathcal{A}}\}_{i\in\mathcal{E}}$; (2) an inference module $\mathcal{I}$ which updates the auto-

mated system's belief b_k of which specified objective the human intends to complete; and (3) a verification-based synthesis module $\mathcal{VS}$ which provides a verified control command u_k for satisfying the LTL specified task whenever the human's decision does not satisfy the specification.

We do not impose any expectations on how a human actually performs a decision-making process, but only assume that the human can synthesize a control input $u_k^{\mathcal{H}}$ at each time instant k. Next, we will show how to design the inferring module $\mathcal{I}$ and the verification synthesis $\mathcal{VS}$. Using these modules, we will then outline the algorithm for our guiding controller $\mathcal{GC}$.

6.3.1 Control Set Synthesis $\mathcal{CS}$

Recall that the TLT-based approach in Chapter 5 can online synthesize a feasible control set at each time step for discrete-time uncertain systems under LTL specifications. Such an approach is applicable for the LTL-specified parking task.

Given the LTL formula φ_i in (6.2), we can construct the controlled TLT of φ_i by using reachability analysis. In detail, we can represent the controlled TLT of φ_i in the form of complete paths, i.e.,

$$\mathbb{Y}_{1,i} \mathsf{U} \mathbb{Y}_{2,i} \mathsf{U} \mathcal{RCI}(\mathbb{T}_i) \Box \mathbb{T}_i \tag{6.3}$$

where $\mathbb{Y}_{1,i} = \mathcal{R}^{\mathrm{c}}(\mathbb{X} \setminus (\cup_{j \in \mathcal{F}} \mathbb{P}_i), \mathbb{Y}_{2,i})$ and $\mathbb{Y}_{2,i} = \mathcal{R}^{\mathrm{c}}(\mathbb{P}_i, \mathcal{RCI}(\mathbb{T}_i))$.

The simple form of the controlled TLT allows us to directly design the control set, without the series of complex operations used in Algorithms 5.4, 5.2, and 5.5 of Chapter 5. Given the measured state x_k, the feasible control set $\mathbb{U}_{ik}^{\mathcal{A}}(x_k)$ for the LTL formula φ_i is synthesized by Algorithm 6.1.

6.3.2 Inference Module $\mathcal{I}$

As mentioned before, we assume that the human's preference is unknown for the guiding controller $\mathcal{GC}$. We introduce a specification belief b_k, which is a probability distribution vector over the specification group. Each element $b_k(i)$ quantifies the preference of the human on the specification φ_i. The inference module $\mathcal{I}$ is to update this belief b_k in a data-driven manner. If the decision of the human $u_k^{\mathcal{H}}$ satisfies the specification φ_i, i.e., $u_k^{\mathcal{H}} \in \mathbb{U}_{ik}^{\mathcal{A}}$, we justify that the human's preference also satisfies this specification at time instant k. We denote by a $0-1$ vector $o_k \in \mathbb{R}^{N_s}$ the observation vector: if

Algorithm 6.1 Control Set Synthesis

Input: x_k and the controlled TLT in (6.3)
Output: a control set $\mathbb{U}_{ik}^{\mathcal{A}}(x_k)$

if $x_k \in \mathcal{RCI}(\mathbb{T}_i)$ **then**
  $\mathbb{U}_{ik}^{\mathcal{A}}(x_k) = \{u \in \mathbb{U} \mid \{Ax_k + Bu\} \oplus \mathbb{W} \subseteq \mathcal{RCI}(\mathbb{T}_i)\}$;
else
  if $x_k \in \mathbb{Y}_{2,i}$ **then**
    $\mathbb{U}_{ik}^{\mathcal{A}}(x_k) = \{u \in \mathbb{U} \mid \{Ax_k + Bu\} \oplus \mathbb{W} \subseteq \mathbb{Y}_{2,i}\}$;
  else
    if $x_k \in \mathbb{Y}_{1,i}$ **then**
      $\mathbb{U}_{ik}^{\mathcal{A}}(x_k) = \{u \in \mathbb{U} \mid \{Ax_k + Bu\} \oplus \mathbb{W} \subseteq \mathbb{Y}_{1,i}\}$;
    else
      $\mathbb{U}_{ik}^{\mathcal{A}}(x_k) = \emptyset$;
    end if
  end if
end if
return $\mathbb{U}_{ik}^{\mathcal{A}}(x_k)$.

$u_k^{\mathcal{H}} \in \mathbb{U}_{ik}^{\mathcal{A}}$, $o_k(i) = 1$; otherwise, $o_k(i) = 0$. According to the Bayesian rule, the specification belief is updated as

$$b_{k+1}(i) = \frac{o_k(i)b_k(i)(\mathrm{vol}(\mathbb{U}_{ik}^{\mathcal{A}}) + \epsilon)}{\sum_{i=1}^{N_s} o_k(i)b_k(i)(\mathrm{vol}(\mathbb{U}_{ik}^{\mathcal{A}}) + \epsilon)}. \tag{6.4}$$

Here, $\mathrm{vol}(\cdot)$ denotes the set volume. We define $\mathrm{vol}(\emptyset) = -\infty$ and $0 \times (-\infty) = 0$. In addition, ϵ is a positive constant to avoid the singular case when $\mathrm{vol}(\mathbb{U}_{ik}^{\mathcal{A}}) \leq 0, \forall i$. Intuitively, the larger the volume of $\mathbb{U}_{ik}^{\mathcal{A}}$ is, the easier for the operator to complete the specification φ_i, which in turn means that the more likely the human chooses φ_i.

6.3.3 Verification-based Synthesis Module $\mathcal{VS}$

After synthesizing the control sets $\{\mathbb{U}_{ik}^{\mathcal{A}}\}_{i=1}^{N_s}$ for all the specifications, we use a verification synthesis scheme to filter the human decision. If the decision of the human satisfies some specification, the decision will be respected. Otherwise, it will be corrected based on the specification belief b_k and the control sets $\{\mathbb{U}_{ik}^{\mathcal{A}}\}_{i=1}^{N_s}$. Mathematically, the control input u_k after verification synthesis is

derived as

$$u_k = \begin{cases} u_k^{\mathcal{H}}, & \text{if } \exists i \text{ s.t. } u_k^{\mathcal{H}} \in \mathbb{U}_{ik}^{\mathcal{A}}, \\ \underset{u \in \mathbb{U}_{ik}^{\mathcal{A}}, i=1,\ldots,N_s}{\operatorname{argmin}} \frac{\|u-u_k^{\mathcal{H}}\|}{b_k(i)}, & \text{otherwise}, \end{cases} \tag{6.5}$$

where $u_k^{\mathcal{H}}$ is the original human decision. In (6.5), the belief $b_k(i)$ plays the role of weighing the distance between $u_k^{\mathcal{H}}$ and $\mathbb{U}_{ik}^{\mathcal{A}}$. Larger $b_k(i)$'s increase the possibility of choosing the projected control input of $u_k^{\mathcal{H}}$ on the set $\mathbb{U}_{ik}^{\mathcal{A}}$.

6.3.4 Online Algorithm

Next we develop an online algorithm for the guiding controller $\mathcal{GC}$. Due to the presence of disturbances w_k, we implement the robust guiding controller in a closed-loop manner. As shown in Algorithm 6.2, at each time instant k, if all the synthesized control sets $\mathbb{U}_{ik}^{\mathcal{A}}(x_k)$ are empty, i.e., all specifications are infeasible, the algorithm outputs $\mathrm{Infeasible}$ (lines 7–8). Otherwise, the guiding controller will mix the decision of the human $u_k^{\mathcal{H}}$ and the synthesized control sets $\mathbb{U}_{ik}^{\mathcal{A}}$ to synthesize the control input u_k (lines 4, 5, and 10). Meanwhile, the specification belief b_k is updated (line 11).

Algorithm 6.2 Guiding Controller Algorithm

1: Initialization: Set $k = 0$ and $\mathrm{TerInd} = 1$;
2: **while** TerInd **do**
3: Measure x_k;
4: Human makes a decision $u_k^{\mathcal{H}}$;
5: Synthesize $\mathbb{U}_{ik}^{\mathcal{A}}$ for each φ_i by Algorithm 6.1;
6: **if** $\mathbb{U}_{ik}^{\mathcal{A}} = \emptyset, \forall i \in \mathbb{N}_{[1,N_s]}$ **then**
7: $\mathrm{TerInd} = 0$;
8: **Output**: $\mathrm{Infeasible}$;
9: **else**
10: Synthesize controller u_k by (6.5);
11: Update specification belief b_k by (6.4);
12: Implement u_k;
13: Update $k = k + 1$;
14: **end if**
15: **end while**

The following theorem shows that Algorithm 6.2 stays feasible.

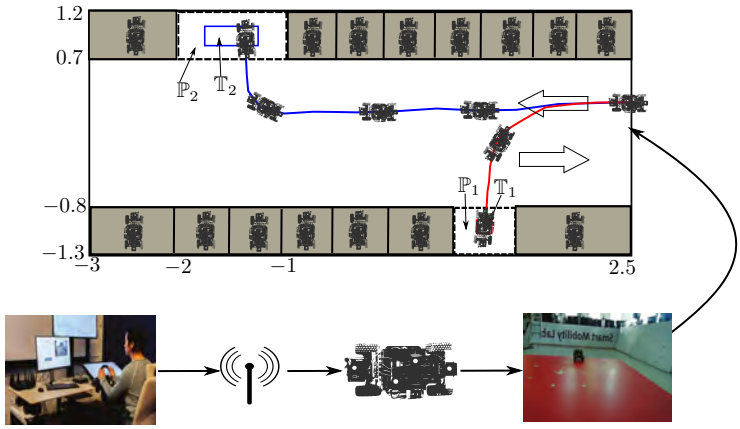


Figure 6.3: A parking situation where a remote human operator would like to drive a vehicle to a narrow parking space $\mathbb{P}_1$ or a broad parking space $\mathbb{P}_2$.

Theorem 6.1. *If the set $\mathcal{E}$ is nonempty and the initial state $x_0 \in \mathbb{Y}_{1,i}$ for some $i \in \mathcal{E}$, then Algorithm 6.2 is feasible for all $k \in \mathbb{N}$;*

Proof. The result directly follows from the recursive feasibility under the TLT-based approach (see Theorem 5.6) and the verification-based synthesis (6.5). □

6.4 Experiments

In this section, we detail our experimental setup and report experimental results. We consider the parking scenario shown in Figure 6.3 where there are two empty parking spots. The human operator would like to drive a vehicle to a narrow parking space $\mathbb{P}_1$ or a broad parking space $\mathbb{P}_2$. For the vehicle model, let the sampling period be 0.2 second, the control set $\mathbb{U} = \{u \in \mathbb{R}^2 \mid [-0.3, -0.3]^T \leq u \leq [0.3, 0.3]^T\}$, and the disturbance set $\mathbb{W} = \{w \in \mathbb{R}^2 \mid [-0.01, -0.01]^T \leq w \leq [0.01, 0.01]^T\}$. The LTL-specified parking tasks are

$$\varphi_1 = \big([\mathbb{X}] \wedge (\wedge_{j \in \mathcal{F}} \neg[\mathbb{P}_j])\big) \mathsf{U} [\mathbb{P}_1] \mathsf{U} \Box [\mathbb{T}_1],$$
$$\varphi_2 = \big([\mathbb{X}] \wedge (\wedge_{j \in \mathcal{F}} \neg[\mathbb{P}_j])\big) \mathsf{U} [\mathbb{P}_2] \mathsf{U} \Box [\mathbb{T}_2].$$

6.4.1 Experimental Setup

The experimental setup consists of three components: the ego vehicle, a human operator interface, and the parking lot environment, see Figure 6.3.

The ego vehicle is represented by the SVEA platform, which is a small robotic car platform designed to evaluate automated vehicle-related software stacks. For our experiment, we equip the SVEA car with an ELP fish-eye camera to provide a wide-angle view for the human operator and a TP-Link 4G LTE modem for streaming both the camera data to the human operator and the control from the human operator back to the SVEA car.

For the human operator interface, we place a human at a teleoperation desk built to support the management of remotely connected vehicles. A computer at the teleoperation desk is connected to the internet and is running a WebRTC-based app that handles the data transmission between the teleoperation station and the SVEA car over a peer-to-peer connection. The human can provide input to the control system with a Logitech G29 steering wheel and pedals. This interface subsumes the $\mathcal{GC}$ block in Figure 6.2. The free parking spots and obstacles are all in the coordinate frame of a Qualisys motion capture system.

6.4.2 Experimental Results

The human operator is parking the vehicle in parking space $\mathbb{P}_2$, corresponding to specification φ_2. The video of the experiment is available at https://youtu.be/WhFNleymOJ8.

We show snapshots of the vehicle's position in Figure 6.4. Here, we highlight the position of the vehicle by the red box and show the view of the human operator in the bottom right corner of each snapshot. The corresponding position trajectory is shown in Figure 6.5. We can see that during the parking process, there is no collision between the vehicle and the obstacles.

Figure 6.6 shows the control inputs and the control sets, where the dashed lines denote the control bounds. Here, $u^x = v^x$ and $u^y = v_y$. The blue lines are the decisions $u_k^{\mathcal{H}}$ of the human driver while the black lines are the implemented control inputs u_k by using the guiding controller in Figure 6.2. The red and cyan regions represent the synthesized control sets $\mathbb{U}_{1k}^{\mathcal{A}}$ and $\mathbb{U}_{2k}^{\mathcal{A}}$ for φ_1 and φ_2, respectively. We can see that all the control commands u_k always belong to the synthesized control sets while this is not true for the human decisions.

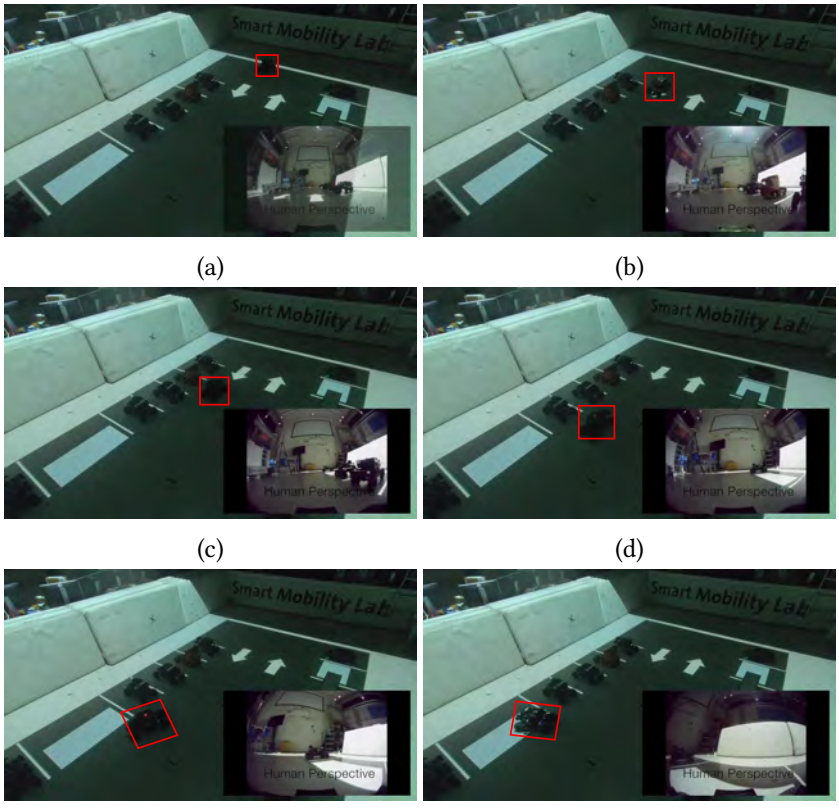


Figure 6.4: Snapshots of the vehicle's position when a human remotely drives the vehicle to the parking space $\mathbb{P}_2$. We highlight the position of the vehicle by the red box and show the view of the human operator in the bottom right corner of each snapshot.

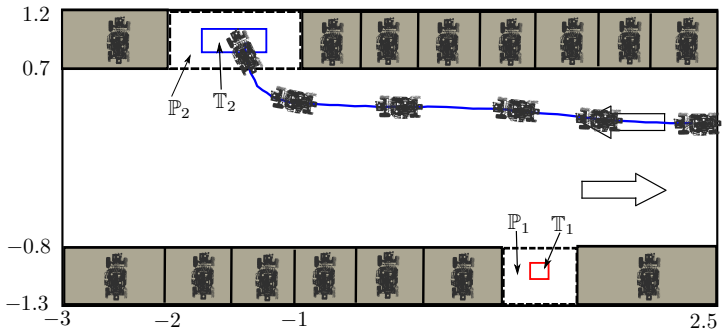


Figure 6.5: Position trajectory of the experimental realization in Figure 6.4.

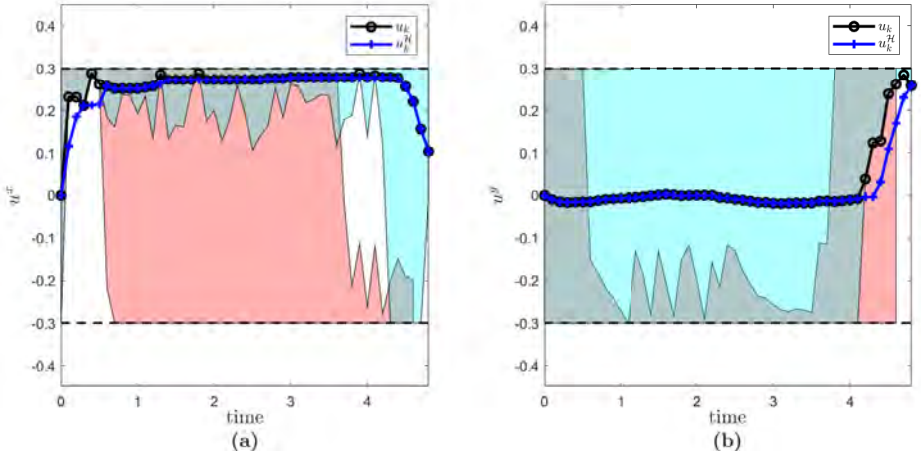


Figure 6.6: Control trajectory by using the guiding controller. The blue lines are the decisions $u_k^{\mathcal{H}}$ of the human driver while the black lines are the implemented control inputs u_k. The red and cyan regions represent the synthesized control sets $\mathbb{U}_{1k}^{\mathcal{A}}$ and $\mathbb{U}_{2k}^{\mathcal{A}}$ for φ_1 and φ_2, respectively.

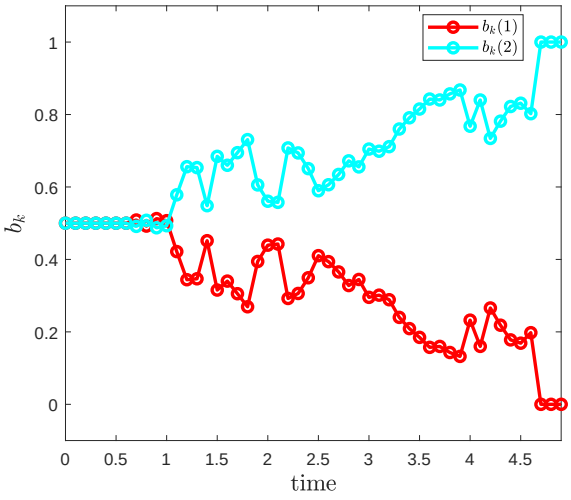


Figure 6.7: Belief update b_k when the human drives the vehicle to the parking space 2.

Note that at some time instants, the human's decision cannot satisfy any specification, thus the input is corrected by the verification-based synthesis according to the synthesized control sets. After 4.6 seconds (at which p_k^x is about 1 m), the synthesized control set for φ_1 is empty since this specification becomes infeasible. This can also be observed from Figure 6.7, which shows the belief update. Note that the beliefs in φ_1 and φ_2 oscillate from 1.2 seconds to 2.6 seconds since the volume of the control sets changes significantly during this time interval. After that, the belief in φ_2 increases since the vehicle passes the parking space $\mathbb{P}_1$ and approaches the parking space $\mathbb{P}_2$, which then becomes more likely.

In this example, we can observe the capabilities of our approach. Even though the system's initial belief is neutral, as the human operates the vehicle, the system updates its belief appropriately. The guiding controller works together with the human operator to complete the parking maneuver.

6.5 Summary

In this chapter, we presented a solution to a remote car parking problem, where the parking task was specified as a set of LTL formulae. We gave priority to the human operator's decision, allowing her to perform actions on the vehicle over a communication network. The framework made no assumptions about the operator's preference. Our system updates a data-driven belief of the operator's intent. We utilized the TLT-based approach to synthesize the control sets for LTL formulae. We proved recursive feasibility of the method, showing that the controller is always feasible and able to guarantee that the human will not be able to drive the system to an unsafe set. We illustrated the effectiveness of the proposed method by hardware experiments.

Chapter 7

Car Overtaking Application

The previous chapter investigates the shared-autonomy systems where the human action are used proactively. In this chapter, we consider another kind of shared-autonomy systems where the human action are used reactively. Specifically, we develop a solution to the overtaking control problem where an automated vehicle tries to overtake a human-driven vehicle with uncertain motion. The uncertainty in the predicted motion makes automated overtaking hard, for example, to guarantee feasibility. We introduce the weak assumption that the predicted velocity of the overtaken vehicle respects a supermartingale, meaning that its velocity is not increasing in expectation during the maneuver. We show that this formulation presents a natural notion of risk. Based on the martingale assumption, we perform a risk-aware reachability analysis by analytically characterizing the predicted collision probability. Then, we design a risk-aware optimal overtaking algorithm with guaranteed levels of collision avoidance. Finally, a simulated example illustrates the effectiveness of the proposed algorithm.

This chapter is structured as follows. In Section 7.1, we motivate the addressed problem. Section 7.2 gives some preliminaries and Section 7.3 formulates the car overtaking problem. Section 7.4 proposes risk-aware reachability analysis based on martingale theory. In Section 7.5, a risk-aware optimal overtaking algorithm is presented while Section 7.6 demonstrates the efficacy of our algorithm and compares our work with state-of-the-art. Finally, Section 7.7 concludes the chapter.

7.1 Introduction

7.1.1 Motivation

Overtaking is a dangerous driving maneuver with both lateral and longitudinal movements. Many researchers believe that by developing an approach for safe, robust, and efficient overtaking, we will significantly progress the safety of automated vehicles [22], [177].

Several existing works propose solutions for performing safe and efficient overtaking maneuvers [178]–[182]. However, these solutions all assume that the vehicle to be overtaken moves at a constant velocity. Under this assumption, overtaking can be formulated as a reference tracking problem or an optimal control problem. Note that these problems do not encounter any feasibility issues if the velocity of the overtaken vehicle is smaller than the velocity limit imposed by the traffic rule and the prediction horizon is chosen appropriately. Although the constant velocity assumption is a natural choice in many practical implementations, it is interesting to consider overtaking vehicles that do not drive at a fixed velocity, especially since human drivers can change their velocities while being overtaken, depending on how they react to the situation. In this chapter, we present examples where if the vehicle being overtaken is changing its velocity, we can improve the overtaking compared to existing control laws.

7.1.2 Main Contributions

We study the process of overtaking a human-driven vehicle V_2 by an automated vehicle V_1, as shown in Figure 7.1. The overtaking maneuver requires the automated vehicle to laterally move into an empty lane when it is safe to initiate the maneuver, longitudinally overtake V_2, and, finally, laterally merge back into the original lane in front of V_2. At each stage of the overtaking maneuver, many factors introduce uncertainty, which make the overtaking hard to perform robustly and safely [183].

The main contribution of this chapter is to develop an algorithm that increases the possibility of a feasible automated overtaking while guaranteeing the safety of both V_1 and V_2, even when V_2 does not move at a constant velocity. More specifically,

(C7.1) We propose a formulation for risk-aware reachability analysis based on martingale theory. We develop risk-aware reachable sets, which are sub-

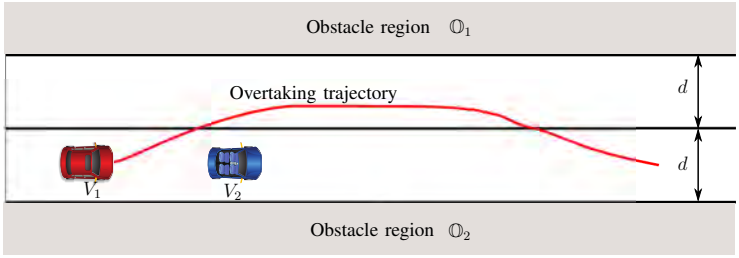


Figure 7.1: Scenario where automated vehicle V_1 is overtaking a human-driven vehicle V_2 on a road with two lanes.

sets of traditional reachable sets. Risk-aware reachable sets are shown to admit predicted collision probabilities between V_1 and V_2. We estimate these collision probabilities analytically using the concentration inequality of martingale theory.

(C7.2) Based on the proposed risk-aware reachability analysis, we design a risk-aware optimal overtaking algorithm. It is used to solve the overtaking problem in a receding-horizon manner. We provide sufficient and necessary conditions for the feasibility of the risk-aware optimal overtaking problem. In other words, by performing a safety check a step ahead of the execution of each control command, our algorithm can guarantee that the overtaking process is collision-free.

7.1.3 Related Work

There is a significant body of work on automated overtaking under the assumption that the vehicle to be overtaken moves at a constant velocity. For example, model predictive control is used to track a reference overtaking trajectory in [178]–[180]. To handle collision avoidance, the overtaking problem is formulated as a mixed integer program in [181], [182]. In [184], a constrained iterative linear quadratic regulator is used to efficiently solve the overtaking problem. The recent work [185] does not adopt the constant velocity assumption and considers measurement noises in the overtaking scenario, but no formal safety guarantees are provided.

Reachability analysis is used to give formal safety guarantees for vehicle control [55], [56]. Robust approaches maintain strict guarantees that despite the bounded disturbances, the state trajectory can be kept by a feedback con-

troller into a safe state tube with a certain time horizon [44]. In [186]–[188], the authors formulate approaches for reachability-based automated overtaking for vehicles that overtake static obstacles using the opposing direction's lane. In [57], the reachability is incorporated a model predictive controller for ensuring the safety of an automated vehicle when interacting with a human-driven vehicle. On the other hand, stochastic approaches guarantee that there will not be an unsafe trajectory within a certain time horizon for a given probability [15]. Stochastic reachability approaches can be beneficial, since they permit a trade-off between collision probability and optimality, while avoiding too conservative decisions. However, the introduction of stochasticity often introduces the additional challenge of finding high-fidelity stochastic models. In [189], a Markov chain is used to model the uncertainty of other traffic participants and applied to probabilistic automated overtaking. Markov decision processes or partially observable Markov decision processes are used to model the stochasticity of human driving behavior in [190], [191]. Such assumptions on human driving behaviors are quite strong. Another work [192] proposes an empirical method for generating approximate stochastic reachable sets for human-in-the-loop driving systems. However, in many overtaking scenarios, an automated vehicle will not have the historical data to generate these empirical sets. In our recent work [193], we choose to use martingales to model the expected behavior of a human driver during overtaking. This assumption is weaker than the Markovian assumptions above, since less historical data is needed for the validation of martingale-based model than the identification of the Markovian models. This chapter generalizes the preliminary conference results of [193] significantly in that we use a more detailed nonlinear vehicle model, formulate a risk-aware optimal overtaking problem, and propose a more general solution for solving this problem.

Historically, martingales are often applied to gambling or pricing problems since they efficiently model the lack of arbitrage [194]. In [195], the author discusses the use of martingales in several classical stochastic control problems. In particular, supermartingales plays an important role in proving the stochastic stability. In addition, martingale theory is also relevant to the risk theory. A risk-neutral measure is also called a martingale measure [196]. In [197], it is shown that the multiportfolio time consistency of a dynamic multivariate risk measure is equivalent to a supermartingale property. Under the supermartingale assumption, we propose in this chapter risk-aware reachable sets and develop a risk-aware overtaking algorithm.

7.2 Preliminaries

This section provides some preliminaries that will be used in this chapter.

7.2.1 Martingale Theory

We will use the following definition and inequality from martingale theory. A supermartingale is a stochastic process for which the conditional expectation of future values given the history information is bounded above by the current value. A formal definition is given as follows.

Definition 7.1. *[198] A discrete-time integrable stochastic process* $\{X_i, i \in \mathbb{N}\}$ *on a probability space* $(\Omega, \mathcal{F}, \Pr)$*, with a filtration* $\{\mathcal{F}_i, i \in \mathbb{N}\}$ *and* $\mathcal{F}_i \subseteq \mathcal{F}$*, is said to be a supermartingale if* $\mathrm{E}[X_{i+1}|X_i] \leq X_i, \forall i \in \mathbb{N}$.

Due to the decreasing property of supermartingales, the following concentration inequality holds.

Lemma 7.1. *[199] Consider a discrete-time supermartingale* $\{X_i, i \in \mathbb{N}\}$ *with a filtration* $\{\mathcal{F}_i, i \in \mathbb{N}\}$ *and* $\mathcal{F}_i \subseteq \mathcal{F}$*. If for all* $i \in \mathbb{N}_{\geq 1}$*, and some positive* σ_i *and* M*,* $\mathrm{Var}[X_i|\mathcal{F}_{i-1}] \leq \sigma_i^2$ *and* $X_i - \mathrm{E}[X_i|X_{i-1}] \leq M$*, then for all* $\eta \geq 0$*,*

$$\Pr[X_i \geq X_0 + \eta] \leq \exp\Big(-\frac{\eta^2}{2(\sum_{j=1}^{i}\sigma_j^2 + M\eta/3)}\Big).$$

7.2.2 Vehicle Collision-free Conditions

Consider a vehicle with state $\boldsymbol{x} = [p^x\ p^y\ \theta\ v]^T$, where (p^x, p^y) is the center of the rear axis (see Figure 7.2), θ the heading angle, and v the velocity. For a given state $\boldsymbol{x}$, the occupancy of the vehicle is

$$\mathbb{S}(\boldsymbol{x}) = R(\boldsymbol{x})\mathbb{B} \oplus p(\boldsymbol{x}),$$

where $\mathbb{B} = \{z \in \mathbb{R}^2 \mid Gz \leq g\}$ is the initial rectangle occupied by the vehicle when the center of the rear axes is $[0; 0]$ and specified by $G \in \mathbb{R}^{2\times 2}$ and $g \in \mathbb{R}^2$, $R(\boldsymbol{x}) = [\cos\theta\ \sin\theta; -\sin\theta\ \cos\theta]$ the rotation matrix, and $p(\boldsymbol{x}) = [p^x; p^y]$.

Consider an obstacle $\mathbb{O} \subseteq \mathbb{R}^2$ of the form

$$\mathbb{O} = \{z \in \mathbb{R}^2 \mid Hz \leq h\},$$

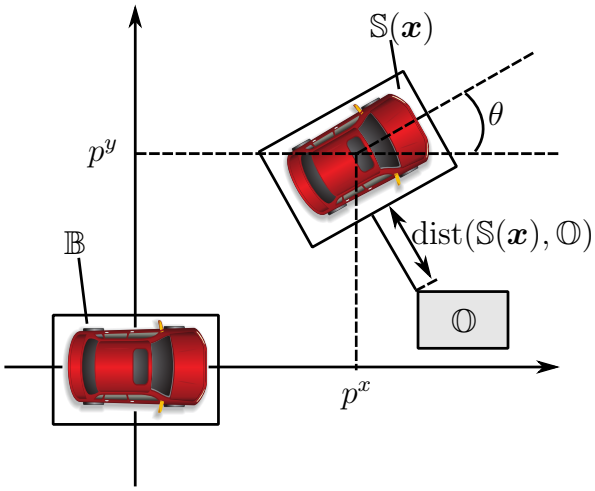


Figure 7.2: Notation for obstacle avoidance.

where $H \in \mathbb{R}^{2\times 2}$ and $h \in \mathbb{R}^2$ are a known matrix and vector, respectively. Define the distance between $\mathbb{S}(\boldsymbol{x})$ and $\mathbb{O}$ as

$$\text{dist}(\mathbb{S}(\boldsymbol{x}), \mathbb{O}) = \min_{z\in\mathbb{R}^2}\{\|z\| \mid (\mathbb{S}(\boldsymbol{x}) \oplus z) \cap \mathbb{O} \neq \emptyset\}.$$

The following lemma provides a computationally useful way for checking if $\text{dist}(\mathbb{S}(\boldsymbol{x}), \mathbb{O}) > d$ for some $d > 0$.

Lemma 7.2. *[200] For any $d > 0$, $\text{dist}(\mathbb{S}(\boldsymbol{x}), \mathbb{O}) > d$ if and only if there exist $\lambda \geq 0$ and $\mu \geq 0$ such that*

$$\begin{cases} -g^T\mu + (Hp(\boldsymbol{x}) - h)^T\lambda > d, \\ G^T\mu + R^T(\boldsymbol{x})H^T\lambda = 0, \\ \|H^T\lambda\| \leq 1. \end{cases}$$

The equivalent condition in Lemma 7.2 is derived by using the dual problem of $\text{dist}(\mathbb{S}(\boldsymbol{x}), \mathbb{O})$. An important property of this equivalent condition is that all the decision variables are real numbers. This is different from the integer-based collision-avoidance formulation in the literature when taking into account the occupancy of the vehicle. The result of Lemma 7.2 will be used to reformulate the safety constraints on the overtaking vehicle in this chapter.

7.3 Problem Formulation

We consider an overtaking scenario with an automated vehicle V_1 and a human-driven vehicle V_2 as illustrated in Figure 7.1. Regard the two regions outside of the lanes as obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$ of the form

$$\mathbb{O}_i = \{z \in \mathbb{R}^2 \mid H_i z \leq h_i\}, i = 1, 2.$$

The width of each lane is d. The longitudinal velocity of each vehicle is bounded $v^R_{\min} \leq v^x \leq v^R_{\max}$.

7.3.1 Automated Vehicle V_1

We describe the dynamics of the automated vehicle V_1 by using the following bicycle model:

$$\boldsymbol{x}_1(k+1) = f(\boldsymbol{x}_1(k), \boldsymbol{u}_1(k)),$$

where

$$\boldsymbol{x}_1 = \begin{bmatrix} p_1^x \\ p_1^y \\ \theta_1 \\ v_1 \end{bmatrix}, \ \boldsymbol{u}_1 = \begin{bmatrix} \psi_1 \\ a_1 \end{bmatrix}.$$

The longitudinal and lateral positions (p_1^x, p_1^y) correspond to the center of the rear axes, θ_1 is the yaw angle with respect to the x-axis, v_1 is the velocity with respect to the rear axes, ψ_1 is the steering angle, and a_1 is the acceleration. The update map is

$$f(\boldsymbol{x}_1, \boldsymbol{u}_1) = \begin{bmatrix} p_1^x \\ p_1^y \\ \theta_1 \\ v_1 \end{bmatrix} + \delta \begin{bmatrix} v_1 \cos\theta_1 \\ v_1 \sin\theta_1 \\ L^{-1} v_1 \tan\psi_1 \\ a_1 \end{bmatrix},$$

where L is the wheel base and δ the sampling period.

There are physical limits on the state and control input:

$$\boldsymbol{x}_1(k) \in \mathbb{X}_1, \ \boldsymbol{u}_1(k) \in \mathbb{U}_1,$$
$$\mathbb{X}_1 = \Big\{ \boldsymbol{z} \in \mathbb{R}^4 \mid \boldsymbol{z} = [p^x, p^y, \theta, v]^T, \ 0 \leq p^y \leq 2d, \theta_{\min} \leq \theta \leq \theta_{\max},$$

$$v^R_{\min} \le v\cos\theta \le v^R_{\max}\Big\},$$

$$\mathbb{U}_1 = \Big\{\boldsymbol{u} \in \mathbb{R}^2 \mid \boldsymbol{u} = [\psi\ a]^T,\ \psi_{\min} \le \psi \le \psi_{\max}, a_{1,\min} \le a \le a_{1,\max}\Big\}.$$

The occupancy of V_1 is

$$\mathbb{S}_1(\boldsymbol{x}_1(k)) = R(\boldsymbol{x}_1(k))\mathbb{B}_1 \oplus \begin{bmatrix} p^x_1(k) \\ p^y_1(k) \end{bmatrix},$$

where $\mathbb{B}_1 = \{z \in \mathbb{R}^2 \mid G_1 z \le g_1\}$ is similarly defined as in previous section.

7.3.2 Overtaken Vehicle V_2

We specify the dynamics of the vehicle V_2 we want to overtake by using a linear model:

$$\boldsymbol{x}_2(k+1) = A_2\boldsymbol{x}_2(k) + B_2\boldsymbol{u}_2(k),$$

where

$$\boldsymbol{x}_2 = \begin{bmatrix} p^x_2 \\ v^x_2 \end{bmatrix},\ \boldsymbol{u}_2 = a^x_2,$$
$$A_2 = \begin{bmatrix} 1 & \delta \\ 0 & 1 \end{bmatrix},\ B_2 = \begin{bmatrix} 0 \\ \delta \end{bmatrix}.$$

In particular, p^x_2, v^x_2, and a^x_2 are the longitudinal position, velocity, and acceleration, respectively. During the overtaking process, we assume that V_2 stays in the same lane and maintains lateral position $p^y_2(k) = d/2, \forall k \in \mathbb{N}$.

Vehicle V_2 has the state and control input constraints:

$$\boldsymbol{x}_2(k) \in \mathbb{X}_2,\ \boldsymbol{u}_2(k) \in \mathbb{U}_2,$$
$$\mathbb{X}_2 = \left\{z \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} -\infty \\ v^R_{\min} \end{bmatrix} \le z \le \begin{bmatrix} \infty \\ v^R_{\max} \end{bmatrix}\right\},$$
$$\mathbb{U}_2 = \{z \in \mathbb{R} \mid a^x_{2,\min} \le z \le a^x_{2,\max}\}.$$

The occupancy of V_2 is

$$\mathbb{S}_2(\boldsymbol{x}_2(k)) = \mathbb{B}_2 \oplus \begin{bmatrix} p^x_2(k) \\ d/2 \end{bmatrix},$$

where $\mathbb{B}_2 = \{z \in \mathbb{R}^2 \mid G_2 z \le g_2\}$.

7.3.3 Problem

Our objective is to design a sequence of control inputs such that V_1 starts behind V_2 and ends in front of V_2 by using the following maneuvers: lane-changing, lane-keeping, and merging. The constraint set $\mathbb{U}_2$ is known to the automated vehicle V_1. At each time step k, the automated vehicle V_1 can measure the states $\boldsymbol{x}_1(k)$ and $\boldsymbol{x}_2(k)$. Throughout the entire process, we maintain the following safety constraints: collision avoidance between V_1 and V_2 and collision avoidance between V_1 and $\mathbb{O}_i$, $i = 1, 2$.

7.4 Risk-Aware Reachability Analysis

The reachable set of a vehicle is a subset of the state space that can be reached by the vehicle state through control actions. In this section, we introduce the reachable set and the risk-aware reachable set of vehicle V_2 and discuss some of their properties.

7.4.1 Reachable Set of Vehicle V_2

At time step k, $p_2^x(k)$ and $v_2^x(k)$ are the longitudinal position and velocity of V_2, respectively.

Definition 7.2. *The reachable set predicted $i \in \mathbb{N}$ steps ahead at time step k is given by*

$$\begin{cases} \mathbb{P}(k+i+1|k) = (A_2\mathbb{P}(k+i|k) \oplus B_2\mathbb{U}_2) \cap \mathbb{X}_2, \\ \mathbb{P}(k|k) = \{\boldsymbol{x}_2(k)\}. \end{cases} \tag{7.1}$$

Since the dynamics of V_2 is linear, the input constraint set $\mathbb{U}_2$ is a compact polyhedron and the state constraint set $\mathbb{X}_2$ is a polyhedron, the reachable sets in Definition 7.2 are compact polyhedra for all finite $i \in \mathbb{N}$. We show the set $\mathbb{P}(k+i|k)$ in blue in Figure 7.3. Here, the x-axis denotes the position and the y-axis the velocity. The regions in different color are other reachable sets to be defined in the following. The rectangles below the x-axis are the corresponding occupancies also to be defined.

We define the projection of the reachable set on the longitudinal position as $\mathbb{P}_x(k+i|k) = \mathrm{Proj}_1\mathbb{P}(k+i|k)$ and the projection of the reachable set on the longitudinal velocity as $\mathbb{P}_v(k+i|k) = \mathrm{Proj}_2\mathbb{P}(k+i|k)$, where $\mathrm{Proj}_j(\mathbb{Q})$ denotes the projection of the set $\mathbb{Q}$ on the jth dimension.

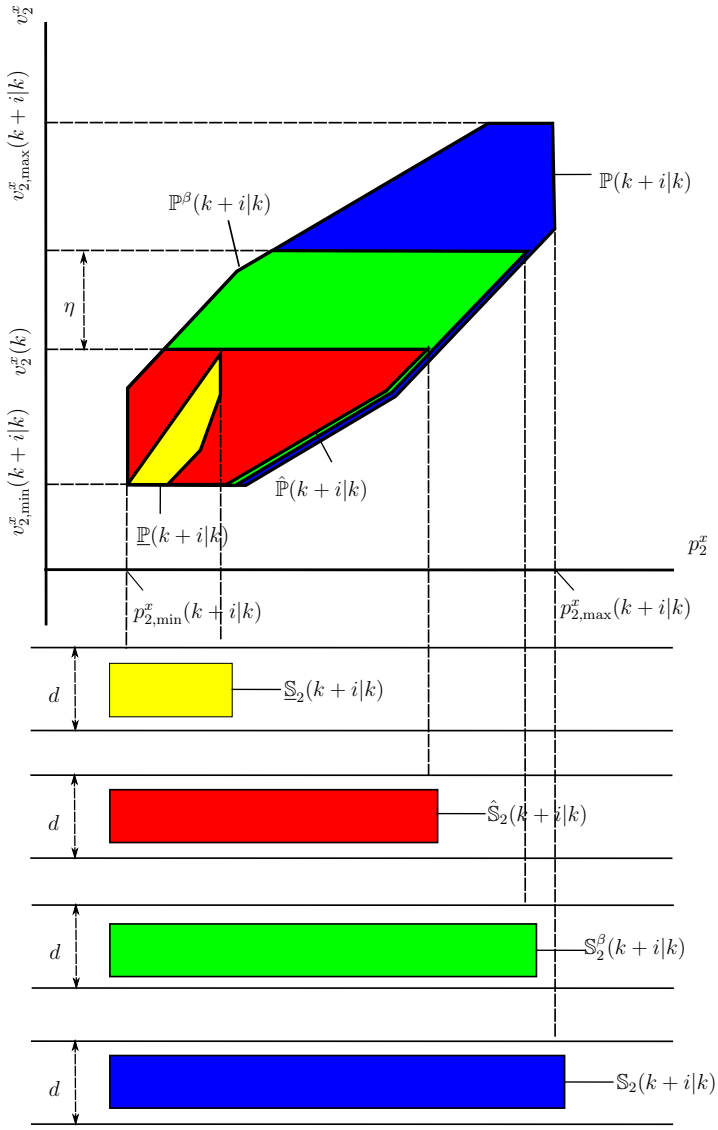


Figure 7.3: Reachable sets for V_2 in the plane and their corresponding occupancy sets. (a) blue: $\mathbb{P}(k+i|k)$ defined in (7.1) and $\mathbb{S}_2(k+i|k)$ defined in (7.2); (b) green: $\mathbb{P}^\beta(k+i|k)$ defined in (7.5) and $\mathbb{S}_2^\beta(k+i|k)$ defined in (7.6); (c) red: $\hat{\mathbb{P}}(k+i|k)$ defined in (7.8) and $\hat{\mathbb{S}}_2(k+i|k)$ defined in (7.7); and (d) $\underline{\mathbb{P}}(k+i|k)$ defined in (7.9) and $\underline{\mathbb{S}}_2(k+i|k)$ defined in (7.10). Here, η is defined in (7.4).

Note that the set $\mathbb{P}(k+i|k)$ is a compact and convex set for all finite $i \in \mathbb{N}$. Furthermore, the sets $\mathbb{P}_x(k+i|k)$ and $\mathbb{P}_v(k+i|k)$ are closed intervals. For notational simplicity, let

$$\mathbb{P}_x(k+i|k) = [p^x_{2,\min}(k+i|k), p^x_{2,\max}(k+i|k)],$$
$$\mathbb{P}_v(k+i|k) = [v^x_{2,\min}(k+i|k), v^x_{2,\max}(k+i|k)].$$

These interval boundaries are also shown in Figure 7.3. Denote all possible occupancies of V_2 corresponding to $\mathbb{P}_x(k+i|k)$ as $\mathbb{S}_2(k+i|k)$, i.e.,

$$\mathbb{S}_2(k+i|k) = \Big\{ z \in \mathbb{R}^2 \mid z \in \mathbb{B}_2 \oplus [p^x_1(k); d/2],\ p^x_2 \in \mathbb{P}_x(k+i|k) \Big\}. \quad (7.2)$$

It is a compact rectangle, which we denote as

$$\mathbb{S}_2(k+i|k) = \{z \in \mathbb{R}^2 \mid S(k+i|k)z \leq s(k+i|k)\},$$

where $S(k+i|k)$ and $s(k+i|k)$ are a matrix and vector, respectively, with appropriate dimensions. The occupancy set $\mathbb{S}_2(k+i|k)$ is the blue rectangle below the x-axis in Figure 7.3.

7.4.2 Risk-Aware Reachable Set of Vehicle V_2

Let us next introduce the risk-aware reachable sets of V_2. Given any state $\boldsymbol{x}_2(k) \in \mathbb{X}_2$ at time step k, assume that the predicted velocity $\{v^x_2(k+i), i \in \mathbb{N}\}$ is a stochastic process with a filtration $\{\mathcal{B}(\mathbb{P}_v(k+i|k)), i \in \mathbb{N}\}$ on the probability space $([v^R_{\min}, v^R_{\max}], \mathcal{B}([v^R_{\min}, v^R_{\max}]), \Pr)$. Then, the future state $\boldsymbol{x}_2(k+i|k)$, $i \geq 0$, is also a stochastic process.

We assume that the predicted velocity $v^x_2(k+i|k)$, $i \geq 0$, is a supermartingale, according to Definition 7.1. Consequently, we assume that for any state $\boldsymbol{x}_2(k) \in \mathbb{X}_2$ at time step k,

$$\forall i \in \mathbb{N}, \begin{cases} \boldsymbol{x}_2(k+i|k) \in \mathbb{P}(k+i|k), \\ \mathrm{E}[v^x_2(k+i+1|k)|v^x_2(k+i|k)] \leq v^x_2(k+i|k). \end{cases}$$

Remark 7.1. *The supermartingale assumption makes it possible to incorporate uncertain behaviors of human drivers [201]. By assuming that the predicted velocity of the overtaken vehicle V_2 respects a supermartingale, we generalize the common assumption in the literature that V_2 moves at a constant velocity.*

Let $\boldsymbol{\alpha}(k) = \{\alpha(k+i|k), i \in \mathbb{N}\}$, where $0 \leq \alpha(k+i|k) \leq 1$, be the risk coefficient sequence at time step k. Under the supermartingale assumption, we define the sets

$$\mathbb{Y}^{\alpha}(k+i|k) = \{y \in \mathbb{R} \mid \Pr[\tilde{z} \geq v_2^x(k) + y] \leq \alpha(k+i|k), \\ \tilde{z} \in \mathbb{P}_v(k+i|k)\},$$

$$\mathbb{P}_v^{\alpha}(k+i|k) = \{v_2^x(k) + y \mid (v_2^x(k) + y) \in \mathbb{P}_v(k+i|k), \\ y \in \mathbb{Y}^{\alpha}(k+i|k)\},$$

which corresponds to the set of velocity that V_2 can reach with probability less than $\alpha(k+i|k)$. Next let us consider how to compute the set $\mathbb{P}_v^{\alpha}(k+i|k)$.

Proposition 7.1. *The set*

$$\mathbb{P}_v^{\alpha}(k+i|k) = [\min\{v_2^x(k) + \eta, v_{2,\max}^x(k+i|k)\},\ v_{2,\max}^x(k+i|k)], \quad (7.3)$$

where

$$\begin{cases} \eta = M\beta/3 + \sqrt{M^2\beta^2/9 + 2iM^2\beta}, \\ M = \max\{\delta|a_{2,\min}^x|, \delta a_{2,\max}^x\},\ \beta = -\ln(\alpha(k+i|k)). \end{cases} \quad (7.4)$$

Proof. Construct a filtration $\mathcal{F}_i = \mathcal{B}(\mathbb{P}_v(k+i|k))$. It follows from Popoviciu's inequality [202] that the variance of $v_2(k+i|k)$, conditioned on $v_2(k+i-1|k)$, is upper bounded by M^2, which implies that we can choose $\forall i \geq 1, \sigma_i^2 = M^2$ in the first condition of Lemma 7.1. From the constraint on the longitudinal acceleration of vehicle V_2, i.e.,

$$|v_2^x(k+i|k) - v_2^x(k+i-1|k)| \leq M,$$

the second condition of Lemma 7.1 also holds. Hence, (7.3) and (7.4) follow from Lemma 7.1 and setting $\alpha(k+i|k) = \exp(-\frac{\eta^2}{2(\sum_{j=1}^{i}\sigma_j^2 + M\eta/3)})$. □

Remark 7.2. *The parameter η in (7.4) corresponds to the minimal y in the set $\mathbb{Y}^{\alpha}(k+i|k)$ such that $v_2^x(k+y) \in \mathbb{P}_v(k+i|k)$. The set $\mathbb{P}_v^{\alpha}(k+i|k)$ is nonempty for all $0 \leq \alpha(k+i|k) \leq 1$.*

Define another sequence $\boldsymbol{\beta}(k) = \{\beta(k+i|k), i \in \mathbb{N}\}$, where $\beta(k+i|k) = 1 - \alpha(k+i|k), \forall i \in \mathbb{N}$. Let $\mathbb{P}_v^{\beta}(k+i|k) = \mathrm{cl}(\mathbb{P}_v(k+i|k) \setminus \mathbb{P}_v^{\alpha}(k+i|k))$, i.e.,

$$\mathbb{P}_v^{\beta}(k+i|k) = [v_{2,\min}^x(k+i|k), \min\{v_2^x(k)+\eta, v_{2,\max}^x(k+i|k)\}].$$

Next we define the risk-aware reachable set.

Definition 7.3. *The risk-aware reachable set for risk coefficient $\alpha(k+i|k)$ is defined as*

$$\mathbb{P}^{\beta}(k+i|k) = \Big\{ z \in \mathbb{R}^2 \mid z \in \mathbb{P}(k+i|k),\ \mathrm{Proj}_2(z) \in \mathbb{P}_v^{\beta}(k+i|k), \\ \beta(k+i|k) = 1 - \alpha(k+i|k) \Big\}. \quad (7.5)$$

Figure 7.3 shows the set $\mathbb{P}^{\beta}(k+i|k)$ in green. Note that it is a subset of the reachable set $\mathbb{P}(k+i|k)$. We remark that the set $\mathbb{P}^{\beta}(k+i|k)$ depends on the risk coefficient $\alpha(k+i|k)$: the smaller risk that is acceptable, the larger risk-aware reachable set.

The projection of the risk-aware reachable set on the longitudinal position is denoted by

$$\mathbb{P}_x^{\beta}(k+i|k) = \mathrm{Proj}_1(\mathbb{P}^{\beta}(k+i|k)).$$

Denote by $\mathbb{S}_2^{\beta}(k+i|k)$ all the possible occupancies of the vehicle V_2 corresponding to $\mathbb{P}_x^{\beta}(k+i|k)$, i.e.,

$$\mathbb{S}_2^{\beta}(k+i|k) = \Big\{ z \in \mathbb{R}^2 \mid z \in \mathbb{B}_2 \oplus [p_2^x(k); d/2], p_2^x \in \mathbb{P}_x^{\beta}(k+i|k) \Big\}. \quad (7.6)$$

The possible occupancy $\mathbb{S}_2^{\beta}(k+i|k)$ of V_2 is a compact rectangle, which can be written as

$$\mathbb{S}_2^{\beta}(k+i|k) = \{ z \in \mathbb{R}^2 \mid S^{\beta}(k+i|k) z \leq s^{\beta}(k+i|k) \},$$

where $S^{\beta}(k+i|k)$ and $s^{\beta}(k+i|k)$ are a matrix and a vector, respectively, with appropriate dimensions. The occupancy set $\mathbb{S}_2^{\beta}(k+i|k)$ is shown as the green rectangle below the x-axis in Figure 7.3.

Remark 7.3. *Another interpretation of the risk-aware reachable set is that*

$$\mathbb{P}^{\beta}(k+i|k) = \Big\{ z \in \mathbb{R}^2 \mid z \in \mathbb{P}(k+i|k), z = [z_1; z_2], \\ \mathbb{Q} = \mathbb{B}_2 \oplus [z_1; d/2], \\ \Pr(\mathbb{S}_1(k+i|k) \cap \mathbb{Q} \neq \emptyset) \leq 1 - \beta(k+i|k) \Big\},$$

which corresponds to the predicted state set that the vehicle V_2 can reach such that the collision probability with the vehicle V_1 is no greater than $1-\beta(k+i|k)$ under the supermartingale assumption.

7.4.3 Geometrical Interpretation of Risk-Aware Reachable Sets

This subsection provides some geometrical interpretations of the reachable sets defined above and establishes the relation among them.

First of all, we show that the risk-aware reachable set $\mathbb{P}^{\beta}(k+i|k)$ scales with the coefficient $\beta(k+i|k) = 1-\alpha(k+i|k)$. If $\boldsymbol{\alpha}(k)$ satisfies $\alpha(k+i|k) = 0$, i.e., $\beta(k+i|k) = 1, \forall i \in \mathbb{N}$, the risk-aware reachable sets equal to the reachable sets, i.e., $\mathbb{P}^{\beta}(k+i|k) = \mathbb{P}(k+i|k)$ and $\mathbb{P}_x^{\beta}(k+i|k) = \mathbb{P}_x(k+i|k)$. In this case, complete safety is guaranteed. If $\boldsymbol{\alpha}(k)$ instead satisfies $\alpha(k+i|k) = 1$, i.e., $\beta(k+i|k) = 0$, $\forall i \in \mathbb{N}$, the parameter η in (7.4) is 0, thereby resulting in

$$\mathbb{P}_v^{\alpha}(k+i|k) = [v_2^x(k), v_{2,\max}^x(k+i|k)],$$
$$\mathbb{P}_v^{\beta}(k+i|k) = [v_{2,\min}^x(k+i|k), v_2^x(k)].$$

We define the reachable sets, the projection on the longitudinal position, and the occupancy set for $\alpha(k+i|k) = 1$, $\forall i \in \mathbb{N}$, as follows

$$\hat{\mathbb{P}}(k+i|k) = \Big\{ z \in \mathbb{R}^2 \mid z \in \mathbb{P}(k+i|k), \\ \mathrm{Proj}_2(z) \in [v_{2,\min}^x(k+i|k), v_2^x(k)] \Big\}, \tag{7.7}$$

$$\hat{\mathbb{P}}_x(k+i|k) = \mathrm{Proj}_1(\hat{\mathbb{P}}(k+i|k)),$$

$$\hat{\mathbb{S}}_2(k+i|k) = \Big\{ z \in \mathbb{R}^2 \mid z \in \mathbb{B}_2 \oplus [p_2^x(k); d/2], \\ p_2^x \in \hat{\mathbb{P}}_x(k+i|k) \Big\}. \tag{7.8}$$

These sets correspond to the red regions in Figure 7.3. In this case, the automated overtaking problem is reduced to the scenario where we predict that the vehicle V_2 moves with an *average* velocity no greater than the current velocity $v_2^x(k)$.

For comparison with the control under the constant velocity assumption, we restrict the control set to $\underline{\mathbb{U}}_2 = [a^x_{2,\min}, 0]$, which implies that the predicted velocity cannot be greater than the measured velocity $v^x_2(k)$. Similarly, we define the reachable sets, the projection on the longitudinal position, and the occupancy set as follows

$$\begin{cases} \underline{\mathbb{P}}(k+i+1|k) = (A_2\underline{\mathbb{P}}(k+i|k) \oplus B_2\underline{\mathbb{U}}_2) \cap \mathbb{X}_2, \\ \underline{\mathbb{P}}(k+i|k) = \{\boldsymbol{x}_2(k)\}, \end{cases} \tag{7.9}$$

$$\underline{\mathbb{P}}_x(k+i|k) = \mathrm{Proj}_1(\bar{\mathbb{P}}(k+i|k)),$$

$$\underline{\mathbb{S}}_2(k+i|k) = \Big\{ z \in \mathbb{R}^2 \mid z \in \mathbb{B}_2 \oplus [p^x_2(k); d/2], \; p^x_2 \in \bar{\mathbb{P}}_x(k+i|k) \Big\}. \tag{7.10}$$

The set $\underline{\mathbb{P}}(k+i|k)$ represents the reachable sets when the vehicle V_2 moves with a velocity no greater than the current velocity, $v^x_2(k)$, i.e., the acceleration is no greater than 0. These sets correspond to the yellow regions in Figure 7.3.

For ease of notation, we drop the time dependence of the sets. The relation among these sets is then given in the following proposition.

Proposition 7.2. *The following set inclusion relations hold:*

$$\begin{aligned} &\underline{\mathbb{P}} \subseteq \hat{\mathbb{P}} \subseteq \mathbb{P}^\beta \subseteq \mathbb{P}, \\ &\underline{\mathbb{P}}_x \subseteq \hat{\mathbb{P}}_x \subseteq \mathbb{P}^\beta_x \subseteq \mathbb{P}_x, \\ &\underline{\mathbb{S}}_2 \subseteq \hat{\mathbb{S}}_2 \subseteq \mathbb{S}^\beta_2 \subseteq \mathbb{S}_2. \end{aligned}$$

Proof. Follows from the definitions. □

Remark 7.4. *Even though the risk coefficient sequence $\boldsymbol{\alpha}(k)$ at time step k is set to be 1, i.e., the most risky value, it follows from $\underline{\mathbb{S}}_2 \subseteq \hat{\mathbb{S}}_2$ that the risk-aware overtaking still provides more safety guarantee along the prediction horizon than control under the constant velocity assumption.*

In the next section, we will treat the risk coefficient sequence $\boldsymbol{\alpha}(k)$ as a decision variable to design a risk-aware overtaking controller.

7.5 Risk-Aware Optimal Overtaking

In this section, we formulate the risk-aware overtaking problem and then design a receding-horizon overtaking algorithm. We provide theoretical guaran-

tee of this algorithm and discuss how to approximately solve the risk-aware overtaking problem to speed up computations.

7.5.1 Risk-Aware Optimal Overtaking

At time step k, the risk-aware overtaking problem can be formulated as $\mathcal{P}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$:

$$
\begin{aligned}
&\min_{T\in\mathbb{N},\boldsymbol{u}_1,\boldsymbol{\alpha}(k)} \max_{i\in\mathbb{N}_{[0,T]}} \ \alpha(k+i|k) \\
&\quad \text{s.t. } \forall i \in \mathbb{N}_{[0,T-1]}: \\
&\quad\ \boldsymbol{x}_1(k+i+1|k) = f(\boldsymbol{x}_1(k+i|k), \boldsymbol{u}_1(k+i|k)), && \text{(7.11a)} \\
&\quad\ \boldsymbol{u}_1(k+i|k) \in \mathbb{U}_1(k+i|k), && \text{(7.11b)} \\
&\quad\ \ \forall i \in \mathbb{N}_{[0,T]}: \\
&\quad\ \beta(k+i|k) = 1-\alpha(k+i|k), && \text{(7.11c)} \\
&\quad\ \boldsymbol{x}_1(k+i|k) \in \mathbb{X}_1(k+i|k), && \text{(7.11d)} \\
&\quad\ \mathbb{S}_1(\boldsymbol{x}_1(k+i|k)) \cap \mathbb{O}_j = \emptyset, j=1,2, && \text{(7.11e)} \\
&\quad\ \mathbb{S}_1(\boldsymbol{x}_1(k+i|k)) \cap \mathbb{S}_2^{\beta}(k+i|k) = \emptyset, && \text{(7.11f)} \\
&\quad\ \begin{cases} p_1^x(k+T|k) \geq p_{2,\max}^{x,\beta}(k+T|k), \\ p_1^y(k+T|k) = d/2, \\ \theta_1(k+T|k) = 0, \end{cases} && \text{(7.11g)}
\end{aligned}
$$

where

$$
p_{2,\max}^{x,\beta}(k+T|k) = \max_{z\in\mathbb{P}_x^{\beta}(k+T|k)} z.
$$

The optimization problem aims to minimize the worst-case risk value over a horizon, subject to a feasible overtaking trajectory. The decision variables are the risk coefficient sequence $\boldsymbol{\alpha}(k)$, the overtaking horizon T, and the control sequence $\{\boldsymbol{u}_1(k+i|k)\}_{i=0}^{T-1}$.

Proposition 7.3. *The optimization problem $\mathcal{P}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ can be reformulated as $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$:*

$$
\begin{aligned}
&\min_{\substack{T\in\mathbb{N},\boldsymbol{u}_1,\boldsymbol{\alpha}(k),\\ \boldsymbol{\lambda}_j,\boldsymbol{\mu}_j,j=1,2,3}} \max_{i\in\mathbb{N}_{[0,T]}} \ \alpha(k+i|k) \\
&\quad \text{s.t. } (7.11a) - -(7.11d), (7.11g), (7.12), (7.13).
\end{aligned}
$$

$$
\begin{aligned}
&\forall i \in \mathbb{N}_{[0,T]}:\\
&\forall j = 1, 2,\\
&\begin{cases}
-g_1^T \mu_j(k+i|k) + (H_j p(\boldsymbol{x}_1(k+i|k)) - h_j)^T \lambda_j(k+i|k) > 0,\\
G_1^T \mu_j(k+i|k) + R^T(\boldsymbol{x}_1(k+i|k)) H_j^T \lambda_j(k+i|k) = 0,\\
\lambda_j(k+i|k) \geq 0, \mu_j(k+i|k) \geq 0, \|H_j^T \lambda_j(k+i|k)\| \leq 1,
\end{cases} \qquad (7.12)\\
&\begin{cases}
-g_1^T \mu_3(k+i|k) + (S^\beta(k+i|k) p(\boldsymbol{x}_1(k+i|k))\\
\qquad\qquad -s^\beta(k+i|k))^T \lambda_3(k+i|k) > 0,\\
G_1^T \mu_3(k+i|k) + R^T(\boldsymbol{x}_1(k+i|k)) S^\beta(k+i|k)^T \lambda_3(k+i|k) = 0,\\
\lambda_3(k+i|k) \geq 0, \mu_3(k+i|k) \geq 0, \|S^\beta(k+i|k)^T \lambda_3(k+i|k)\| \leq 1.
\end{cases} \qquad (7.13)
\end{aligned}
$$

Proof. The collision avoidance constraints (7.11e)–(7.11f) are equivalent to

$$
\begin{aligned}
&\operatorname{dist}(\mathbb{S}_1(k+i|k), \mathbb{O}_j) > 0, j = 1, 2,\\
&\operatorname{dist}(\mathbb{S}_1(k+i|k), \mathbb{S}_2^\beta(k+i|k)) > 0.
\end{aligned}
$$

Then, according to Lemma 7.2, we derive the constraints (7.12) and (7.13). □

Remark 7.5. *The essence of the optimization problem is to trade off the feasibility of automated overtaking and the collision risk along the prediction horizon. In general, the overtaking problem encounters feasibility issues if the reachable set $\mathbb{P}(k+i|k)$ is used to set up potential collision regions. The introduction of a risk coefficient sequence $\boldsymbol{\alpha}(k)$ could release a larger overtaking space, which improves feasibility. The problem $\mathcal{P}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ looks for a minimal risk coefficient that makes the overtaking feasible.*

Denote the optimal objective function of $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ as $\underline{\alpha}^*(k)$ and the corresponding optimal solution as T_k^*, $\{\boldsymbol{u}_1^*(k+i|k), i \in \mathbb{N}_{[0,T_k^*-1]}\}$, and $\{\alpha^*(k+i|i), i \in \mathbb{N}_{[0,T_k^*]}\}$. Then, $\underline{\alpha}^*(k)$ represents the minimal worst-case risk that will be taken along the prediction horizon T_k^*.

7.5.2 Risk-Aware Optimal Overtaking Algorithm

The risk-aware optimal overtaking algorithm is shown in Algorithm 7.1. Here, TerInd is an indicator to determine whether the while loop (line 2) terminates or not. When it terminates, there are three possible outputs: $\{\mathrm{Successful}, \mathrm{Infeasible}, \mathrm{Undecidable}\}$.

If the longitudinal position $p_1^x(k)$ of V_1 is in front of V_2, the lateral position $p_1^y(k)$ of V_1 is $d/2$, and the heading angle $\theta_1(k)$ of V_1 is 0, then Algorithm 7.1 terminates with an output $\mathrm{Successful}$.

Before the completion of the overtaking process, we solve the optimization problem $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ to decide the overtaking control command. The feasibility of $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ determines the possibility of automated overtaking. If $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ is infeasible, then whatever risk is imposed on the predictions, the enlarged overtaking region is still not enough to find a sequence of feasible control inputs. In this case, Algorithm 7.1 terminates with an output $\mathrm{Infeasible}$ and vehicle V_1 should stop the overtaking maneuver.

On the other hand, if $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ is feasible, the resulting solution will not be directly implemented. As shown in line 13, we use $\mathbb{S}_1(\boldsymbol{x}_1(k+1|k)) \cap \mathbb{S}_2(k+1|k) = \emptyset$ to detect whether the first input can ensure safety at next time step. If $\mathbb{S}_1(\boldsymbol{x}_1(k+1|k)) \cap \mathbb{S}_2(k+1|k) \neq \emptyset$, it implies that there exist some possible states $\boldsymbol{x}_2(k+1)$ in $\mathbb{P}(k+1|k)$ such that $\mathbb{S}_1(\boldsymbol{x}_1(k+1|k)) \cap \mathbb{S}_2(\boldsymbol{x}_1(k+1)) \neq \emptyset$. In this case, Algorithm 7.1 terminates with an output $\mathrm{Undecidable}$.

Remark 7.6. *Even though the initial optimal objective function $\underline{\alpha}^*(0)$ is not zero, i.e., there exists collision risk along the prediction horizon, successful overtaking without collision is still possible. Intuitively, this is because uncertainty about future positions of V_2 is reduced significantly as time proceeds.*

To check the feasibility of $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$, we define another optimization problem $\hat{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$:

$$
\begin{aligned}
&\min_{T\in\mathbb{N}, \boldsymbol{u}_1(k)} \quad T \\
&\quad \text{s.t. } (7.11a), (7.11b), (7.11d), (7.11e), \\
&\quad \forall i \in \mathbb{N}_{[0,T]} : \mathbb{S}_1(\boldsymbol{x}_1(k+i|k)) \cap \hat{\mathbb{S}}_2(k+i|k) = \emptyset, \qquad (7.14) \\
&\quad \begin{cases} p_1^x(k+T|k) \geq p_{2,\max}^x(k+T|k), \\ p_1^y(k+T|k) = d/2, \\ \theta_1(k+T|k) = 0, \end{cases} \qquad (7.15)
\end{aligned}
$$

Algorithm 7.1 Risk-Aware Optimal Overtaking Algorithm

Initialization: Set $k = 0$ and $\mathrm{TerInd} = 1$;
while TerInd **do**
  Measure $\boldsymbol{x}_1(k)$ and $\boldsymbol{x}_2(k)$;
  if $p_1^x(k) > p_2^x(k)$ & $p_1^y(k) = d/2$ & $\theta_1(k) = 0$ **then**
    $\mathrm{TerInd} = 0$;
    Output: Successful;
  else
    if $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ is infeasible **then**
      $\mathrm{TerInd} = 0$;
      Output: Infeasible;
    else
      Solve $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$;
      if $\mathbb{S}_1(\boldsymbol{x}_1(k+1|k)) \cap \mathbb{S}_2(k+1|k) = \emptyset$ **then**
        Implement $\boldsymbol{u}_1^*(k|k)$;
        $k = k + 1$;
      else
        $\mathrm{TerInd} = 0$;
        Output: Undecidable;
      end if
    end if
  end if
end while

where $p^x_{2,\max}(k+T|k) = \max \hat{\mathbb{P}}_x(k+T|k)$.

Proposition 7.4. *(*Feasibility*) The optimization problem $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ is feasible* if and only if *$\hat{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ is feasible.*

Proof. Sufficiency: Note that the sets $\hat{\mathbb{P}}(k+i|k)$, $\hat{\mathbb{P}}_x(k+i|k)$, and $\hat{\mathbb{S}}_2(k+i|k)$ are the corresponding sets $\mathbb{P}^\beta(k+i|k)$, $\mathbb{P}^\beta_x(k+i|k)$, and $\mathbb{S}^\beta_2(k+i|k)$ when $\alpha(k+i|k) = 1$ and $\beta(k+i|k) = 1 - \alpha(k+i|k) = 0$. Thus, the feasibility of $\hat{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ implies the feasibility of $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ when $\alpha(k+i|k) = 1, \forall i \in \mathbb{N}$.

Necessity: It follows from Proposition 7.2 that the satisfaction of the constraints (7.11f)–(7.11g) implies the satisfaction of (7.14)–(7.15). Thus, the feasibility of $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ implies the feasibility of $\hat{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$. □

From Proposition 7.4, we have that the feasibility of the min-max problem, $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$, can be checked by the feasibility of the time-optimal problem $\hat{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$. This is very useful and enables us to approximately solve $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$.

Under Algorithm 7.1, the overtaking maneuver is in the closed-loop manner. The closed-loop system for V_1 is

$$\boldsymbol{x}_1(k+1) = f(\boldsymbol{x}_1(k), \boldsymbol{u}^*_1(k|k)),$$

where $\boldsymbol{u}^*_1(k|k)$ is derived by the optimal solution to $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$.

Proposition 7.5. *(*Safety*) If Algorithm 7.1 terminates with the output* Successful*, the closed-loop overtaking trajectory is collision-free.*

Proof. In lines 13–19 of Algorithm 7.1, the collision avoidance between vehicles is guaranteed by $\mathbb{S}_1(\boldsymbol{x}_1(k+1|k)) \cap \mathbb{S}_2(k+1|k) = \emptyset$, i.e., the occupancy intersection of two vehicles at the next time step is empty. The feasibility of $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ implies that under the control of $\boldsymbol{u}^*_1(k|k)$, there is no collision between V_1 and the obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$. If the output of Algorithm 7.1 is Successful, the closed-loop overtaking trajectory is collision-free throughout the whole overtaking process. □

Proposition 7.6. *(*Successful Overtaking*) By implementing Algorithm 7.1, if the optimal objective function $\underline{\alpha}^*(k)$ of $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ is 0 at some time step k, Algorithm 7.1 will terminate with the output* Successful.

Proof. If $\underline{\alpha}^*(k) = 0$ at some time step k, it follows that there exist a time horizon T_k and a sequence of control inputs, $\{\boldsymbol{u}_1(k+i|k), i \in \mathbb{N}_{[0,T_k-1]}\}$, such that the constraints (7.11f)–(7.11g) are satisfied with $\alpha(k+i|k) = 0$, $\forall i \in \mathbb{N}_{[0,T_k]}$. At next time step $k+1$, with time horizon $T_k - 1$ and control inputs $\{\boldsymbol{u}_1(k+i+1|k), i \in \mathbb{N}_{[0,T_k-2]}\}$, the constraints (7.11f)–(7.11g) are still satisfied with $\alpha(k+1+i|k+1) = 0$, $\forall i \in \mathbb{N}_{[0,T_k-1]}$. By induction, it follows that the optimal solution of the problem $\tilde{\mathcal{P}}(\boldsymbol{x}_1(j), \boldsymbol{x}_2(j))$ is 0 for all $j \geq k$. We conclude that Algorithm 7.1 will terminate with the output Successful at most T_k steps. □

7.5.3 Approximate Solution

In general, it is difficult to exactly solve the min-max optimization problem, $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$. In this section, we provide an approximate solution, consisting of the following steps:

Step 1: Check whether there exists a time horizon T_k such that the optimization problem, $\hat{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$, is feasible. If yes, go to Step 2; otherwise, by Proposition 7.4, $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ is infeasible.

Step 2: For the time horizon T_k, iteratively reduce the value of the elements in the sequence $\{\alpha(k+i|k)\}_{i=0}^{T_k}$ from 1, until the problem $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ becomes infeasible. For computational efficiency, we can construct a sequence by quantizing the interval $[0, 1]$ into a finite number of elements and update $\alpha(k+i|k)$ from this sequence. Note that this sequence should include 0 and 1.

Step 3: Choose $T_k^* = T_k$, $\{\boldsymbol{u}(k+i|k)\}_{i=0}^{T_k^*-1}$, $\{\alpha^*(k+i|k)\}_{i=0}^{T_k^*}$ from the last iteration such that $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ is feasible. The optimal objective function $\underline{\alpha}^*(k)$ can be approximately obtained by $\underline{\alpha}^*(k) = \max_{i \in \mathbb{N}_{[0,T^*]}} \alpha^*(k+i|k)$.

7.6 Numerical Evaluations

This section provides simulated case studies to demonstrate the effectiveness of our theoretical results. The following numerical experiments are run on Matlab R2018b with ICLOCS2 toolbox [203].

7.6.1 Successful and Unsuccessful Overtakings

The parameters used in the simulated example are listed in Table 7.1, where l_i and w_i denote the length and width of vehicle V_i, respectively. We choose the

Table 7.1: Case study parameters

Lane	Vehicle 1	Vehicle 2
$v^R_{\min} = 0$ m/s	$\theta_{\min} = -\pi/18$ rad	$a^x_{2,\min} = -4$ m/s^2
$v^R_{\max} = 15$ m/s	$\theta_{\max} = \pi/18$ rad	$a^x_{2,\max} = 4$ m/s^2
$d = 5$ m	$a^x_{1,\min} = -4$ m/s^2	$l_2 = 4.7$ m
$\delta = 0.2$ s	$a^x_{1,\max} = 4$ m/s^2	$w_2 = 2$ m
	$\psi_{\min} = -0.6$ rad/s	
	$\psi_{\max} = 0.6$ rad/s	
	$L = 2.7$ m	
	$l_1 = 4.7$ m, $w_1 = 2$ m	

initial state: $\boldsymbol{x}_1(0) = [0; 1.5; 0; 8]$ and $\boldsymbol{x}_2(0) = [20; 3]$. To measure safety, we define the distance between the vehicles as

$$D(k) = \min_{\substack{\boldsymbol{z}_1 \in \mathbb{S}_1(\boldsymbol{x}_1(k)) \\ \boldsymbol{z}_2 \in \mathbb{S}_2(\boldsymbol{x}_2(k))}} \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|.$$

In the following, we implement Algorithm 7.1 and provide two overtaking scenarios: one results in a Successful output while the other results in an Infeasible output.

A successful overtaking attempt is shown in Figure 7.4, where V_1 finally arrives in front of V_2 through a set of sequential maneuvers: lane-changing, lane-keeping, and merging. Figure 7.4(a) shows the position trajectories of the two vehicles. In Figure 7.4(b)–(c), we show the yaw angle of V_1 and the longitudinal velocities, respectively. Here, the longitudinal velocity of V_1 is $v_1(k)\cos\theta_1(k)$. Note that the longitudinal velocity of V_2 increases. Both yaw angle and longitudinal velocities satisfy their corresponding constraints. In Figure 7.4(d)–(e), we show that the control inputs of the automated vehicle (steering angle and acceleration) satisfy the control input constraints. Furthermore, Figure 7.4(f) shows that the collision avoidance between two vehicles is guaranteed. In Figure 7.4(g), the minimal risk coefficients that are chosen to release the feasibility are provided. Although the initial risk coefficients are high (close to 1), as time proceeds, the uncertainty about the future positions of V_2 is reduced and thereby the risk coefficients decrease significantly (reaching 0 when $p^x_1 \approx 25$m). According to Proposition 7.6, Algorithm 7.1 terminates with output Successful.

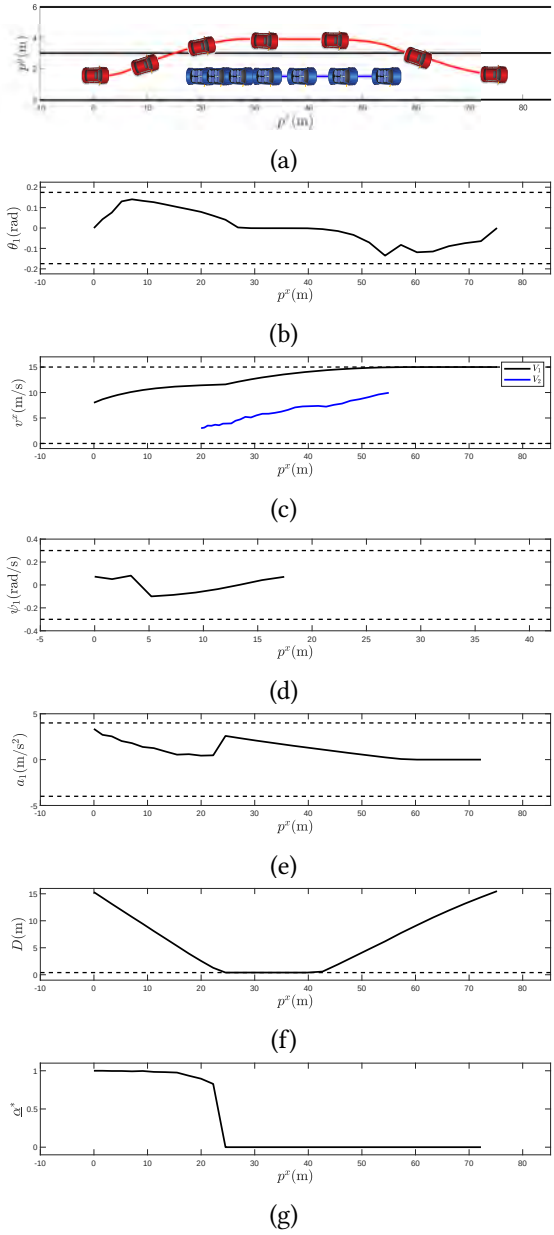


Figure 7.4: **Successful overtaking**: (a) position of two vehicles; (b) yaw angle θ_1 of V_1; (c) longitudinal velocities v_1 of two vehicles; (d) steering angle ψ_1 of V_1; (e) acceleration a_1 of V_1; (f) distance D between two vehicles; (g) risk coefficient $\underline{\alpha}^*$.

An infeasible overtaking attempt is shown in Figure 7.5. In Figure 7.5(a), we show the position trajectories of two vehicles. In Figure 7.5(b)–(c), we show the yaw angle of V_1 and longitudinal velocities of two vehicles, respectively. Both yaw angle and longitudinal velocities satisfy their corresponding constraints. Initially, since the problem $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ is feasible, the vehicle V_1 performs a lane change. However, due to the significant increase of the longitudinal velocity of V_2, the problem $\tilde{\mathcal{P}}(\boldsymbol{x}_1(k), \boldsymbol{x}_2(k))$ becomes infeasible when the longitudinal position of V_1 is about $5\mathrm{m}$. Then, the vehicle V_1 gives up the overtaking attempt and moves back to the previous lane. In Figure 7.5(d)–(e), we show that the control inputs of the automated vehicle (steering angle and acceleration) satisfy the control input constraints. Furthermore, Figure 7.5(f) shows that the collision avoidance between two vehicles is still guaranteed even though the overtaking attempt fails. The risk coefficients are shown in Figure 7.5(g) where they are set to 0 when vehicle V_1 moves back.

7.6.2 Comparison with the Constant Velocity Assumption

We run a Monte Carlo simulation to compare Algorithm 7.1 with receding horizon control with constant velocity prediction adapted from [180], [182]. We sample 100 realizations of state trajectories for V_2 with random inputs generated by $\boldsymbol{u}_2(k) = -2 + 6r$ but still guarantee the constraint satisfaction, where r is a uniformly sampled random variable in interval $[0, 1]$. For the 100 realizations, we implement two methods. The simulation results show that the probability of successful overtaking under Algorithm 7.1 is 93%. This is much higher than the 27% successful rate under receding horizon control with constant velocity prediction. One explanation for this is that the risk-aware optimal overtaking control takes into account the (partial) reachable sets of V_2 at each time step and such design exhibits more robustness than the control with constant velocity prediction. To illustrate, we detail and compare two realizations.

In the first realization, both control methods allow V_1 to safely overtake V_2, as shown in Figures 7.6–7.7. We can see that the state, control, and safety constraints are respected in this realization. From Figure 7.6(a) and Figure 7.7(a), we see that our risk-aware optimal overtaking control achieves shorter overtaking distance and time ($77.61\ \mathrm{m}$ and $6.2\ \mathrm{s}$) than the control with constant velocity prediction, ($100.30\ \mathrm{m}$ and $9.0\ \mathrm{s}$).

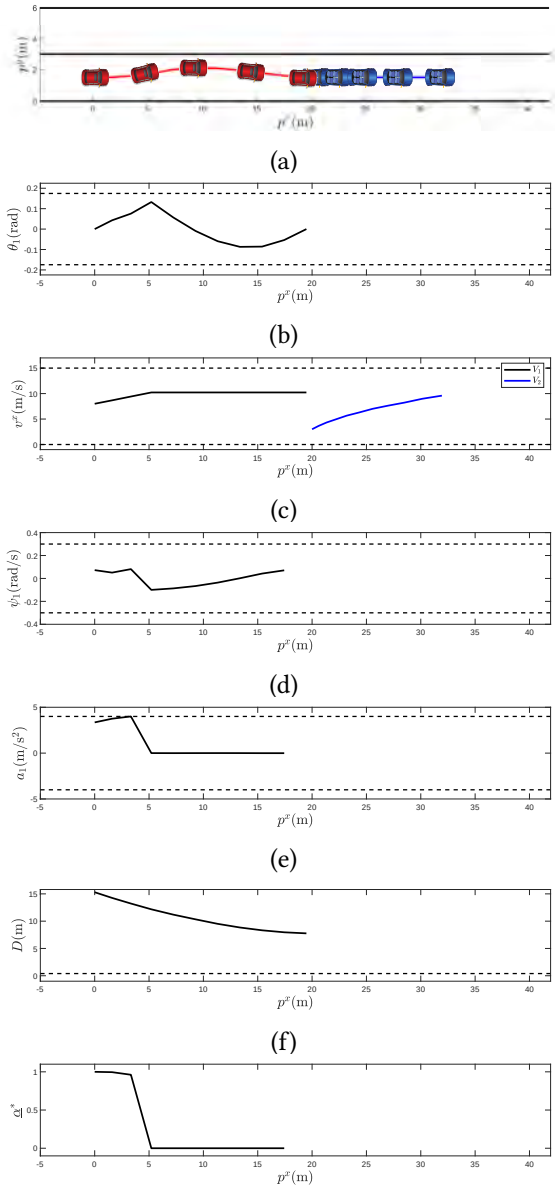


Figure 7.5: **Unsuccessful overtaking**: (a) position of two vehicles; (b) yaw angle θ_1 of V_1; (c) longitudinal velocities v_1 of two vehicles; (d) steering angle ψ_1 of V_1; (e) acceleration a_1 of V_1; (f) distance D between two vehicles; (g) risk coefficient $\underline{\alpha}^*$.

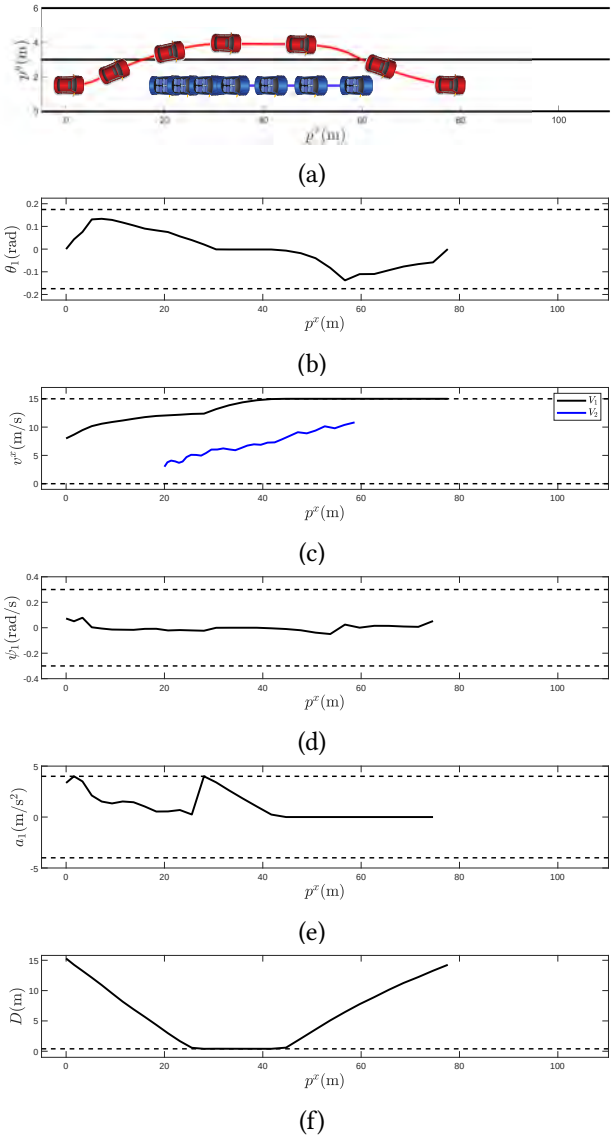


Figure 7.6: **Risk-aware optimal overtaking**, first realization: (a) position of two vehicles; (b) yaw angle θ_1 of V_1; (c) longitudinal velocities v_1 of two vehicles; (d) steering angle ψ_1 of V_1; (e) acceleration a_1 of V_1; (f) distance D between two vehicles.

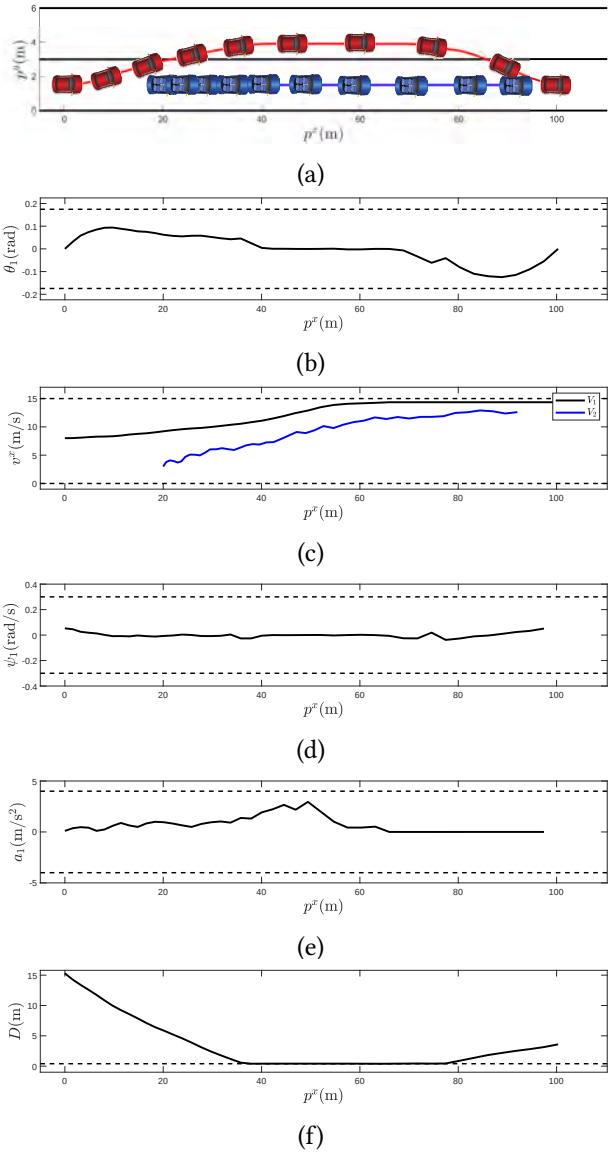


Figure 7.7: **Receding horizon control with constant velocity prediction**, first realization: (a) position of two vehicles; (b) yaw angle θ_1 of V_1; (c) longitudinal velocities v_1 of two vehicles; (d) steering angle ψ_1 of V_1; (e) acceleration a_1 of V_1; (f) distance D between two vehicles.

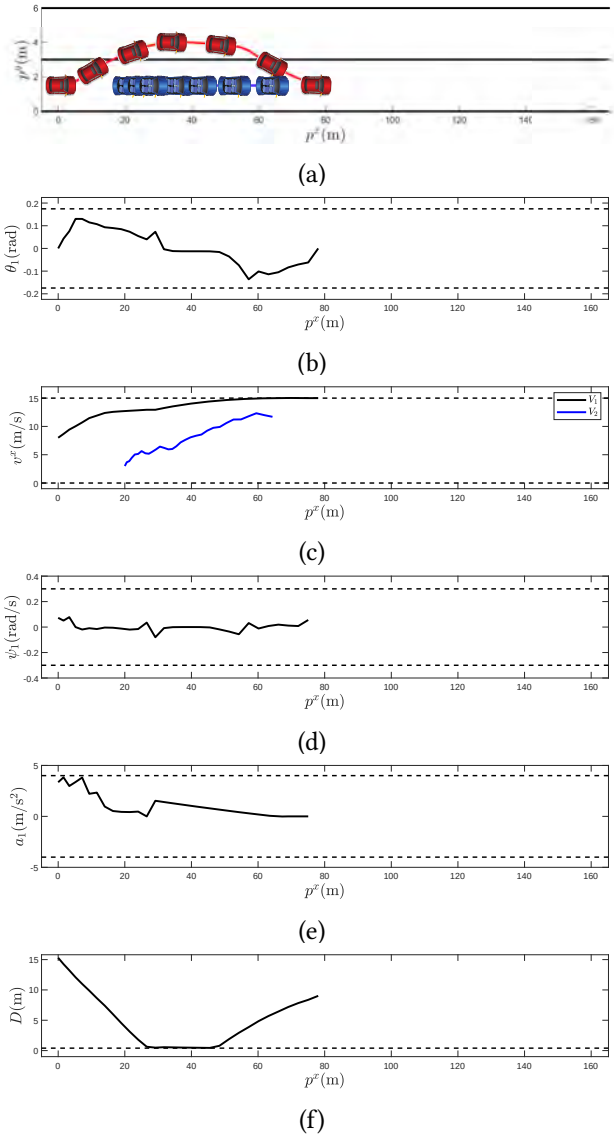


Figure 7.8: **Risk-aware optimal overtaking**, second realization: (a) position of two vehicles; (b) yaw angle θ_1 of V_1; (c) longitudinal velocities v_1 of two vehicles; (d) steering angle ψ_1 of V_1; (e) acceleration a_1 of V_1; (f) distance D between two vehicles.

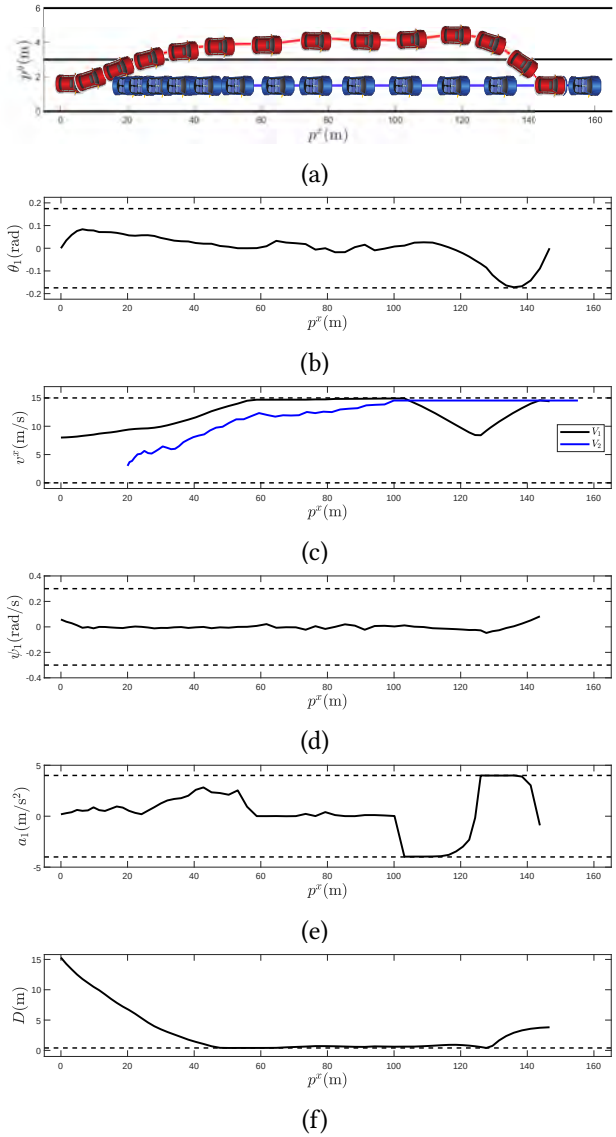


Figure 7.9: **Receding horizon control with constant velocity prediction**, second realization: (a) position of two vehicles; (b) yaw angle θ_1 of V_1; (c) longitudinal velocities v_1 of two vehicles; (d) steering angle ψ_1 of V_1; (e) acceleration a_1 of V_1; (f) distance D between two vehicles.

For the second realization, the risk-aware optimal overtaking control achieves successful overtaking as shown in Figure 7.8, while the control with constant velocity prediction is unable to perform a safe overtaking before the overtaking becomes infeasible as shown in Figure 7.9. In Figure 7.8, we show the state and control trajectories, which satisfy the constraints. We can see that V_1 is in front of V_2 when V_2 is at the x-position with $64.34\ \mathrm{m}$. At the same position, however, the vehicle V_1 is still moving in the other lane under the control with constant velocity prediction. Even when the velocity of V_2 reaches the maximal velocity (at about $102.28\ \mathrm{m}$), which implies that overtaking becomes infeasible, V_1 has no chance to merge back and fails to safely overtake V_2. Finally, V_1 gives up overtaking attempt and moves back to the previous lane, as shown in Figure 7.9(a).

7.7 Summary

In this chapter, we studied the overtaking problem where an automated vehicle tries to overtake a human-driven vehicle. Here, we did assume that the overtaken vehicle moves at a constant velocity, which might impose feasibility issues. To increase the possibility of feasible overtaking, we used martingale theory to perform a risk-aware reachability analysis by analytically characterizing the predicted collision probability. We designed a risk-aware optimal overtaking algorithm which can ensure collision avoidance during the whole overtaking process. Finally, we illustrated the effectiveness of the proposed algorithm in a simulated case study and compared our approach with the other approaches that has been suggested in the literature.

Chapter 8

Conclusions and Future Research

In this chapter, we summarize the main results of this thesis and outline possible directions for future research.

8.1 Conclusions

This thesis studied the problem of assuring safety for uncertain control systems performing complex tasks. More precisely, for such systems, we developed computational tools and verification and control synthesis algorithms, and evaluated these results on two applications.

In Chapter 3, we investigated the extension of set invariance to stochastic control systems. We proposed finite- and infinite-horizon probabilistic controlled invariant sets (PCISs), and provided some of their fundamental properties. We designed iterative algorithms to compute the PCIS within a given set. For systems with discrete state and control spaces, it was shown that finite- and infinite-horizon PCISs can be computed by solving an LP and an MILP at each iteration, respectively. We proved that the iterative algorithms are computationally tractable and can be terminated in a finite number of steps. For systems with continuous state and control spaces, we established an approximate algorithm for stochastic control systems and proved its convergence when computing finite-horizon PCIS. In addition, thanks to the sufficient conditions for the existence of infinite-horizon PCIS, we can compute an infinite-horizon PCIS by the stochastic backward reachable set from the robust controlled invariant set contained in it.

In Chapter 4, we considered some fundamental problems concerning the invariant cover for uncertain discrete-time linear control systems. We provided computationally tractable necessary and sufficient conditions on the

existence of an invariant cover, as well as upper and lower bounds on the minimal cardinality of the invariant cover. In addition, we gave an algorithm to compute an invariant cover in finite time. Numerical examples were given to illustrate the effectiveness of the results.

In Chapter 5, we studied linear temporal logic (LTL) model checking and control synthesis for discrete-time uncertain systems. Unlike automaton-based methods, our solutions build on the connection between LTL formulae and temporal logic trees (TLT) via reachability analysis. For a given transition system and LTL formula, we proved that the TLT provide an underapproximation or overapproximation for the LTL via minimal and maximal reachability analysis, respectively. We provided sufficient conditions and necessary conditions to the model checking problem. For a given controlled transition system and LTL formula, we showed that the TLT is an underapproximation for the LTL formula and thereby proposed an online control synthesis algorithm, under which a set of feasible control inputs is generated at each time step. We proved that this algorithm is recursively feasible. We also illustrated the effectiveness of the proposed methods through several examples.

In Chapter 6, we proposed a solution for a remote car parking problem. The parking task was specified as a set of LTL formulae. The framework made no assumptions about the operator's preference. Our system updates a data-driven belief of the operator's intent. We utilized the TLT-based approach to synthesize the control sets for LTL formulae. We proved recursive feasibility of the method, showing that the controller is always feasible and able to guarantee that the human will not be able to drive the system to an unsafe set. We illustrated the effectiveness of the proposed method by hardware experiments.

In Chapter 7, we studied the overtaking problem where an automated vehicle tried to overtake another human-driven vehicle. Here, we did not assume that the overtaken vehicle moves at a constant velocity, but assumed that the predicted velocity of the overtaken vehicle respects a supermartingale, meaning that its velocity is not increasing in expectation during the maneuver. We used martingale theory to perform a risk-aware reachability analysis by analytically characterizing the predicted collision probability. We designed a risk-aware optimal overtaking algorithm which can ensure collision avoidance. Finally, we illustrated the effectiveness of the proposed algorithm in a simulated case study and compared with other approaches in the literature.

8.2 Future Research Directions

There are several interesting research directions based on the work of this thesis. Here we discuss three proposals related to invariant cover, reachability analysis, and mixed human-machine systems.

Invariant Cover for Large-Scale Systems

Distributed control of large-scale systems relies on network communication. In Chapter 5, we have shown that invariant cover plays an important role for the design of coder-controller pairs with finite data-rate guarantees. Although there are many existing distributed control approaches under data-rate constraints [204], [205], quantization for set invariance is rarely taken into account. It would thus be interesting to extend the invariant cover for such systems to provide fundamental guarantees on how much information exchange is needed for a specific control invariance objective. Interesting problems not only concern the relation between the existence of an invariant cover and the network topology, but would also involve more efficient computational algorithms.

Real-time Reachability Analysis for Uncertain Systems

Traditional approaches on formal control synthesis, e.g., automaton-based approaches, cannot be implemented in real-time, in particular, when the environment is only partially known or dynamic. The TLT-based approaches in Chapter 5 are promising to avoid this restriction due to the recursive construction of the TLT. Since reachability analysis provides a bridge between the TLT and temporal logic formulae, real-time reachability could be used to extend our methods to online implementation. In the past few years, computation of reachable sets in real-time has become possible for certain deterministic systems [206], [207]. Built on these methods, it would be of great interest to study real-time reachability also for the classes of uncertain control systems in Chapter 5.

Scalable Synthesis for Multi-Human-Agent Systems

To deploy the self-driving vehicles in real traffic, it is important to develop approaches that allow multiple human and automated decision-makers. It would

be interesting to extend the results on human modeling and control synthesis in Chapters 6 and 7 to multi-human-agent traffic scenarios. New vehicle-to-vehicle and vehicle-to-infrastructure communication technologies enable many interesting scenarios for future studies. A fundamental question is how to make safe decisions for complex tasks in a distributed way.

Bibliography

[1] G. A. Bekey, *Autonomous Robots: From Biological Inspiration to Implementation and Control*. MIT press, 2005.

[2] "Innovation in autonomous systems", Royal Academy of Engineering, Tech. Rep., 2015. [Online]. Available: https://www.raeng.org.uk/publications/reports/innovation-in-autonomous-systems.

[3] A. C. Hernandez, *Twenty-fine astonishing self-driving car statistics for 2020*, 2020. [Online]. Available: https://policyadvice.net/car-insurance/insights/self-driving-car-statistics/.

[4] C. Jacobsson, K. Axelsson, P. O. Osterlind, and A. Norberg, "How people with stroke and healthy older people experience the eating process.", *Journal of clinical nursing*, vol. 9, no. 2, pp. 255–264, 2000.

[5] T. L. Mitzner, J. A. Sanford, and W. A. Rogers, "Closing the capacity-ability gap: Using technology to support aging with disability", *Innovation in Aging*, vol. 2, no. 1, 2018.

[6] C. Wu, "Learning and optimization for mixed autonomy systems–a mobility context", PhD thesis, University of California, Berkeley, 2018.

[7] D. A. Lazar, R. Pedarsani, K. Chandrasekher, and D. Sadigh, "Maximizing road capacity using cars that influence people", in *Proceedings of 2018 IEEE Conference on Decision and Control*, 2018, pp. 1801–1808.

[8] E. Bıyık, D. A. Lazar, R. Pedarsani, and D. Sadigh, "Altruistic autonomy: Beating congestion on shared roads", in *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*, 2018, pp. 887–904.

[9] E. Bıyık, D. A. Lazar, D. Sadigh, and R. Pedarsani, "The green choice: Learning and influencing human decisions on shared roads", in *Proceedings of 2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 347–354.

[10] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions.", in *Proceedings of Robotics: Science and Systems*, 2016.

[11] S. Nyholm and J. Smids, "Automated cars meet human drivers: Responsible human-robot coordination and the ethics of mixed traffic", *Ethics and Information Technology*, pp. 1–10, 2018.

[12] I. G. Jin, B. Schürmann, R. M. Murray, and M. Althoff, "Risk-aware motion planning for automated vehicle among human-driven cars", in *Proceedings of 2019 American Control Conference (ACC)*, 2019, pp. 3987–3993.

[13] J. Hu, M. Prandini, and S. Sastry, "Aircraft conflict prediction in the presence of a spatially correlated wind field", *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 326–340, 2005.

[14] J. Ding, M. Kamgarpour, S. Summers, A. Abate, J. Lygeros, and C. Tomlin, "A stochastic games framework for verification and control of discrete time stochastic hybrid systems", *Automatica*, vol. 49, no. 9, pp. 2665–2674, 2013.

[15] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Automatica probabilistic reachability and safety for controlled discrete time stochastic", *Automatica*, vol. 44, no. 11, pp. 2724–2734, 2008.

[16] T. Wongpiromsarn and R. M. Murray, "Formal verification of an autonomous vehicle system", 2008. [Online]. Available: http://www.cds.caltech.edu/~murray/preprints/wm08-cdc_s.pdf.

[17] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Synthesis of control protocols for autonomous systems", *Unmanned Systems*, vol. 1, no. 01, pp. 21–39, 2013.

[18] ——, "Receding horizon temporal logic planning", *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[19] J. Bhuiyan, *We rode in the self-driving cab that will hit Singapore streets in 2019*, 2016. [Online]. Available: https://www.vox.com/2016/5/17/11689064/nutonomy-self-driving-car-singapore-test.

[20] P. Sawers, *Einride demos a single teleoperator taking control of multiple autonomous trucks*, 2020. [Online]. Available: https://venturebeat.com/2020/04/07/einride-demos-a-single-teleoperator-taking-control-of-multiple-autonomous-trucks/.

[21] D. Wakabayashi, *Self-driving uber car kills pedestrian in Arizona, where robots roam*, 2020. [Online]. Available: https://www.einride.tech/news/einride-will-hire-its-first-remote-autonomous-truck-operator-in-2020.

[22] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge", *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.

[23] P. Koopman and F. Fratrik, "How many operational design domains, objects, and events?", in *Proceedings of Workshop on Artificial Intelligence Safety co-located with the Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[24] Y. Mulgaonkar, A. Makineni, L. Guerrero-Bonilla, and V. Kumar, "Robust aerial robot swarms without collision avoidance", *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 596–603, 2017.

[25] C. Liu, T. Tang, H.-C. Lin, and M. Tomizuka, *Designing Robot Behavior in Human-robot Interactions*. CRC Press, 2019.

[26] S. V. Raković, E. C. Kerrigan, K. I. Kouramas, and D. Q. Mayne, "Invariant approximations of the minimal robust positively invariant set", *IEEE Transactions on Automatic Control*, vol. 50, no. 3, pp. 406–410, 2005.

[27] M. Rungger and P. Tabuada, "Computing robust controlled invariant sets of linear systems", *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3665–3670, 2017.

[28] E. Gilbert and K. T. Tan, "Linear systems with state and control constraints: The theory and practice of maximal admissible sets", *IEEE Transactions on Automatic Control*, vol. 36, no. 9, pp. 1008–1020, 1991.

[29] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems", *IEEE Control Systems Magazine*, vol. 1, no. 21, pp. 84–99, 2001.

[30] A. Bemporad, M. Heemels, and M. Johansson, *Networked Control Systems*. Springer, 2010.

[31] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations", *IEEE Transactions on Automatic Control*, vol. 9, no. 49, pp. 1453–1464, 2004.

[32] M. Rungger and M. Zamani, "Invariance feedback entropy of nondeterministic control systems", in *Proceedings of ACM International Conference on Hybrid Systems: Computation and Control*, 2017, pp. 91–100.

[33] ——, "On the invariance feedback entropy of linear perturbed control systems", in *Proceedings of IEEE 56th Conference on Decision and Control*, 2017, pp. 3998–4003.

[34] M. S. Tomar, M. Rungger, and M. Zamani, *Invariance feedback entropy of uncertain control systems*, arXiv:1706.05242, 2017.

[35] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-time Dynamical Systems*. Springer, 2017.

[36] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT press, 2008.

[37] C. Fan, "Formal methods for safe autonomy: Data-driven verification, synthesis, and applications", PhD thesis, University of Illinois at Urbana-Champaign, 2019.

[38] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control", *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, 2013.

[39] J. Fu and U. Topcu, "Synthesis of shared autonomy policies with temporal logic specifications", *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 7–17, 2015.

[40] E. Fernández Cara and E. Zuazua Iriondo, "Control theory: History, mathematical achievements and perspectives", *Boletín de la Sociedad Española de Matemática Aplicada, 26, 79-140.*, 2003.

[41] S. Bennett, "A brief history of automatic control", *IEEE Control Systems Magazine*, vol. 16, no. 3, pp. 17–25, 1996.

[42] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1239–1258, 2009.

[43] D. Bertsekas, "Infinite time reachability of state-space regions by using feedback control", *IEEE Transactions on Automatic Control*, vol. 17, no. 5, pp. 604–613, 1972.

[44] D. P. Bertsekas and I. B. Rhodes, "On the minimax reachability of target sets and target tubes", *Automatica*, vol. 7, no. 2, pp. 233–247, 1971.

[45] I. Kolmanovsky and E. G. Gilbert, "Theory and computation of disturbance invariant sets for discrete-time linear systems", *Mathematical problems in engineering*, vol. 4, 1998.

[46] F. Blanchini, "Set invariance in control", *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[47] F. Blanchini and S. Miani, *Set-theoretic Methods in Control*. Springer, 2007.

[48] L. M. Bujorianu, *Stochastic Reachability Analysis of Hybrid Systems*. Springer, 2012.

[49] B. Ghosh and P. S. Duggirala, "Robust reachable set: Accounting for uncertainties in linear dynamical systems", *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5, pp. 1–22, 2019.

[50] E. C. Kerrigan, "Robust constraint satisfaction: Invariant sets and predictive control", PhD thesis, University of Cambridge, 2001.

[51] A. Abate, "Probabilistic reachability for stochastic hybrid systems: Theory, computations, and applications", PhD thesis, University of California, Berkeley, 2007.

[52] E. Kofman, J. A. De Doná, and M. M. Seron, "Probabilistic set invariance and ultimate boundedness", *Automatica*, vol. 48, no. 10, pp. 2670–2676, 2012.

[53] E. Kofman, J. A. De Doná, M. M. Seron, and N. Pizzi, "Continuous-time probabilistic ultimate bounds and invariant sets: Computation and assignment", *Automatica*, vol. 71, pp. 98–105, 2016.

[54] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala, "Automatic reachability analysis for nonlinear hybrid models with c2e2", in *Proceedings of International Conference on Computer Aided Verification*, 2016, pp. 531–538.

[55] E. Coelingh, L. Jakobsson, H. Lind, and M. Lindman, "Collision warning with auto brake: A real-life safety perspective", Innovations for Safety: Opportunities and Challenges, Tech. Rep., 2007.

[56] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars", PhD thesis, Technische Universität München, 2010.

[57] K. Leung, E. Schmerling, M. Zhang, M. Chen, J. Talbot, J. C. Gerdes, and M. Pavone, "On infusing reachability-based safety assurance within planning frameworks for human–robot vehicle interactions", *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1326–1345, 2020.

[58] K. Hashimoto and D. V. Dimarogonas, "Synthesizing communication plans for reachability and safety specifications", *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 561–576, 2019.

[59] T. Dang and O. Maler, "Reachability analysis via face lifting", in *Proceedings of International Workshop on Hybrid Systems: Computation and Control*, 1998, pp. 96–109.

[60] E. Asarin, O. Bournez, T. Dang, and O. Maler, "Approximate reachability analysis of piecewise-linear dynamical systems", in *Proceedings of International Workshop on Hybrid Systems: Computation and Control*, 2000, pp. 20–31.

[61] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0", in *Proceedings of European Control Conference*, 2013, pp. 502–510.

[62] I. M. Mitchell and J. A. Templeton, "A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems", in *Proceedings of ACM International Conference on Hybrid Systems: Computation and Control*, 2005, pp. 480–494.

[63] A. P. Vinod, J. D. Gleason, and M. M. Oishi, "Sreachtools: A matlab stochastic reachability toolbox", in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 33–38.

[64] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, and C. Schilling, "Juliareach: A toolbox for set-based reachability", in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 39–44.

[65] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems", *IEEE Transactions on Automatic Control*, vol. 5, no. 52, pp. 782–798, 2007.

[66] M. Zamani, G. Pola, M. Mazo, and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions", *IEEE Transactions on Automatic Control*, vol. 7, no. 57, pp. 1804–1809, 2012.

[67] P. Yu and D. V. Dimarogonas, "Approximately symbolic models for a class of continuous-time nonlinear systems", in *Proceedings of 58th IEEE Conference on Decision and Control*, 2019, pp. 4349–4354.

[68] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[69] R. Alur, *Principles of Cyber-Physical Systems*. MIT Press, 2015.

[70] A. Ulusoy and C. Belta, "Receding horizon temporal logic control in dynamic environments", *The International Journal of Robotics Research*, vol. 12, no. 33, pp. 1593–1607, 2014.

[71] M. Guo, J. Tumová, and D. V. Dimarogonas, "Communication-free multi-agent control under local temporal tasks and relative-distance constraints", *IEEE Transactions on Automatic Control*, vol. 12, no. 61, pp. 3948–3962, 2016.

[72] S. Coogan, E. A. Gol, M. Arcak, and C. Belta, "Traffic network control from temporal logic specifications", *IEEE Transactions on Control of Network Systems*, vol. 2, no. 3, pp. 162–171, 2016.

[73] X. Ding, M. Lazar, and C. Belta, "LTL receding horizon control for finite deterministic systems", *Automatica*, vol. 2, no. 50, pp. 399–408, 2014.

[74] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Hierarchical LTL-task MDPs for multi-agent coordination through auctioning and learning", 2019. [Online]. Available: http://kth.diva-portal.org.

[75] L. Lindemann and D. V. Dimarogonas, "Feedback control strategies for multi-agent systems under a fragment of signal temporal logic tasks", *Automatica*, vol. 106, pp. 284–293, 2019.

[76] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: verification of probabilistic real-time systems", in *Proceedings of 23rd International Conference on Computer Aided Verification*, 2011, pp. 585–591.

[77] R. C. Goertz, "Master-slave manipulator", Tech. Rep., 1949. [Online]. Available: https://www.osti.gov/servlets/purl/1054625.

[78] M. Minsky, "Telepresence: A manifesto", *IEEE Spectrum*, 2010.

[79] A. Fagg, M. Rosenstein, R. Platt, and R. Grupen, "Extracting user intent in mixed initiative teleoperator control", in *Proceedings of AIAA Intelligent Systems Technical Conference*, 2004.

[80] J. Kofman, X. Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface", *IEEE transactions on industrial electronics*, vol. 52, no. 5, pp. 1206–1219, 2005.

[81] D.-J. Kim, R. Hazlett-Knudsen, H. Culver-Godfrey, G. Rucks, T. Cunningham, D. Portee, J. Bricout, Z. Wang, and A. Behal, "How autonomy impacts performance and satisfaction: Results from a study with spinal cord injured subjects using an assistive robot", *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 42, no. 1, pp. 2–14, 2011.

[82] M. Cubuktepe, N. Jansen, M. Alshiekh, and U. Topcu, "Synthesis of provably correct autonomy protocols for shared control", *IEEE Transactions on Automatic Control*, 2020.

[83] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning", in *Proceedings of AAAI Conference on Artificial Intelligence*, 2008, pp. 1433–1438.

[84] S. Reddy, A. D. Dragan, and S. Levine, "Shared autonomy via deep reinforcement learning", in *Proceedings of Robotics: Science and Systems*, 2018.

[85] C. Sentouh, A.-T. Nguyen, M. A. Benloucif, and J.-C. Popieul, "Driver-automation cooperation oriented approach for shared control of lane keeping assist systems", *IEEE Transactions on Control Systems Technology*, vol. 27, no. 5, pp. 1962–1978, 2018.

[86] M. Marcano, S. Díaz, J. Pérez, and E. Irigoyen, "A review of shared control for automated vehicles: Theory and applications", *IEEE Transactions on Human-Machine Systems*, 2020.

[87] W. Wang, X. Na, D. Cao, J. Gong, J. Xi, Y. Xing, and F.-Y. Wang, "Decision-making in driver-automation shared control: A review and perspectives", *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 5, pp. 1289–1307, 2020.

[88] K. Brown, K. Driggs-Campbell, and M. J. Kochenderfer, "Modeling and prediction of human driver behavior: A survey", *arXiv:2006.08832*, 2020.

[89] D. P. Bertsekas and S. Shreve, *Stochastic Optimal Control: the Discrete-time Case*. Athena Scientific, 2004.

[90] M. B. Stinchcombe and H. White, "Some measurability results for extrema of random functions over random sets", *The Review of Economic Studies*, vol. 59, no. 3, pp. 495–514, 1992.

[91] A. Tarski, "A lattice-theoretical fixpoint theorem and its application", *Pacific Journal of Mathematics*, no. 5, pp. 285–309, 1955.

[92] I. M. Mitchell, S. Kaynama, M. Chen, and M. Oishi, "Safety preserving control synthesis for sampled data systems", *Nonlinear Analysis: Hybrid Systems*, vol. 10, pp. 63–82, 2013.

[93] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research", *IEEE Control Systems*, vol. 36, no. 6, pp. 30–44, 2016.

[94] M. Cannon, B. Kouvaritakis, S. Raković, and Q. Cheng, "Stochastic tubes in model predictive control with probabilistic constraints", *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 194–200, 2011.

[95] M. A. Hernández-Mejías, A. Sala, C. Ariño, and A. Querol, "Reliable controllable sets for constrained markov-jump linear systems", *International Journal of Robust and Nonlinear Control*, vol. 10, no. 26, pp. 2075–2089, 2016.

[96] M. A. Hernández-Mejías and A. Sala, "Reliability and time-to-failure bounds for discrete-time constrained markov jump linear systems", *International Journal of Robust and Nonlinear Control*, vol. 10, no. 27, pp. 1773–1791, 2017.

[97] J. Burlet, O. Aycard, and T. Fraichard, "Robust motion planning using markov decision processes and quadtree decomposition", in *Proceedings of IEEE Conference on Robotics and Automation*, 2004, pp. 2820–2825.

[98] S. Battilotti and A. De Santis, "Stabilization in probability of nonlinear stochastic systems with guaranteed region of attraction and target set", *IEEE Transactions on Automatic Control*, vol. 48, no. 9, pp. 1585–1599, 2003.

[99] G. Pola, J. Lygeros, and M. D. Di Benedetto, "Invariance in stochastic dynamical control systems", in *Proceedings of the 17th International symposium on Mathematical Theory of Network and Systems*, 2006.

[100] G. Pola and G. Pola, "A stochastic reachability approach to portfolio construction in finance industry", *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 189–195, 2012.

[101] M. Cannon, B. Kouvaritakis, and X. Wu, "Probabilistic constrained MPC for multiplicative and additive stochastic uncertainty", *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1626–1632, 2009.

[102] N. A. Nguyen, "Stochastic output feedback control: Convex lifting approach", *Automatica*, vol. 20, no. 1, pp. 212–220, 2018.

[103] L. Hewing, A. Carron, K. Wabersich, and M. Zeilinger, "On a correspondence between probabilistic and robust invariant sets for linear systems", in *Proceedings of 2018 European Control Conference*, 2018, pp. 876–881.

[104] S. Amin, A. Abate, M. Prandini, S. Sastry, and J. Lygeros, "Reachability analysis for controlled discrete time stochastic hybrid systems", in *Proceedings of International Workshop on Hybrid Systems: Computation and Control*, 2014, pp. 49–63.

[105] C. S. Chow and J. N. Tsitsiklis, "An optimal one-way multigrid algorithm for discrete-time stochastic control", *IEEE Transactions on Automatic Control*, vol. 36, no. 8, pp. 898–914, 1991.

[106] A. Bhattacharya and J. P. Kharoufeh, "Linear programming formulation for non-stationary, finite-horizon markov decision process models", *Operations Research Letters*, vol. 45, no. 6, pp. 570–574, 2017.

[107] N. Megiddo, "Linear programming in linear time when the dimension is fixed", *Journal of the ACM*, vol. 31, no. 1, pp. 114–127, 1984.

[108] I. Tkachev and Abate, "On infinite-horizon probabilistic properties and stochastic bisimulation functions", in *Proceedings of the 50th IEEE Conference on Decision and Control and 2011 European Control Conference*, 2011, pp. 526–531.

[109] D. Bertsekas, *Dynamic Programming and Optimal Control: vol II*. Athena Scientific, 2012.

[110] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Courier Corporation, 1998.

[111] J. T. Linderoth and A. Lodi, "MILP software", *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[112] P. Bonami, M. Kilinç, and J. Linderoth, "Algorithms and software for convex mixed integer nonlinear programs", in *Mixed Integer Nonlinear Programming*, J. Lee and S. Leyffer, Eds., Springer, 2012, pp. 1–39.

[113] M. Prandini and J. Hu, "A stochastic approximation method for reachability computations", in *Stochastic Hybrid Systems*, H. A. Blom and J. Lygeros, Eds., 107-139, 2006, pp. 1–39.

[114] B. Kouvaritakis, M. Cannon, S. Raković, and Q. Cheng, "Explicit use of probabilistic distributions in linear predictive control", *Automatica*, vol. 46, no. 10, pp. 1719–1724, 2010.

[115] M. Lorenzen, F. Dabbene, R. Tempo, and F. Allgöwer, "Constraint-tightening and stability in stochastic model predictive control", *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3165–3177, 2017.

[116] A. Prékopa, *Stochastic Programming*. Springer, 2013.

[117] M. Guo and M. M. Zavlanos, "Probabilistic motion planning under temporal tasks and soft constraints", *IEEE Transactions on Automatic Control*, vol. 63, no. 12, pp. 4051–4066, 2018.

[118] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, and T. Weng, "Occupancy-driven energy management for smart building automation", in *Proceedings of the 2nd ACM workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, 2010, pp. 1–6.

[119] B. Zhou, J. Cao, X. Zeng, and H. Wu, "Adaptive traffic light control in wireless sensor network-based intelligent transportation system", in *Proceedings of IEEE 72nd Vehicular Technology Conference-Fall*, 2010, pp. 1–5.

[120] G. N. Nair, F. Fagnani, S. Zampieri, and R. J. Evans, "Feedback control under data rate constraints: An overview", *Proc. of the IEEE*, vol. 1, no. 95, pp. 108–137, 2007.

[121] G. N. Nair and R. J. Evans, "Exponential stabilisability of finite-dimensional linear systems with limited data rates", *Automatica*, vol. 4, no. 39, pp. 585–593, 2003.

[122] S. Tatikonda and S. Mitter, "Control under communication constraints", *IEEE Transactions on Automatic Control*, vol. 7, no. 49, pp. 1056–1068, 2004.

[123] J. Lunze, *Control Theory of Digitally Networked Dynamic Systems*. Springer, 2014.

[124] G. N. Nair, R. J. Evans, I. M. Y. Mareels, and W. Moran, "Topological feedback entropy and nonlinear stabilization", *IEEE Transactions on Automatic Control*, vol. 9, no. 49, pp. 1585–1597, 2004.

[125] R. L. Adler, A. G. Konheim, and M. H. McAndrew, "Topological entropy", *Transactions of the American Mathematical Society*, vol. 2, no. 114, pp. 309–319, 1965.

[126] R. Bowen, "Entropy for group endomorphisms and homogeneous spaces", *Transactions of the American Mathematical Society*, no. 153, 401–414, 1971.

[127] C. Kawan, *Invariance Entropy for Deterministic Control Systems: An Introduction*. Springer, 2013.

[128] F. Colonius, C. Kawan, and G. N. Nair, "A note on topological feedback entropy and invariance entropy", *Systems & Control Letters*, vol. 5, no. 62, pp. 377–381, 2013.

[129] T. Gal, *Postoptimal Analyses, Parametric Programming, and Related Topics (2nd ed.)* Berlin: de Gruyter, 1995.

[130] M. Henk, J. Richter-Gebert, and G. M. Ziegler, "Basic properties of convex polytopes", in *Handbook of Discrete and Computational Geometry (3rd ed.)* CRC Press, 2017.

[131] R. Schneider, "Convex bodies: The Brunn–Minkowski Theory: 2nd expanded edition", in *Encyclopedia Math. Appl., v. 151*, Cambridge University Press, 2014.

[132] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems", *Automatica*, vol. 1, no. 38, pp. 3–20, 2002.

[133] P. M. Gruber and J. M. Wills, *Handbook of Convex Geometry*. North Holland, 1993.

[134] D. Avis, "Lrs: A revised implementation of the reverse search vertex enumeration algorithm", in *Polytopes - Combinatorics and Computation*, G. Kalai and G. Ziegler, Eds., Birkhauser-Verlag, 2000, pp. 177–198.

[135] S. Lai, M. Lan, and B. M. Chen, "Model predictive local motion planning with boundary state constrained primitives", *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3577–3584, 2019.

[136] K. J. Åström and B. Wittenmark, *Computer-controlled Systems: Theory and Design*. Courier Corporation, 2013.

[137] M. Y. Vardi, "An automata-theoretic approach to linear temporal logic", in *Logics for Concurrency*, F. Moller and G. Birtwistle, Eds., Springer, 1996, pp. 238–266.

[138] M. O. Rabin, "Decidability of second-order theories and automata on infinite trees", *Transactions of the American Mathematical Society*, no. 141, pp. 1–35, 1969.

[139] E. A. Emerson, "Automata, tableaux, and temporal logics", in *Proceedings of Workshop on Logic of Programs*, 1985, pp. 79–88.

[140] N. Piterman and A. Pnueli, "Faster solutions of rabin and streett games", in *Proceedings of 21st Annual IEEE Symposium on Logic in Computer Science*, 2006, pp. 275–284.

[141] F. Horn, "Streett games on finite graphs", in *Proceedings of 2nd Workshop on Games in Design and Verification*, 2005.

[142] A. Girard, G. Pola, and G. J. Pappas, "Approximately bisimilar symbolic models for incrementally stable switched systems", *IEEE Transactions on Automatic Control*, vol. 1, no. 55, pp. 116–126, 2010.

[143] M. Zamani, P. M. Esfahani, R. Majumdar, A. Abate, and J. Lygeros, "Symbolic control of stochastic systems via approximately bisimilar finite abstractions", *IEEE Transactions on Automatic Control*, vol. 12, no. 59, pp. 3135–3150, 2014.

[144] P. Tabuada and G. J. Pappas, "Model checking LTL over controllable linear systems is decidable", in *Proceedings of ACM International Conference on Hybrid Systems: Computation and Control*, 2003, pp. 498–513.

[145] B. Yordanov, J. Tumová, I. Černá, J. Barnat, and C. Belta, "Formal analysis of piecewise affine systems through formula guided refinement", *Automatica*, vol. 1, no. 49, pp. 261–266, 2013.

[146] ——, "Temporal logic control of discrete-time piecewise affine systems", *IEEE Transactions on Automatic Control*, vol. 6, no. 57, pp. 1491–1504, 2012.

[147] P.-J. Meyer and D. V. Dimarogonas, "Hierarchical decomposition of LTL synthesis problem for nonlinear control systems", *IEEE Transactions on Automatic Control*, vol. 11, no. 64, pp. 4676–4683, 2019.

[148] S. Haesaert and S. Soudjani, "Robust dynamic programming for temporal logic control of stochastic systems", 2018. [Online]. Available: arXiv:1811.11445.

[149] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications", in *Proceedings of 47th IEEE Conference on Decision and Control*, 2008, pp. 2117–2122.

[150] N. Cauchi and A. Abate, "StocHy-automated verification and synthesis of stochastic processes", in *Proceedings of ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 258–259.

[151] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning", *IEEE Transactions on Automatic Control*, vol. 11, no. 57, pp. 2817–2830, 2012.

[152] C. Belta, "Formal synthesis of control strategies for dynamical systems", in *Proceedings of 55th IEEE Conference on Decision and Control*, 2016, pp. 3407–3431.

[153] P. G. Sessa, D. Frick, T. A. Wood, and M. Kamgarpour, "From uncertainty data to robust policies for temporal logic planning", in *Proceedings of ACM International Conference on Hybrid Systems: Computation and Control*, 2018, pp. 157–166.

[154] K. Hashimoto and D. V. Dimarogonas, "Resource-aware networked control systems under temporal logic specifications", *Discrete Event Dynamic Systems*, 2019.

[155] M. Inoue and V. Gupta, "'Weak' control for human-in-the-loop systems", *IEEE Control Systems Letters*, vol. 3, no. 2, pp. 440–445, 2018.

[156] Y. Gao, F. J. Jiang, X. Ren, L. Xie, and K. H. Johansson, "Reachability-based human-in-the-loop control with uncertain specifications", in *Proceedings of 21st IFAC World Congress*, to appear, 2020.

[157] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems", *Automatica*, vol. 3, no. 35, pp. 349–370, 1999.

[158] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin, "Decomposition of reachable sets and tubes for a class of nonlinear systems", *IEEE Transactions on Automatic Control*, vol. 11, no. 63, pp. 3675–3688, 2018.

[159] M. Althoff and B. H. Krogh, "Reachability analysis of nonlinear differential-algebraic systems", *IEEE Transactions on Automatic Control*, vol. 2, no. 59, pp. 371–383, 2014.

[160] I. M. Mitchell, "Scalable calculation of reach sets and tubes for nonlinear systems with terminal integrators: a mixed implicit explicit formulation", in *Proceedings of ACM International Conference on Hybrid Systems: Computation and Control*, 2011, pp. 103–112.

[161] M. Chen, Q. Tam, S. C. Livingston, and M. Pavone, "Signal temporal logic meets Hamilton-Jacobi reachability: connections and applications", in *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*, 2018. [Online]. Available: http://asl.stanford.edu/wp-content/papercite-data/pdf/Chen.Tam.Livingston.Pavone.WAFR18.pdf.

[162] E. A. Gol, M. Lazar, and C. Belta, "Language-guided controller synthesis for linear systems", *IEEE Transactions on Automatic Control*, vol. 5, no. 59, pp. 1163 –1176, 2014.

[163] INRIX, "The impact of parking pain in the us, uk and germany", Tech. Rep., 2017. [Online]. Available: http://www2.inrix.com/research-parking-2017.

[164] Hikvision. (2018). Mobile robot solutions: Smart parking industry solutions, [Online]. Available: http://en.hikrobotics.com/robot/robotplaninfo.htm?type=546\&oid=1598.

[165] D. AG. (2020). Innovative parking solutions: Convenient, customer-oriented and efficient, [Online]. Available: https://www.daimler.com/innovation/parking.html.

[166] B. AG. (2018). Intelligent parking, [Online]. Available: https://www.bmw.co.uk/bmw-ownership/connecteddrive/driver-assistance/intelligent-parking\#gref.

[167] X. Zhang, A. Liniger, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance", in *Proceedings of 57th IEEE Conference on Decision and Control*, 2018, pp. 4327–4332.

[168] Y. Li, K. H. Johansson, and J. Mårtensson, "A hierarchical control system for smart parking lots with automated vehicles: Improve efficiency by leveraging prediction of human drivers", in *Proceedings of 18th European Control Conference*, 2019, pp. 2675–2681.

[169] X. Shen, X. Zhang, and F. Borrelli, "Autonomous parking of vehicle fleet in tight environments", 2019. [Online]. Available: https://arxiv.org/abs/1910.02349.

[170] Wired. (2020). The war to remotely control self-driving cars heats up, [Online]. Available: https://www.wired.com/story/designated-driver-teleoperations-self-driving-cars/.

[171] M. Guo, S. Andersson, and D. Dimarogonas, "Human-in-the-loop mixed-initiative control under temporal tasks", in *Proceedings of IEEE International Conference on Robotics and Automation*, 2018, pp. 6395–6400.

[172] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2004.

[173] G. Fainekos, H. Kress-Gazit, and G. Pappas., "Temporal logic motion planning for mobile robots", in *Proceedings of IEEE International Conference on Robotics and Automation*, 2005, pp. 2020–2025.

[174] P. Tabuada and G. Pappas, "Linear time logic control of discrete-time linear systems", *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[175] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications", *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[176] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding", in *Proceedings of 32rd AAAI Conference on Artificial Intelligence*, 2018.

[177] S. Dixit, U. Montanaro, S. Fallah, M. Dianati, D. Oxtoby, T. Mizutani, and A. Mouzakitis, "Trajectory planning for autonomous high-speed overtaking using mpc with terminal set constraints", in *Proceedings of 21st IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 1061–1068.

[178] S. Dixit, S. Fallah, U. Montanaro, M. Dianati, A. Stevens, F. Mccullough, and A. Mouzakitis, "Trajectory planning and tracking for autonomous overtaking : State-of-the- art and future prospects trajectory planning and tracking for autonomous overtaking : State-of-the-art and future prospects", *Annual Reviews in Control*, vol. 45, pp. 76–86, 2018.

[179] N. Murgovski and J. Sjöberg, "Predictive cruise control with autonomous overtaking", in *Proceedings of 54th IEEE Conference on Decision and Control*, 2015, pp. 644–649.

[180] J. Karlsson, N. Murgovski, and J. Sjöberg, "Temporal vs. spatial formulation of autonomous overtaking algorithms", in *Proceedings of 19th IEEE International Conference on Intelligent Transportation Systems*, 2016, pp. 1029–1034.

[181] F. Molinari, N. N. Anh, and L. D. Re, "Efficient mixed integer programming for autonomous overtaking", in *Proceedings of IEEE American Control Conference*, 2017, pp. 2303–2308.

[182] J. Karlsson, N. Murgovski, and J. Sjöberg, "Comparison between mixed-integer and second order cone programming for autonomous overtaking", in *Proceedings of IEEE 2018 European Control Conference*, 2018, pp. 386–392.

[183] B. Vanholme, D. Gruyer, B. Lusetti, S. Glaser, and S. Mammar, "Highly automated driving on highways based on legal safety", *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 333–347, 2013.

[184] J. Chen, W. Zhan, and M. Tomizuka, "Autonomous driving motion planning with constrained iterative lqr", *IEEE Transactions on Intelligent Vehicles*, 2019.

[185] A. Raghavan, J. Wei, J. S. Baras, and K. H. Johansson, "Stochastic control formulation of the car overtake problem", *IFAC-PapersOnLine*, vol. 51, no. 9, pp. 124–129, 2018.

[186] M. Althoff and J. M. Dolan, "Set-based computation of vehicle behaviors for the online verification of autonomous vehicles", in *Proceedings of 21st IEEE International Conference on Intelligent Transportation Systems*, 2011, pp. 1162–1167.

[187] M. Althoff, O. Stursberg, and M. Buss, "Safety assessment of autonomous cars using verification techniques", in *Proceedings of American Control Conference*, 2007, pp. 4154–4159.

[188] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis", *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[189] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving", *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.

[190] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for cars that coordinate with people: Leveraging effects on human actions for planning and active information gathering over human internal state", *Autonomous Robots*, vol. 42, no. 7, pp. 1405–1426, 2018.

[191] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning", in *Proceedings of 21st International Conference on Machine Learning*, 2004, p. 1.

[192] K. Driggs-Campbell, R. Dong, and R. Bajcsy, "Robust, informative human-in-the-loop predictions via empirical reachable sets", *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 300–309, 2018.

[193] Y. Gao, F. Jiang, K. H. Johansson, and L. Xie, "Stochastic modeling and optimal control for automated overtaking", in *Proceedings of 58th IEEE Conference on Decision and Control*, 2019, pp. 1273–1278.

[194] D. Duffie, *Dynamic Asset Pricing Theory*. Princeton University Press, 2010.

[195] M. H. A. Davis, "Martingale methods in stochastic control", in *Stochastic control theory and stochastic differential systems*, 1979, pp. 85–117.

[196] R. J. Elliott and D. B. Madan, "A discrete time equivalent martingale measure", *Mathematical finance*, vol. 8, no. 2, pp. 127–152, 1998.

[197] Z. Feinstein and B. Rudloff, "A supermartingale relation for multivariate risk measures", *Quantitative Finance*, vol. 18, no. 12, pp. 1971–1990, 2018.

[198] D. Williams, *Probability with Martingales*. Cambridge University Press, 1991.

[199] F. Chung and L. Lu, "Concentration inequalities and martingale inequalities – a survey", *Internet Mathematics*, vol. 3, no. 1, pp. 79–127, 2006.

[200] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance", *IEEE Transactions on Control Systems Technology*, 2020.

[201] A. Quimby, G. Maycock, C. Palmer, and S. Buttress, "The factors that influence a driver's choice of speed: a questionnaire study", UK:Transport Research Laboratory, Tech. Rep., 1999.

[202] R. Sharma, M. Gupta, and G. Kapoor, "Some better bound on variance with applications", *Journal of Mathematical Inequalities*, vol. 4, no. 3, pp. 355–363, 2010.

[203] Y. Nie, O. Faqir, and E. C. Kerrigan, "Iclocs2: Solve your optimal control problems with less pain", in *Proceedings of 6th IFAC Conference on Nonlinear Model Predictive Control*, 2018, pp. 2405–8963.

[204] H. Ishii and B. A. Francis, *Limited Data Rate in Control Systems with Networks*. Springer, 2002.

[205] T. Li, M. Fu, L. Xie, and J.-F. Zhang, "Distributed consensus with limited communication data rate", *IEEE Transactions on Automatic Control*, vol. 56, no. 2, pp. 279–292, 2010.

[206] R. E. Allen, A. A. Clark, J. A. Starek, and M. Pavone, "A machine learning approach for real-time reachability analysis", in *Proceedings of 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2202–2208.

[207] M. Chen, "High dimensional reachability analysis: Addressing the curse of dimensionality in formal verification", PhD thesis, University of California, Berkeley, 2017.