Technische Universität München
TUM School of Computation, Information and Technology

Formal Methods for Sandboxing Controllers in Cyber-Physical Systems

Bingzhuo Zhong

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 26.04.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 03.07.2023 angenommen.

To my family, to my beloved.

Acknowledgments

Pursuing a Ph.D. is a long journey, and getting the degree is a remarkable milestone. Before opening a new chapter of my life, I would like to express my gratitude to those who accompanied me along this journey.

First and foremost, I am truly lucky to have Prof. Marco Caccamo and Prof. Majid Zamani as my advisors. During the whole journey of the pursuit of my Ph.D., Marco gave me a lot of guidance and inspiration for my research and provided me with substantial resources to realize my research goals. In particular, without his trust and support, I would not have had the opportunity to lead the construction of the drone lab in the chair, with which I greatly developed my skills for managing scientific projects that will absolutely play a crucial role in my future academic career. At the same time, despite the far distances between Munich and Boulder, Majid is always available to give me constructive and insightful guidance for my research. His enthusiasm, creativity, and perfectionism for research greatly impact my professional demeanor as a researcher. I would like to express my deepest gratitude to both of them. They are not only my advisors but also my friends. None of the achievements during my pursuit of the Ph.D., including this dissertation, would be possible without their help. Their wisdom, enthusiasm, and creativity will inspire me to be a better researcher in my future life. Besides, my deep thanks also go to Prof. Murat Arcak for hosting my visit to his group at UC Berkeley and for those insightful discussions we had during my visit, which broadened my horizon on the research. The research visit in UC Berkeley was a special and memorable experience, and I am sincerely honored to have him on my thesis committee.

Moreover, I would like to thank my research collaborators, including Mr. Hongpeng Cao, Dr. Siyuan Liu, Prof. Abolfazl Lavaei, Ms. Ameneh Nejati, Ms. Niloofar Jahanshahi, Dr. Paul Griffioen, Dr. Julien Provost, and Mr. Claudius V. Jordan. This dissertation benefited a lot from those fruitful cooperations with them. I would also like to thank all my colleagues at the Chair of Cyber-Physical Systems in Production Engineering at the Technical University of Munich and the HyConSys Lab at the University of Colorado Boulder. I am fortunate to have had the privilege to work with them during the past five years. My thanks also go to the colleagues I met during my visit in UC Berkeley for the welcoming and inclusive ambiance and for those worthwhile discussions, in particular for those during the reading group every Friday.

Last but not least, no words can express my gratitude to my parents and my girlfriend Li. Due to the pandemic, I am not able to see them in person most of the time during my Ph.D. in the past four years. This is indeed a hard time. However, the geographical barriers and time differences between Europe and China do not burden our connections.

Whenever I have any difficulties, I know they are always there for me. Without their endless support, I would not have overcome those hard times.

Bingzhuo Zhong
Munich, April 2023

Abstract

Cyber-Physical Systems (CPS) are complex, heterogeneous systems, which combine cyber (computation and communication) and physical components that interact tightly with each other in a feedback loop. In the past decade, many high-performance but unverified controllers, such as artificial-intelligence-based (a.k.a. AI-based) controllers, are desired to be employed in these systems to accomplish increasingly complex control missions, such as motion planning and autonomous driving. Nevertheless, the application of unverified controllers makes it increasingly challenging to ensure the overall safety of CPS. Meanwhile, ensuring safety of CPS is of vital importance due to their safety-critical nature in the sense that any malfunctions in these systems may lead to catastrophic consequences and even loss of life. To cope with this challenge, this thesis proposes a correct-by-construction control architecture called *Safe-visor architecture* to provide system-level safety guarantees over CPS by sandboxing unverified controllers in the control loop. The proposed architecture contains a *supervisor* that checks inputs from the unverified controller and makes a compromise between the functionality and safety of the system. Meanwhile, a *safety advisor* runs in parallel to provide fall-back control inputs to ensure safety in case the unverified controller is rejected by the supervisor for safety reasons.

As the main contributions, multiple novel approaches for constructing the Safe-visor architecture using formal methods are proposed in this thesis, including history-based and state-based approaches. In the context of history-based approaches, given complex logical safety specifications modeled by deterministic finite automata (DFA), and a history state-run of the system at runtime, the proposed history-based supervisors estimate the risk of violating the desired safety specifications presuming that the unverified controller is accepted. To provide formal probabilistic guarantees of satisfying these specifications, a new abstraction-based construction scheme for synthesizing controllers over non-cooperative stochastic games is introduced in this thesis to design the safety advisors associated with the history-based supervisors.

As for state-based approaches, the supervisor decides whether or not to accept the unverified controller by leveraging the *current state* and a controlled invariant set over the state set of the system. A key insight is that the controlled invariant set refers to a set of safe states from which there exist controllers that maintain the system within this set. Accordingly, the supervisor will reject those inputs provided by the unverified controllers which would drive the system away from this set. Meanwhile, the controller associated with this set is used as the safety advisor. By leveraging abstraction-free approaches, this thesis considers two new notions of invariant sets over uncertain linear systems. In case the system model is known, this thesis proposes new set-based approaches to compute so-called *hybrid control invariant sets* to construct the Safe-visor architecture enforcing ω-regular properties modeled by deterministic Streett automata. If the system model is unknown, the thesis introduces a direct-data-driven approach to synthesize so-called *γ-robust safety invariant sets* for constructing the Safe-visor architecture against invariant properties. To showcase the proposed methodologies for designing the Safe-visor architecture, they are applied to several case studies in simulation, including DC motor, 3DOF-helicopter, inverted pendulum, etc., as well as a case study of controlling a quadrotor helicopter on a physical testbed.

Zusammenfassung

Cyber-Physische Systeme (CPS) sind komplexe, heterogene Systeme, die cyber (Berechnung und Kommunikation) und physische Komponenten kombinieren, die eng miteinander in einer Rückkopplungsschleife interagieren. In den letzten zehn Jahren werden vermehrt leistungsstarke, aber nicht verifizierte Controller wie künstliche Intelligenz (KI)-basierte Controller in diesen Systemen eingesetzt, um zunehmend komplexe Steueraufgaben wie Bewegungsplanung und autonomes Fahren zu erfüllen. Die Verwendung nicht verifizierter Controller stellt jedoch eine wachsende Herausforderung dar, um die Gesamtsicherheit von CPS zu gewährleisten. Gleichzeitig ist die Sicherstellung der Sicherheit von CPS von entscheidender Bedeutung, da sie sicherheitskritisch sind und Fehlfunktionen in diesen Systemen zu katastrophalen Folgen und sogar zum Verlust von Menschenleben führen können. Um dieser Herausforderung gerecht zu werden, schlägt diese Arbeit eine korrekte-durch-Konstruktion Steuerungsarchitektur namens "Safe-visor-Architektur" vor, um auf Systemebene Sicherheitsgarantien für CPS zu bieten, indem nicht verifizierte Controller in der Steuerungsschleife isoliert werden. Die vorgeschlagene Architektur enthält einen "Supervisor", der die Eingaben vom nicht verifizierten Controller überprüft und einen Kompromiss zwischen Funktionalität und Sicherheit des Systems findet. Gleichzeitig läuft ein "Safety Advisor" parallel, um alternative Steuereingaben bereitzustellen, um die Sicherheit zu gewährleisten, falls der nicht verifizierte Controller aus Sicherheitsgründen vom Supervisor abgelehnt wird.

Als Hauptbeiträge werden in dieser Arbeit mehrere neuartige Ansätze zur Konstruktion der Safe-visor-Architektur unter Verwendung formaler Methoden vorgeschlagen, einschließlich geschichts- und zustandsbasierter Ansätze. Im Kontext der geschichtsbasierten Ansätze schätzen die vorgeschlagenen geschichtsbasierten Supervisoren das Risiko der Verletzung der gewünschten Sicherheitsspezifikationen unter der Annahme, dass der nicht verifizierte Controller akzeptiert wird, basierend auf komplexen logischen Sicherheitsspezifikationen, die durch deterministische endliche Automaten modelliert sind und einem Laufzustandshistorie des Systems zur Laufzeit. Um formale probabilistische Garantien für die Erfüllung dieser Spezifikationen bereitzustellen, wird in dieser Arbeit ein neues abstraktionsbasiertes Konstruktionsverfahren zur Synthese von Controllern über nicht-kooperative stochastische Spiele eingeführt, um die Safety Advisors, die mit den geschichtsbasierten Supervisoren verbunden sind, zu entwerfen.

Im Hinblick auf zustandsbasierte Methoden entscheidet der Supervisor, ob der nicht verifizierte Controller akzeptiert wird oder nicht, indem er den aktuellen Zustand und eine kontrollierte invariante Menge über der Zustandsmenge des Systems nutzt. Ein wichtiger Erkenntnis ist, dass die kontrollierte invariante Menge auf eine Menge sicherer Zustände verweist, für die es Controller gibt, die das System innerhalb dieser Menge halten. Entsprechend wird der Supervisor jene Eingaben der nicht verifizierten Controller ablehnen, die das System von dieser Menge entfernen würden. Gleichzeitig wird der Controller, der mit dieser Menge assoziiert ist, als Safety Advisor verwendet. Durch den Einsatz von abstraktionsfreien Methoden betrachtet diese Arbeit zwei neue Konzepte von invarianten Mengen über unsichere lineare Systeme. Falls das Systemmodell bekannt ist, schlägt die Arbeit neue satzbasierte Ansätze vor, um sogenannte hybride Steuerungs-Invariantenmengen zu berechnen und die Safe-visor-Architektur zur Durchsetzung von ω-regulären Eigenschaften, die durch deterministische Streett-Automaten modelliert werden, zu konstruieren. Falls das Systemmodell unbekannt ist, stellt die Arbeit einen direkt datengetriebenen Ansatz vor, um sogenannte *γ-robuste Sicherheitsinvariante Mengen* zur Konstruktion der Safe-visor-Architektur gegen invariante Eigenschaften zu synthetisieren. Um die vorgeschlagenen Methoden zur Gestaltung der Safe-visor-Architektur zu demonstrieren, werden sie auf mehrere Fallstudien in Simulation angewendet, darunter Gleichstrommotor, 3DOF-Hubschrauber, umgekehrtes Pendel usw., sowie eine Fallstudie zur Steuerung eines Quadrotor-Hubschraubers auf einem physischen Prüfstand.

Publications by the Author during Ph.D.

Under Submission and Review

1. **B. Zhong**, S. Liu, M. Caccamo, and M. Zamani, ”Secure-by-Construction Synthesis for Control Systems”, In: *IEEE Transactions on Automatic Control*, under review, 2023.
2. N. Jahanshahi, **B. Zhong**, and M. Zamani, ”Sandboxing (AI-based) unverified controllers in partially-observable systems: A data-driven approach”, In: *IEEE Transactions on Automatic Control*, under review, 2023.
3. **B. Zhong**, P. Griffioen, M. Arcak, M. Caccamo, and M. Zamani, ”Data-Driven Controlled Invariant Sets for Gaussian Process State Space Models”, under submission, 2023.

Journal Papers

4. **B. Zhong**, M. Zamani, and M. Caccamo, ”Formal synthesis of controllers for uncertain linear systems against ω-regular properties: A set-based approach”, In: *IEEE Transactions on Automatic Control*, 2023.
5. **B. Zhong**, A. Lavaei, M. Zamani, and M. Caccamo, ”Automata-based controller synthesis for stochastic systems: A game framework via approximate probabilistic relations”, In: *Automatica*, Vol. 147C, 2023.
6. **B. Zhong**, A. Lavaei, H. Cao, M. Zamani, and M. Caccamo, ”Safe-visor architecture for sandboxing (AI-based) unverified controllers in stochastic Cyber-Physical Systems”, In: *Nonlinear Analysis: Hybrid Systems*, Vol. 43, 2021.
7. **B. Zhong**, C. Jordan, and P. Julien, ”Extending signal temporal logic with quantitative semantics by intervals for robust monitoring of Cyber-Physical Systems”, In: *ACM Transactions on Cyber-Physical Systems*, Vol. 5, No. 2, 2021.

Conference Papers

8. **B. Zhong**, S. Liu, M. Caccamo, and M. Zamani, ”Towards trustworthy AI: Sandboxing AI-based unverified controllers for safe and secure Cyber-Physical Systems”, In: *Proceedings of 62nd IEEE Conference on Decision and Control (CDC)*, to appear, 2023.
9. **B. Zhong**, S. Liu, M. Caccamo, and M. Zamani, ”Secure-by-construction controller synthesis via control barrier functions”, In: Proceedings of The 22nd IFAC World Congress, to appear, 2023.

10. **B. Zhong**, H. Cao, M. Zamani, and M. Caccamo, ”Towards safe AI: Sandboxing DNNs-based controllers in stochastic games”, In: Proceedings of the 37th AAAI Conference on Artificial Intelligence, Vol. 37, No. 12, pp. 15340-15349, 2023.
11. **B. Zhong**, M. Zamani, and M. Caccamo, ”Synthesizing safety controllers for uncertain linear systems: A direct data-driven approach”, In: Proceedings of IEEE Conference on Control Technology and Applications (CCTA), pp. 1278-1284, 2022.
12. **B. Zhong**, A. Lavaei, M. Zamani, and M. Caccamo, ”Poster Abstract: Controller synthesis for nonlinear stochastic games via approximate probabilistic relations”, In: Proceedings of 25th ACM International Conference on Hybrid Systems: Computation and Control (HSCC), ACM, 2022.
13. A. Nejati*, **B. Zhong***, M. Caccamo, and M. Zamani, ”Controller synthesis for unknown polynomial-type systems: A data-driven approach”, In: Proceedings of the International Workshop on Computation-Aware Algorithmic Design for Cyber-Physical Systems (CAADCPS), IEEE, 2022.
14. A. Nejati*, **B. Zhong***, M. Caccamo, and M. Zamani, ”Data-driven controller synthesis of unknown nonlinear polynomial systems via control barrier certificates”, In: Proceedings of the 4th Annual Learning for Dynamics and Control Conference (L4DC), PMLR 168, 2022.
15. **B. Zhong**, M. Zamani, and M. Caccamo, ”Set-based approach for synthesizing controllers enforcing ω-regular properties over uncertain linear control systems”, In: Proceedings of American Control Conference (ACC), pp. 1575-1581, 2022.
16. A. Lavaei, **B. Zhong**, M. Caccamo, and M. Zamani, ”Towards trustworthy AI: Safe-visor architecture for uncertified controllers in stochastic Cyber-Physical Systems”, In: Proceedings of the International Workshop on Computation-Aware Algorithmic Design for Cyber-Physical Systems (CAADCPS), ACM, 2021.
17. **B. Zhong**, M. Zamani, and M. Caccamo, ”Sandboxing controllers for stochastic Cyber-Physical Systems”, In: 17th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS), Lecture Notes in Computer Science (vol 11750), Springer, 2019

*Both authors have contributed equally.

Contents

List of Figures

List of Tables

List of Abbreviations

CPS	Cyber-Physical Systems
AI	artificial intelligence
dtLCS	discrete-time linear control system
gMDP	general Markov decision process
gDTSGs	general discrete-time stochastic games
i.i.d.	independent and identically distributed
DFA	deterministic finite automata
DSA	deterministic Streett automata
LTL	linear temporal logic
SDP	semi-definite programming
CBF	control barrier functions
HCI set	hybrid controlled invariant set
iff	if and only if
LQR	linear–quadratic regulator
γ-RSI set	γ-robust safety invariant set
LMI	linear matrix inequality
DNNs	deep-neural-networks
GCS	ground control station
NP-complete	nondeterministic polynomial-time complete
DDPG	deep deterministic policy gradient

1 Introduction

1.1 Motivation

Cyber-Physical Systems (CPS) are intricate systems that combine physical and cyber components interacting with each other in a feedback loop. In the past decades, CPS have been widely deployed in our daily life; typical applications include autonomous cars, smart power networks, and unmanned aerial vehicles. Nowadays, with the control tasks for CPS becoming increasingly complex, there is a growing need for using various kinds of complex, (potentially) high-performance controllers in these systems. In particular, inspired by those remarkable achievements of artificial-intelligence (AI) in various domains, such as natural language processing and image recognition, there is surging ambition of applying those state-of-the-art AI-based techniques to design controllers for CPS (see e.g., [32, 91, 101, 133, 182, 66, 172]). Nevertheless, the complex nature of these controllers make it very challenging to ensure the overall safety of CPS and may give rise to various reliability problem [151, 146, 111]. Meanwhile, CPS are typically safety-critical in the sense that any design faults or malfunctions in the systems can lead to catastrophic consequences and even loss of life.

To ensure the safety of CPS using complex control software, large-scale simulation and exhaustive testing are the main strategies. However, these strategies suffer from many practical problems. Specifically, modern Cyber-Physical Systems contain various uncertainties, leading to an enormous set of test cases to be covered, while testing all scenarios in this set is expensive. For instance, it is estimated that hundreds of millions of miles of test-driving are necessary to show that an autonomous driving software may behave not worse than human beings [92]. Even if such a test can be performed, it may still be incapable of ensuring safety as it is inevitable that certain scenarios in reality may remain untested, particularly for those scenarios that are outside so-called *the operation design domain* [106]. Additionally, plenty of complex control software are iterated in rapid development processes in the sense that new versions of these software may be released on a monthly or even weekly basis [134]. The same testing needs to be performed on each new version of the control software, which is not achievable in reality (for instance, the testing scenario mentioned above requires more than 12 years with 100 test vehicles driving 24 hours a day).

Instead of conventional trial-and-error schemes through testing, rigorous methodologies combining formal methods and control theory could be promising solutions. Specifically, the concept of verification and formal synthesis [13] could be applied to generate provably correct control software and provide formal safety guarantees. However, verification of complex controllers over CPS are shown to be very challenging as the complexities of verifying them grow quickly with the sizes of the problems of

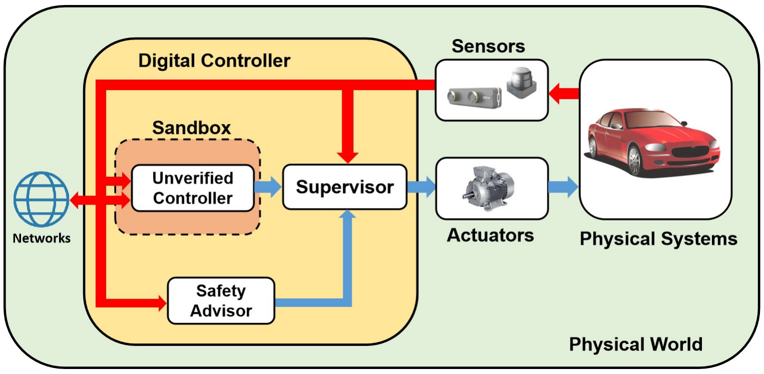


Figure 1.1: Safety advisor-supervisor (Safe-visor architecture) for sandboxing unverified controllers.

interest [61]. Notably, verifying those AI-based controllers, in which deep neural networks (DNNs) are deployed, is very challenging and is shown to be nondeterministic polynomial-time complete (NP-complete) in general [57]. In fact, theoretical results for verifying DNNs lag far behind the current state-of-the-art regarding those DNNs applied in practice. More precisely, while existing results can handle the verification of a single DNNs with only a few layers and a few hundred neurons per layer [84, 99, 186], many complex DNNs applied nowadays have billions of parameters and complicated architectures [39, 134].

To summarize, despite the growing demands in applying various types of complex controllers, in particular those AI-based ones, in modern CPS, promising approaches for applying them in safety-critical scenarios are still missing. To pave the way for employing these controllers in safety-critical CPS, this thesis focuses on how to provide formal safety guarantees for CPS without testing or formally verifying those complex controllers deployed in the control loop. Hereafter, those controllers that are difficult to be verified or tested are referred to as *unverified controllers* throughout this thesis.

1.2 Safe-visor Architecture

In general, the idea of *sandbox* [160] is leveraged to tackle the main problem of this thesis, which is recalled as follows.

Sandbox *The sandbox is a common mechanism that ensures the security of CPSs containing untested and untrusted software components. A verified sandbox is designed to isolate these components from the critical part of a host machine in a sandbox control environment, including its operating system. Consequently, an untrusted component can only actuate the plant through a tightly-monitored interface so that the plant would not be endangered by accidental or malicious harm caused by the untrusted component.*

Leveraging this idea, I propose a correct-by-construction architecture, namely *Safe-visor architecture*, to sandbox unverified controllers at runtime and provide a system-level [47, 203] safety guarantee over the physical systems. The proposed architecture employs a safety advisor and a supervisor, as demonstrated in Figure 1.1. Concretely, to provide safety guarantees for the physical system, verifiable safety rules should be specified for the unverified controller in the Safe-visor architecture. At runtime, the safety advisor is responsible for providing advisory inputs to ensure the overall safety of the physical system. Meanwhile, the supervisor checks the input given by the unverified controller at every time step according to the safety rules. The unverified inputs would only be accepted when they follow these rules; otherwise, the supervisor would accept the advisory input from the safety advisor. Note that the unverified controller is designed to realize some tasks which are much more complex than just keeping the system safe, while safety advisor *only* focuses on keeping the system safe. Therefore, the safety advisor should only be used if the unverified controller tries to perform some harmful actions. By deploying the proposed architecture, one can exploit the functionalities offered by the unverified controller while preventing the system from being unsafe, even though there is no safety guarantee over the unverified controller.

Considering those unverified controllers developed based on deep-neural-networks, it is worth mentioning those existing results (e.g. [82, 100, 45, 51, 204, 82, 112, 65, 193, 194, 145, 79, 123]) in which formal guarantees are achieved by appropriately incorporating the desired objectives in the reward functions when training DNNs-based controllers. Compared with these results, the desired safety properties are decoupled from the construction of the (DNNs-based) unverified controller in the proposed architecture. Therefore, the proposed architecture can provide formal guarantees for any type of unverified controllers regardless of their design.

1.3 Contributions and Outline of the Dissertation

This dissertation provides various methodologies to construct the Safe-visor architecture, as introduced in Section 1.2, over different types of control systems against various complex temporal logical safety properties. In general, these methodologies can be categorized into *history-based approaches* and *state-based approaches.*

In the context of the history-based approaches, the supervisor decides whether or not to accept the unverified controllers based on a *history state-run* of a system at runtime. Concretely, based on a state-run obtained at runtime, the supervisor first estimates the risk of the system violating the desired safety properties presuming that the unverified controller is accepted. If the violation risk is lower than the maximal tolerable probability of violating the desired properties, then the unverified controller can be accepted. Leveraging this idea, **Chapter 3 and 4** introduce abstraction-based construction schemes of the Safe-visor architecture over stochastic systems modeled by general Markov decision processes (gMDP) and general discrete-time stochastic games (gDTSG). Concretely, **Chapter 3** introduces how to synthesize controllers for stochastic games against those safety properties modeled by deterministic finite automata.

Such controllers are used as the safety advisor in the Safe-visor architecture, based on which the design of history-based supervisors will be introduced in **Chapter 4**.

As for the state-based approaches, instead of looking into the history state-run of a system, the supervisor makes decisions based on the *current state* of the system as well as a controlled invariant set over the system's state set. More specifically, a controlled invariant set here refers to a set of safe states, from which there exist controllers keeping the system staying in this set regardless of a given set of disturbances. Accordingly, whenever the unverified controller would not drive the system leaving this set, it can be accepted by the supervisor; otherwise, the safety advisor would be used to keep the system safe. Following this idea, **Chapter 5** provides computational approaches for synthesizing *hybrid controlled invariant sets*, with which one can design the Safe-visor architecture enforcing ω-regular properties. In **Chapter 6**, I focus on linear systems with unknown dynamics affected by unknown-but-bounded disturbances. To design the Safe-visor architecture with respect to invariance properties, a direct data-driven approach is proposed for computing so-called *γ-robust safety invariant sets* over these systems. Note that different from Chapter 3 and 4, the results in Chapter 5 and 6 do not require constructing finite abstractions over the original systems. Therefore, these approaches are also referred to as *abstraction-free approaches*.

Additionally, **Chapter 2** provides some mathematical notations and preliminaries from mathematics, control theory, and formal methods, which will be used throughout this thesis. **Chapter 7** concludes the results in this thesis and discusses potential future directions on top of these results. The technical contributions of Chapter 3-6 are summarized as follows.

1) **Abstraction-based approaches for synthesizing controllers over non-cooperative stochastic games (Chapter 3)**

   In order to provide an abstraction-based computation framework for constructing Safe-visor architecture that is robust to noises and disturbances, the main focus of Chapter 3 is to provide abstraction-based methodologies to construct controllers enforcing complex logical safety properties over non-cooperative stochastic games. These controllers will be used as the safety advisor in Chapter 4. To this end, (ϵ, δ)-approximate probabilistic relation is deployed to quantify the probabilistic dependency between an original model and its finite abstraction, which plays a crucial role in providing formal safety guarantees and considering model-order reduction when constructing the controllers. Here, a systematic approach is proposed to establish such a relation (if it exists) over a class of nonlinear stochastic games with slope restrictions on the nonlinearity, while existing results in the literature focus on linear systems only and without providing a systematic approach for building this relation. Moreover, focusing on complex logical properties modeled by deterministic finite automata (DFA), new Bellman operators are proposed for synthesizing controllers concerning two types of problems, namely the problem of *robust satisfaction* and *worst-case violation.* Compared with existing results for abstraction-based controller synthesis over gMDP and gDTSGs, the new Bellman operators proposed in this chapter provide less conservative probabilistic guarantees for satisfying the desired safety properties.

The materials in this chapter were published in [214].

2) **Abstraction-based construction of Safe-visor architecture (Chapter 4)**

   On top of the abstraction-based controller synthesis approaches proposed in Chapter 3, Chapter 4 presents the construction of history-based supervisors in the Safe-visor architecture following the idea of the history-based approaches as introduced above. Concretely, the history-based supervisor deploys an augmented memory state containing the state (and input) of the original system, the finite abstraction, and the DFA modeling the desired safety properties. Having a sequence of augmented memory states at runtime, efficient algorithms concerning the problem of robust satisfaction and worst-case violation are proposed for estimating the risk of violating the desired safety properties in case of accepting the unverified controller. Additionally, rigorous results are provided for maintaining the approximate probabilistic relation between the original system and the finite abstraction if the unverified controllers are accepted. The proposed abstraction-based construction scheme for Safe-visor architecture is validated through simulation over several case studies and, for the first time, through experiments on a physical test-bed of quadrotor helicopters.

   The materials presented in this chapter were published in [213, 212].

3) **Set-based (Abstraction-free) methodologies for synthesizing controllers against ω-regular properties (Chapter 5)**

   In this chapter, new discretization-free approaches are proposed for synthesizing controllers over uncertain linear systems enforcing ω-regular properties. Inspired by standard set-based approaches [165] for synthesizing controllers over continuous sets against invariance properties, new set-based approaches are proposed to compute so-called *hybrid controlled invariant (HCI) set* by leveraging new iterative schemes. Following the basic idea of state-based approaches, this set is used to construct a controller deployed as the safety advisor, and used by the supervisor to check the validity of those control inputs provided by the unverified controller. More precisely, considering a deterministic Streett automata (DSA), a (systematic) approach is first provided to properly exploit the structure of the DSA in the computation of the HCI sets. Then, a new set-based iterative scheme is proposed for computing the HCI sets, and it is shown that the proposed iterative scheme converges to the maximal HSI set. To guarantee the termination of the iterative scheme for computing HCI sets within a finite number of iterations, two alternative iterative schemes are provided. Moreover, the relation between the worst-case space and time complexities of these methodologies and the structure of the automata representing the desired ω-regular properties is investigated. Finally, comparisons among the proposed approaches and existing tools for synthesizing controllers enforcing ω-regular properties are conducted.

The results of this chapter were published in the publications [215, 217].

4) **Direct data-driven approaches for synthesizing controllers against invariance properties (Chapter 6)**

   This chapter introduces a direct data-driven approach to compute a state feedback controller enforcing invariance properties over linear systems with unknown dynamics. To this end, the notion of γ-robust safety invariant (γ-RSI) sets is first proposed. Note that the existence of γ-RSI sets indicates that there exists a controller keeping the (unknown) system staying in the γ-RSI sets regardless of exogenous disturbances bounded by γ. Hence, the γ-RSI set can be leveraged to construct the Safe-visor architecture following the basic idea of state-based approaches. Here, the γ-RSI sets are computed via solving a semi-definite programming (SDP) problem, which can be formulated leveraging a *single trajectory* collected from the underlying system without an intermediate system identification phase. Moreover, the relation between the proposed data-driven approach and the condition of persistency of excitation [200] is also investigated.

   The covered materials in this chapter were published in [216].

2 Preliminaries

By considering the classical results in the fields of mathematics, control theory, and formal methods, some preliminaries and notations are introduced in this chapter, which will be used throughout this thesis.

2.1 Notations

$\mathbb{R}$ and $\mathbb{N}$ are used to denote sets of real and natural numbers, respectively. These symbols are annotated by subscripts to restrict the sets in a usual way, *e.g.,* $\mathbb{R}_{\geq 0}$ denotes the set of non-negative real numbers. Moreover, $\mathbb{R}^{n\times m}$ with $n, m \in \mathbb{N}_{\geq 1}$ denotes the vector space of real matrices with n rows and m columns. For $a, b \in \mathbb{R}$ (resp. $a, b \in \mathbb{N}$) with $a \leq b$, the close, open, and half-open intervals in $\mathbb{R}$ (resp. $\mathbb{N}$) are denoted by $[a, b]$, (a, b) ,$[a, b)$, and $(a, b]$, respectively. $\mathbf{0}_n$ and $\mathbf{1}_n$ denote, respectively, the column vector in $\mathbb{R}^n$ with all elements equal to 0 and 1. Moreover, $\mathbf{0}_{n\times m}$ and $\mathbf{I}_n$ denote the zero matrix in $\mathbb{R}^{n\times m}$, and the identity matrix in $\mathbb{R}^{n\times n}$, respectively. Their indices are omitted if the dimension is clear from the context. Given a matrix $A \in \mathbb{R}^{n\times n}$, $im(A)$ denotes the image of A, and $\|A\|$ represents the operator norm of A, which is equal to the largest singular value of A. Additionally, $\text{rank}(A)$, $\det(A)$, $A^\top$, $A(i)$, and $A(i, j)$ denote the rank, the determinant, the transpose, the i-th column, and the entry in i-th row and j-th column of A, respectively.

Given a vector $x \in \mathbb{R}^n$, $\|x\|$ and $|x|$ denote the Euclidean norm and infinity norm of x, respectively. Additionally, $\mathbb{B}^n$ denotes the closed unit ball centered at the origin in $\mathbb{R}^n$ with respect to the infinity norm. Given N vectors $x_i \in \mathbb{R}^{n_i}$, $n_i \in \mathbb{N}_{\geq 1}$, and $i \in \{1, \ldots, N\}$, $x = [x_1; \ldots; x_N]$ represents the corresponding column vector of the dimension $\sum_i n_i$. Moreover, given a set X, $X^{\mathbb{N}}$ and X^ω denote the Cartesian product of the countably and uncountable infinite number of X, respectively. Given sets X_i, $i \in [1, N]$, and their Cartesian product $\mathsf{M} = X_1 \times \ldots \times X_N$, the projection of X onto X_i is denoted by mapping $\pi_{X_i} : X \to X_i$, and give a vector $\mathsf{m} = (x_1, x_2, \ldots, x_N) \in \mathsf{M}$ with $x_i \in X_i$, one has $\mathsf{m}_{X_i} := x_i$. Given sets X and Y, a relation $\mathscr{R} \in X \times Y$ is a subset of the Cartesian product $X \times Y$ that relates $x \in X$ with $y \in Y$ if $(x, y) \in \mathscr{R}$, which is equivalently denoted by $x\mathscr{R}y$.

Given sets A and B, $f : A \to B$ denotes an ordinary map from A to B. Accordingly, given functions $f : X \to Y$ and $g : Y \to Z$, $g \circ f : X \to Z$ denotes the composition of functions f and g. Given sets A and B, $B\backslash A = \{x | x \in B \text{ and } x \notin A\}$ denotes the complement of A with respect to B. The Minkowski sum of two sets A, $B \subseteq \mathbb{R}^n$ is denoted by $A + B = \{x \in \mathbb{R}^n | \exists a \in A, \exists b \in B, x = a + b\}$. In this thesis, $x + A$ instead of $\{x\} + A$ is used to denote the Minkowski sum of set A and $\{x\}$ where $x \in \mathbb{R}^n$ for the sake of simple presentation. Moreover, $A - B = \{a \in A | a + B \subseteq A\}$ denotes the Pontryagin set difference between A and B.

2.2 Mathematical Preliminaries

A topological space S is called a Borel space if it is homeomorphic to a Borel subset of a Polish space (*i.e.,* a separable and completely metrizable space). One of the examples of Borel space is the Euclidean spaces $\mathbb{R}^n$. Here, any Borel space S is assumed to be endowed with a Borel σ-algebra denoted by $\mathcal{B}(S)$. A map $f : X \to Y$ is measurable whenever it is Borel measurable. Moreover, a map $f : X \to Y$ is universally measurable if the inverse image of every Borel set under f is measurable w.r.t. every complete probability measure on X that measures all Borel subsets of X.

A probability space in this thesis is presented by $(\hat{\Omega}, \mathcal{F}_{\hat{\Omega}}, \mathbb{P}_{\hat{\Omega}})$, where $\hat{\Omega}$ is the sample space, $\mathcal{F}_{\hat{\Omega}}$ is a sigma-algebra on $\hat{\Omega}$ which comprises subsets of $\hat{\Omega}$ as events, and $\mathbb{P}_{\hat{\Omega}}$ is a probability measure that assigns probabilities to events. Throughout this chapter, random variables, denoted by X, are of particular interest, which take values from measurable spaces $(S,\mathcal{B}(S))$, *i.e.,* random variables here are measurable functions $X : (\hat{\Omega}, \mathcal{F}_{\hat{\Omega}}) \to (S, \mathcal{B}(S))$ such that one has $Prob\{\mathcal{Q}\} = \mathbb{P}_{\hat{\Omega}}\{X^{-1}(\mathcal{Q})\}$, $\forall \mathcal{Q} \in \mathcal{B}(S)$. For brevity, the probability measure on $(S,\mathcal{B}(S))$ is directly presented without explicitly mentioning the underlying probability space and the function X itself. Additionally, $\mathbf{P}(S, \mathcal{B}(S))$ denotes a set of probability measures on $(S,\mathcal{B}(S))$.

Throughout the thesis, $\mathcal{N}(\cdot|\mu, \Sigma)$ represents the *normal distribution* with mean μ and covariance matrix Σ, and $\delta_d(\cdot|c)$ indicates *Dirac delta distribution* centered at c.

2.3 Deterministic Systems

In some parts of the thesis, *discrete-time linear control systems (dtLCS)* are considered, which are defined as follows.

Definition 2.3.1. *(*dtLCS*) A discrete-time linear control system S is a tuple*

$$S = (X, X_0, U, W, f), \tag{2.3.1}$$

where $X \subseteq \mathbb{R}^n$ is the state set, $U \subset \mathbb{R}^m$ and $W \subset \mathbb{R}^n$ are compact sets of input and exogenous disturbances, respectively. Set $X_0 \subseteq X$ is the set of initial states. Function $f : X \times U \times W \to X$ characterizes the discrete-time dynamics as:

$$x(k+1) = f(x(k), u(k), w(k)) := Ax(k) + Bu(k) + w(k), \tag{2.3.2}$$

with $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$.

With these notations, the evolution of a system S as in (2.3.1) can be described by its paths, as defined below.

Definition 2.3.2. *(*Path*) A path of a dtLCS S as in* (2.3.2) *is defined as*

$$\xi := (x(0), u(0), \ldots, x(k-1), u(k-1), x(k), \ldots), k \in \mathbb{N},$$

where $x(k+1) = Ax(k) + Bu(k) + w(k)$ for some $w(k) \in W$.

Moreover, $\xi_x := (x(0), x(1), \ldots,\ x(k), \ldots)$ and $\xi_u := (u(0), u(1), \ldots,\ u(k), \ldots)$ represent the subsequences of states and inputs in ξ, respectively.

2.4 Stochastic Systems

2.4.1 General Markov Decision Processes

In this thesis, one of the stochastic systems of interest are those which can be formulated as a class of general Markov decision processes (gMDPs) that evolve over continuous or uncountable state sets. This class of models generalizes the usual notion of MDPs [13] by adding an output set over which properties of interest are defined.

Definition 2.4.1. (gMDP) *A general Markov decision process is a tuple*

$$\mathfrak{D} = (X, U, x_0, T, Y, h), \tag{2.4.1}$$

where,

- *$X \subseteq \mathbb{R}^n$ is a Borel set as the state set of the system. $(X, \mathcal{B}(X))$ denotes the measurable space with $\mathcal{B}(X)$ being the Borel σ-algebra on the state set;*
- *$U \subseteq \mathbb{R}^m$ is a Borel set as the input set of the system;*
- *$x_0 \in X$ is the initial state;*
- *$T : \mathcal{B}(X) \times X \times U \to [0, 1]$ is a conditional stochastic kernel that assigns to any $x \in X$ and $u \in U$ a probability measure $T(\cdot|x, u)$ on the measurable space $(X, \mathcal{B}(X))$. This stochastic kernel specifies probabilities over executions $\{x(k), k \in \mathbb{N}\}$ of the gMDP such that for any set $\mathcal{X} \in \mathcal{B}(X)$ and any $k \in \mathbb{N}$,*

$$\mathbb{P}\big\{x(k+1) \in \mathcal{X} \,|\, x(k), u(k)\big\} = \int_{\mathcal{X}} T(\mathsf{d}x(k+1)|x(k), u(k));$$

- *$Y \subseteq \mathbb{R}^q$ is a Borel set as the output set of the system;*
- *$h : X \to Y$ is a measurable function that maps a state $x \in X$ to its output $y = h(x)$.*

Alternatively, a gMDP $\mathfrak{D}$ as in (2.4.1) can be described by difference equations

$$\mathfrak{D}\colon \begin{cases} x(k+1) = f(x(k), u(k), \varsigma(k)), \\ y(k) = h(x(k)), \end{cases} \qquad k \in \mathbb{N}, \tag{2.4.2}$$

where $x(k) \in X$, $u(k) \in U$, $y(k) \in Y$, and $\varsigma := \{\varsigma(k) : \hat{\Omega} \rightarrow V_\varsigma, k \in \mathbb{N}\}$ is a sequence of independent and identically distributed (i.i.d.) random variables from the sample space $\hat{\Omega}$ to a set V_ς. Then, the evolution of a gMDP can be described by its paths and output sequences as defined below.

Definition 2.4.2. (Path and Output Sequence of gMDP) *A path of a gMDP* $\mathfrak{D} = (X, U, x_0, T, Y, h)$ *is*

$$\omega = (x(0), u(0), \ldots, x(k-1), u(k-1), x(k), \ldots), k \in \mathbb{N}$$

where $x(k) \in X$ *and* $u(k) \in U$*. Accordingly,* $\omega_x = (x(0), x(1), \ldots, x(k), \ldots)$ *and* $\omega_u = (u(0), u(1), \ldots, u(k), \ldots)$ *denote the subsequences of states and inputs in* ω*, respectively. The corresponding* output sequence *is denoted by*

$$y_\omega = (y(0), y(1), \ldots, y(k), \ldots), k \in \mathbb{N}$$

with $y(k) = h(x(k))$*. Moreover,* ω_k *denotes the path up to the time instant* k*;* ω_{xk}*,* ω_{uk}*, and* $y_{\omega k}$ *denote the state, input, and output subsequences corresponding to* ω_k*, respectively.*

The space of all infinite paths $\Omega = (X \times U)^{\mathbb{N}}$ along with their product σ-algebra $(\mathcal{B}(X) \times \mathcal{B}(U))^{\mathbb{N}}$ is called a *canonical sample space for the gMDP.*

2.4.2 General Discrete-Time Stochastic Games

In some parts of the thesis, stochastic systems modeled as general discrete-time stochastic games (gDTSGs) between two non-cooperative players are considered. Following standard conventions, control input and adversary input in gDTSGs are referred to as Player I and Player II, respectively. This class of games, as formalized in the next definition, evolves over continuous or uncountable state sets with an output set over which properties of interest are defined.

Definition 2.4.3. *A general discrete-time stochastic game (gDTSG) is a tuple*

$$\mathfrak{D} = (X, U, W, X_0, T, Y, h), \tag{2.4.3}$$

where,

- $X \subseteq \mathbb{R}^n$ *is a Borel set as the state set.* $(X, \mathcal{B}(X))$ *denotes the measurable space with* $\mathcal{B}(X)$ *being the Borel sigma-algebra on* X*;*
- $U \subseteq \mathbb{R}^m$ *is a compact Borel set as the input set of Player I;*
- $W \subseteq \mathbb{R}^p$ *is a compact Borel set as the input set of Player II;*
- $X_0 \subseteq X$ *is the set of initial states;*

- $T : \mathcal{B}(X) \times X \times U \times W \to [0,1]$ *is a conditional stochastic kernel that assigns to any* $x \in X$*,* $u \in U$*, and* $w \in W$ *a probability measure* $T(\cdot|x,u,w)$ *on the measurable space* $(X, \mathcal{B}(X))$*. This stochastic kernel specifies probabilities over executions* $\{x(k), k \in \mathbb{N}\}$ *of the gDTSG such that for any set* $\mathcal{Q} \subseteq \mathcal{B}(X)$ *and for any* $k \in \mathbb{N}$*,*
$$\mathbb{P}\{x(k+1) \in \mathcal{Q} \,|\, x(k), u(k), w(k)\} = \int_{\mathcal{Q}} T(\mathsf{d}x(k+1)|x(k), u(k), w(k));$$
- $Y \subseteq \mathbb{R}^q$ *is a Borel set as the output set;*
- $h : X \to Y$ *is a measurable function that maps a state* $x \in X$ *to its output* $y = h(x)$*.*

Alternatively, a gDTSG $\mathfrak{D}$ as in (2.4.3) can be described by the following difference equations
$$\mathfrak{D}\colon \begin{cases} x(k+1) = f(x(k), u(k), w(k), \varsigma(k)), \\ y(k) = h(x(k)), \qquad k \in \mathbb{N}, \end{cases} \tag{2.4.4}$$
where $x(k) \in X$, $u(k) \in U$, $w(k) \in W$, $y(k) \in Y$, and $\varsigma := \{\varsigma(k) : \hat{\Omega} \to V_\varsigma, k \in \mathbb{N}\}$ is a sequence of independent and identically distributed (i.i.d.) random variables from the sample space $\hat{\Omega}$ to a set V_ς. With this notion, the evolution of a gDTSG can be described by its paths and output sequences as defined below.

Definition 2.4.4. (Path and output sequence of gDTSGs) *A path of a gDTSG* $\mathfrak{D}$ *as in* (2.4.3) *is define as*
$$\omega = (x(0), u(0), w(0), \ldots, x(k-1), u(k-1), w(k-1), x(k), \ldots),$$
where $x(k) \in X$*,* $u(k) \in U$*, and* $w(k) \in W$ *with* $k \in \mathbb{N}$*. Moreover,* $\omega_x = (x(0), x(1), \ldots, x(k), \ldots)$*,* $\omega_u = (u(0), u(1), \ldots, u(k), \ldots)$*, and* $\omega_w = (w(0), w(1), \ldots, w(k), \ldots)$ *denote the subsequences of states, control inputs of Player I, and adversarial inputs of Player II, respectively, which are associated with* ω*. The corresponding* output sequence *is denoted by*
$$y_\omega = (y(0), y(1), \ldots, y(k), \ldots),$$
with $y(k) = h(x(k))$*. In addition,* ω_k *denotes the path up to time instant* k*, and* $y_{\omega k}$ *denotes the output sequence corresponding to* ω_k*.*

The space for all infinite paths $\Omega = (X \times U \times W)^{\mathbb{N}}$ along with its product σ-algebra $(\mathcal{B}(X) \times \mathcal{B}(U) \times \mathcal{B}(W))^{\mathbb{N}}$ is called a *canonical sample space for the gDTSG.*

2.4.3 Approximate Probabilistic Relations

The stochastic models defined above will be deployed in Chapter 3 and 4 for designing the Safe-visor architecture using abstraction-based approaches. The formal probabilistic guarantees provided by these approaches rely on an approximate probabilistic relation that captures the probabilistic dependency between the executions of two gDTSGs (resp. gMDPs). Note that such relation between two gDTSGs is an extension of the approximate probabilistic relation between two stochastic systems without rational adversarial input [76]. Here, the definition of a δ-lifted relation over general state spaces is first recalled from [76], which paves the way for defining the approximate probabilistic relation between gDTSGs afterward.

Definition 2.4.5. (δ-lifted Relation) *Let $X, \hat{X}$ be two sets with associated measurable spaces $(X, \mathcal{B}(X))$ and $(\hat{X}, \mathcal{B}(\hat{X}))$. Consider a relation $\mathscr{R} \subseteq X \times \hat{X}$ that is measurable, i.e., $\mathscr{R} \in \mathcal{B}(X \times \hat{X})$, probability distributions $\Phi \in \mathbf{P}(X, \mathcal{B}(X))$, and $\Theta \in \mathbf{P}(\hat{X}, \mathcal{B}(\hat{X}))$. One has $(\Phi, \Theta) \in \bar{\mathscr{R}}_\delta$, denoted by $\Phi \bar{\mathscr{R}}_\delta \Theta$, with $\bar{\mathscr{R}}_\delta \subseteq \mathbf{P}(X, \mathcal{B}(X)) \times \mathbf{P}(\hat{X}, \mathcal{B}(\hat{X}))$ being a* δ-lifted relation*, if there exists a probability measure $\mathscr{L}$, referred to as a* lifting*, with a probability space $(X \times \hat{X}, \mathcal{B}(X \times \hat{X}), \mathscr{L})$ such that*

- $\forall \mathcal{X} \in \mathcal{B}(X)$, $\mathscr{L}(\mathcal{X} \times \hat{X}) = \Phi(\mathcal{X})$,
- $\forall \hat{\mathcal{X}} \in \mathcal{B}(\hat{X})$, $\mathscr{L}(X \times \hat{\mathcal{X}}) = \Theta(\hat{\mathcal{X}})$,
- $\mathscr{L}(\mathscr{R}) \geq 1 - \delta$, *i.e., for the probability space $(X \times \hat{X}, \mathcal{B}(X \times \hat{X}), \mathscr{L})$, one has $x \mathscr{R} \hat{x}$ with a probability of at least $1 - \delta$.*

Next, (ϵ, δ)-approximate probabilistic relations between two gDTSGs are defined based on the δ-lifted relations between their probability measures.

Definition 2.4.6. ((ϵ, δ)-Approximate Probabilistic Relations) *Consider gDTSGs $\mathfrak{D} = (X, U, W, X_0, Y, h)$ and $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ with the same output set. The gDTSG $\widehat{\mathfrak{D}}$ is (ϵ, δ)-stochastically simulated by $\mathfrak{D}$, denoted by $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$, if there exist relations $\mathscr{R} \subseteq X \times \hat{X}$, $\mathscr{R}_w \subseteq W \times \hat{W}$ and a Borel measurable stochastic kernel $\mathscr{L}_T(\cdot \mid x, \hat{x}, \hat{u}, w, \hat{w})$ on $X \times \hat{X}$ such that*

1. *$\forall (x, \hat{x}) \in \mathscr{R}$, $\|y - \hat{y}\| \leq \epsilon$, with $y = h(x)$ and $\hat{y} = \hat{h}(\hat{x})$;*
2. *$\forall (x, \hat{x}) \in \mathscr{R}$, and $\forall \hat{u} \in \hat{U}$, $\exists u \in U$ such that $\forall w \in W$, $\exists \hat{w} \in \hat{W}$ with $(w, \hat{w}) \in \mathscr{R}_w$ such that one has $T(\cdot \mid x, u, w)\ \bar{\mathscr{R}}_\delta\ \hat{T}(\cdot|\hat{x}, \hat{u}, \hat{w})$ with lifting $\mathscr{L}_T(\cdot|x, \hat{x}, \hat{u}, w, \hat{w})$;*
3. *$\forall x_0 \in X_0$, $\exists \hat{x}_0 \in \hat{X}_0$ such that $x_0 \mathscr{R} \hat{x}_0$.*

The second condition of Definition 2.4.6 implies implicitly that there exists an *interface function* [69]

$$\nu : X \times \hat{X} \times \hat{U} \to U, \tag{2.4.5}$$

such that the state probability measures are in the δ-lifted relation after one transition, when the control input $\hat{u}$ for $\widehat{\mathfrak{D}}$ is refined to u for $\mathfrak{D}$ using this function. Intuitively, δ-lifted relation between the state probability measures of $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$ indicates that the Euclidean distance between their outputs, *i.e.*, y and $\hat{y}$, is not larger than ϵ, with a probability of at least $1-\delta$ at each time step. This distance-based δ-lifted relation is the key for providing the safety guarantee. With this relation, one can control y indirectly by controlling $\hat{y}$ while keeping y sufficiently closed to $\hat{y}$ with some probability. Once one has $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$, a product gDTSG between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$ can be constructed as in the following definition.

Definition 2.4.7. *Consider gDTSGs $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ and $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$, interface function $\nu(x, \hat{x}, \hat{u})$, and the corresponding lifted kernel $\mathscr{L}_T$. The product gDTSG of $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$ is a gDTSG and defined as*

$$\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}} := (X_{||}, U_{||}, W_{||}, X_{0||}, T_{||}, Y_{||}, h_{||}),$$

where $X_{||} := X \times \hat{X}$ is the state set; $U_{||} := \hat{U}$ is the input set of Player I; $W_{||} := W$ is the input set of Player II; $X_{0||}$ the initial state set, with $x_{0||} := (x_0, \hat{x}_0) \in X_{0||}$, $x_0 \in X_0$, $\hat{x}_0 \in \hat{X}_0$, and $(x_0, \hat{x}_0) \in \mathscr{R}$; $T_{||} := \mathscr{L}_T$ is the stochastic transition kernel; $Y_{||} := Y$ is the output set; and $h_{||}(x, \hat{x}) := h(x)$ is the output map.

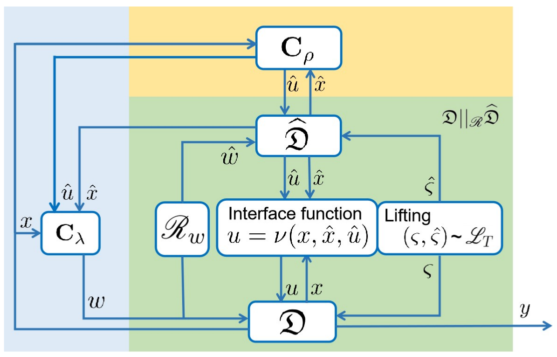


Figure 2.1: A coupling gDTSG $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ (green region) controlled by $\mathbf{C}_\rho$ (yellow region) and $\mathbf{C}_\lambda$ (blue region).

All ingredients of Definitions 2.4.6 and 2.4.7 are schematically depicted in Figure 2.1. Here, $\mathscr{L}_T$ characterizes the transition of states in $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ and specifies the relation of stochasticities between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$. Moreover, given an input w from $\mathbf{C}_\lambda$, $\hat{w}$ is selected such that $(w, \hat{w}) \in \mathscr{R}_w$ and fed to $\widehat{\mathfrak{D}}$. In practice, a function

$$\Pi_w : W \to \hat{W}, \tag{2.4.6}$$

is defined for matching each $w \in W$ to a $\hat{w} \in \hat{W}$. With Π_w, the stochastic kernel $\mathscr{L}_T$ as in Definition 2.4.6 can be written as $\mathscr{L}_T(\mathsf{d}x' \times d\hat{x}' \mid x, \hat{x}, \hat{u}, w)$. Moreover, according to [33, Corollary 3.1.2], one can decompose $\mathscr{L}_T$ as

$$\mathscr{L}_T(\mathsf{d}x'|x, \hat{x}, \hat{x}', \nu(x, \hat{x}, \hat{u}), w)\hat{T}(d\hat{x}'|\hat{x}, \hat{u}, \Pi_w(w)), \tag{2.4.7}$$

where $\mathscr{L}_T(\mathsf{d}x'|x, \hat{x}, \hat{x}', \nu(x, \hat{x}, \hat{u}), w)$ is a conditional stochastic kernel on x' given x, $\hat{x}$, $\hat{x}'$, $\hat{u}$, and w. Finally, the definition of an (ϵ, δ)-approximate probabilistic relation between two gMDPs is recalled fromn [74] for the sake of completeness.

Definition 2.4.8. *Consider two gMDPs $\mathfrak{D} = (X, U, x_0, T, Y, h)$ and $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{x}_0, \hat{T}, Y, \hat{h})$ with the same output set. System $\widehat{\mathfrak{D}}$ is (ϵ, δ)-stochastically simulated by $\mathfrak{D}$, denoted by $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$, if there exist a measurable relation $\mathscr{R} \subseteq X \times \hat{X}$ and a Borel measurable stochastic kernel $\mathscr{L}_T(\cdot \mid x, \hat{x}, \hat{u})$ on $X \times \hat{X}$ such that:*

- *$\forall (x,\hat{x}) \in \mathscr{R}$, $\|y-\hat{y}\| \leq \epsilon$, with $y = h(x)$ and $\hat{y} = \hat{h}(\hat{x})$;*
- *$\forall (x,\hat{x}) \in \mathscr{R}$ and $\forall \hat{u} \in \hat{U}$, there exists $u \in U$ such that $T(\cdot \mid x,u)\ \bar{\mathscr{R}}_\delta\ \hat{T}(\cdot \mid \hat{x},\hat{u})$ with lifting $\mathscr{L}_T(\cdot \mid x,\hat{x},\hat{u})$;*
- *$x_0 \mathscr{R} \hat{x}_0$.*

2.5 Automata for Modeling Complex Logical Properties

Throughout this thesis, two different automata are deployed to model the desired specifications. The first one is so-called *deterministic finite automata* (DFA) [13], as defined below.

Definition 2.5.1. *A* deterministic finite automata (DFA) *is a tuple $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, Π is a finite set of alphabet, $\tau : Q \times \Pi \to Q$ is a transition function, and $F \subseteq Q$ is a set of accepting states.*

Without loss of generality [13, Section 4.1], one can focus on those DFA which are *total*, i.e., given any $q \in Q$, $\forall \sigma' \in \Pi$, $\exists q' \in Q$ such that $q' = \tau(q,\sigma')$. A finite word $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_{k-1}) \in \Pi^k$ is accepted by $\mathcal{A}$ if there exists a finite state run $q = (q_0, q_1, \ldots, q_k) \in Q^{k+1}$ such that $q_{z+1} = \tau(q_z, \sigma_z)$, $\sigma_z \in \Pi$ for all $0 \leq z < k$ and $q_k \in F$. The set of words accepted by $\mathcal{A}$ is called the *language* of $\mathcal{A}$ and denoted by $\mathcal{L}(\mathcal{A})$.

Note that DFA is a powerful tool for capturing those properties that can be evaluated over finite traces [63]. In some part of the thesis, synthesizing controllers enforcing ω-regular properties [13] will also be of interest. These properties specify behaviours of a control system over *infinite* time horizons and can be modeled by *deterministic Streett automata* (DSA) [178], as defined below.

Definition 2.5.2. *A DSA is a tuple $\mathcal{A} = (Q, q_0, \Pi, \delta, Acc)$, where Q is a finite set of states, $q_0 \in Q$ is an initial state, Π is a finite set of alphabet, $\delta \subseteq Q \times \Pi \times Q$ is the set of all feasible transitions among Q, and $Acc = \{\langle E_1, F_1\rangle, \langle E_2, F_2\rangle, \ldots, \langle E_r, F_r\rangle, \ldots, \langle E_{\mathsf{r}}, F_{\mathsf{r}}\rangle\}$ denotes the accepting condition of the DSA where $\langle E_r, F_r\rangle$, $\forall r \in \{1, \ldots, \mathsf{r}\}$, are accepting state set pairs, with $E_r, F_r \subseteq Q$.*

Consider an infinite word denoted by $\sigma = (\sigma_0, \sigma_1, \ldots) \in \Pi^\omega$. An *infinite state run* $\mathbf{q} = (q_0, q_1, \ldots) \in Q^\omega$ on σ is an infinite sequence of states in which one has $(q_k, \sigma_k, q_{k+1}) \in \delta$, $\forall k \in \mathbb{N}$. Similarly, consider an finite word denoted by $\sigma_f = (\sigma_0, \ldots, \sigma_H) \in \Pi^{H+1}$, with $H \in \mathbb{N}$, $\mathbf{q} = (q_0, \ldots, q_H) \in Q^{H+1}$ denotes the corresponding *finite state run.* An infinite state run $\mathbf{q}$ is an *accepting run* of $\mathcal{A}$, if for all $\langle E_r, F_r\rangle \in$ Acc, $r \in \{1, \ldots, \mathsf{r}\}$, one has

$$inf(\mathbf{q}) \wedge E_r = \emptyset \text{ or } inf(\mathbf{q}) \wedge F_r \neq \emptyset, \tag{2.5.1}$$

where $inf(\mathbf{q})$ is the set of states in Q that are visited infinitely often in $\mathbf{q}$. Additionally, an infinite word σ corresponding to an accepting run $\mathbf{q}$ is said to be *accepted by $\mathcal{A}$*. The set of words accepted by $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is called the *language of $\mathcal{A}$*.

3 Abstraction-based Controller Synthesis for Non-cooperative Stochastic Games

3.1 Introduction

Formal synthesis of controllers for continuous-space stochastic systems have gained significant attention in the past two decades due to the increasing demand for synthesizing correct-by-construction controllers in real-life safety-critical applications, including self-driving cars, power grids, etc., to name a few. In particular, these problems are more challenging when controllers are required to enforce high-level logic properties, *e.g.,* those expressed by automata [13]. Since closed-form solutions of synthesized policies for continuous-space stochastic systems are not available, a promising approach is to approximate these models by simpler ones with finite state sets (a.k.a. finite abstraction). A challenging step during this approximation phase is to provide formal guarantees such that the controller synthesized over (simpler) finite models can be refined back to original complex ones. In this chapter, I focus on synthesizing controllers over non-cooperative stochastic games leveraging abstraction-based approaches. These controllers will then be deployed as the safety advisor in the Safe-visor architecture in Chapter 4.

3.1.1 Related Works

There have been many results on the controller synthesis for discrete-time stochastic systems in the past few years. Results in [12, 40] focus on enforcing invariance properties for stochastic linear systems. As for nonlinear stochastic systems with continuous state and input sets, an abstraction-based approximation approach is initially proposed in [2]. This result is later improved in [174] regarding the scalability issue and extended in [184, 94, 95] for abstraction-based policy synthesis enforcing temporal logic properties characterized by deterministic finite automata. An (ϵ, δ)-approximate probabilistic relation is introduced in [76] to characterize the probabilistic dependency between an original model and its finite abstraction. With this type of relation, one can synthesize controllers enforcing the desired properties with less conservative lower bounds on probability of satisfaction. Later, results in [75, 74] propose Bellman operators for synthesizing controllers enforcing co-safe LTL_F properties [63] based on this relation. In the context of constructing finite abstractions for large-scale stochastic systems, compositional abstraction-based techniques have been introduced to alleviate the scalability issue due to discretizing the state sets; see for example [175, 121, 116, 120, 119, 148, 118].

Note that the above-mentioned works mainly focus on stochastic systems that are only affected by control inputs and noises. In some safety-critical real-life applications, systems are also affected by (rational) adversarial inputs, whose objectives are opposed to that of control inputs. In these scenarios, synthesis approaches for non-cooperative stochastic games [218] are required. Results in [179] handle synthesis problem for linear stochastic games based on iterative abstraction-refinement [97]. Results in [93, 52] utilize a grid-based approximation framework [1] to synthesize controllers for nonlinear stochastic games. Within the same framework, those results which are initially proposed for stochastic games with a finite or countably infinite number of states (*e.g.,* stochastic games with generalized mean-payoff objectives [42, 43], reachability objectives [110, 44, 67, 77], multiple objectives [18, 201, 109]) can also be employed to synthesize controllers for stochastic games with continuous state sets. However, the guarantee provided under this framework is sometimes very conservative (this is shown with an example in Section 3.4.3). There have also been some results to synthesize controllers for non-cooperative stochastic games (see e.g. [80, 142, 144, 70, 5]), but they are not applicable to enforce high-level temporal logic properties for nonlinear systems.

3.1.2 Contributions

In this chapter, I focus on synthesizing controllers for discrete-time nonlinear stochastic systems with continuous state and input sets over a finite time horizon. Particularly, I consider those systems modeled by zero-sum stochastic games [218], which are subject to not only control inputs but also (rational) adversarial inputs whose objective is opposed to that of control inputs. Moreover, I am interested in a class of complex logical properties expressed by deterministic finite automata (DFA), which are powerful in specifying behaviours that occurs within finite time [63]. Here, an abstraction-based technique is proposed to synthesize controllers based on an (ϵ, δ)-approximate probabilistic relation. Concretely, I first construct a finite abstraction of the original game and then establish such a relation between the finite abstraction and the original one. Then, by leveraging the probabilistic relation, new Bellman operators are proposed to synthesize a controller over the finite abstraction. Finally, this controller is refined back over the original game while providing probabilistic guarantees for the satisfaction of desired properties. Here, the contribution of this chapter is summarized as follows:

1. Given the notion of approximate probabilistic relations for stochastic games, new Bellman operators are proposed for synthesizing controllers over *nonlinear* stochastic games with continuous state and input sets. Leveraging the proposed operators, one can handle complex logic properties modeled by deterministic finite automata, while providing less conservative probabilistic guarantees for satisfying those properties compared with the results in [93, 52](cf. Section 3.4.3).

2. For a class of *nonlinear* stochastic games, a systematic algorithm is proposed to establish an approximate probabilistic relation between the original game and its abstraction. In comparison, results in [75, 74, 187] only establish such a relation for *linear* stochastic systems without rational adversarial inputs. Although [121]

provides the notion of approximate probabilistic relations for stochastic games, it does not provide any constructive algorithm for establishing the relation.

3. The proposed operators in this chapter can also be applied to synthesis problems for stochastic systems without adversarial inputs. In this case, compared with the operators in [74], the results in this chapter provide a less conservative probabilistic guarantee for satisfying the desired properties (cf. Lemma 3.3.8, Corollary 3.3.9, and Section 3.4.3).

3.1.3 Problem Formulation

Since the main target of this chapter is to synthesize controller over non-cooperative stochastic games, I deploy the general discrete-time stochastic games (gDTSG) introduced in Definition 2.4.3 as the modeling framework in this chapter. For robustness concern, an asymmetric information pattern is considered here that favors Player II, *i.e.*, Player II may select its action in a rational fashion based upon the choice of Player I. This results in a zero-sum Stackelberg game [38] with Player I as the leader, which is crucial for the existence of deterministic policies (cf. Definition 3.1.1, Remarks 3.1.2 and 3.3.7). Note that the setting here is common for robust control problems in which control inputs are selected considering that adversarial inputs are provided in a worst-case manner. The motivation for using such a setting is to provide formal probabilistic guarantees regardless of how adversarial inputs are chosen by Player II. This also indicates that Player II does not have to select adversarial inputs rationally in practice.

To formally formulate the main problem for this chapter, a few additional definitions are required. First, the notion of *Markov policy* for controlling the gDTSG is recalled.

Definition 3.1.1. (Markov Policy) *Consider a gDTSG $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$. A Markov policy ρ defined over the time horizon $[0, H-1] \subset \mathbb{N}$ for Player I is a sequence $\rho = (\rho_0, \rho_1, \dots, \rho_{H-1})$ of universally measurable maps $\rho_k : X \to \mathbf{P}(U, \mathcal{B}(U))$, with $\rho_k(U \mid x(k)) = 1$. Similarly, a Markov policy λ for Player II is a sequence $\lambda = (\lambda_0, \lambda_1, \dots, \lambda_{H-1})$ of universally measurable maps $\lambda_k : X \times U \to \mathbf{P}(W, \mathcal{B}(W))$, with $\lambda_k(W \mid x(k), u(k)) = 1$, for all $k \in [0, H-1]$. Here, $\mathcal{P}$ and Λ denote the set of all Markov policies for Players I and II, respectively. Moreover, $\mathcal{P}^H$ and Λ^H represent the set of all Markov policies for Players I and II within time horizon $[0, H-1]$, respectively.*

Remark 3.1.2. *In general, the Markov policy assigns a probability measure over $(U, \mathcal{B}(U))$ (resp. $(W, \mathcal{B}(W))$). From practical implementations' point of view,* nonrandomized *Markov policies [171, Definition 8.2] are of particular interest. In this chapter, by Markov policies, I refer to* nonrandomized *ones; otherwise, I explicitly say that the Markov policies are randomized ones.*

Next, a more general set of control strategies for Players I and II is defined. The definition here is adapted from [74] by allowing their output update map to be time dependent.

Definition 3.1.3. (Control Strategy) *A control strategy for Player I or II of a gDTSG $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ is a tuple*

$$\mathbf{C} = (\mathsf{M}, \mathsf{U}, \mathsf{Y}, \mathsf{H}, \mathsf{M}_0, \pi_{\mathsf{M}}, \pi_{\mathsf{Y}}), \tag{3.1.1}$$

where M *is a Borel set as the* memory state set*;* U *is a Borel set as the* observation set*;* Y *is a Borel set as the* output set*, which should be equal to U for Player I, and to W for Player II;* $\mathsf{H} \subseteq \mathbb{N}$ *is the* time domain*;* $\mathsf{M}_0 \subseteq \mathsf{M}$ *is the set of initial memory state;* $\pi_{\mathsf{M}} : \mathsf{M} \times \mathsf{U} \times \mathsf{H} \rightarrow \mathbf{P}(\mathsf{M}, \mathcal{B}(\mathsf{M}))$ *is a* memory update function*;* $\pi_{\mathsf{Y}} : \mathsf{M} \times \mathsf{H} \rightarrow \mathbf{P}(\mathsf{Y}, \mathcal{B}(\mathsf{Y}))$ *is an* output update function*.*

Remark 3.1.4. *A Markov policy $\rho = (\rho_0, \rho_1, \dots, \rho_{H-1})$ for Player I (resp. $\lambda = (\lambda_0, \lambda_1, \dots, \lambda_{H-1})$ for Player II) can be redefined as a control strategy $\mathbf{C} = (\mathsf{M}, \mathsf{U}, \mathsf{Y}, \mathsf{H}, \mathsf{M}_0, \pi_{\mathsf{M}}, \pi_{\mathsf{Y}})$ with $\mathsf{M} = \{\mathsf{m}\}$, where m is the sole element in M; $\mathsf{U} = X$ (resp. $\mathsf{U} = X \times U$); $\mathsf{Y} = U$ (resp. $\mathsf{Y} = W$); $\mathsf{H} = [0, H-1]$; $\mathsf{M}_0 = \{\mathsf{m}\}$; and $\pi_{\mathsf{Y}} := \rho_k$ (resp. $\pi_{\mathsf{Y}} := \lambda_k$) for all $k \in \mathsf{H}$.*

Given a gDTSG $\mathfrak{D}$, $(\rho,\lambda)\times\mathfrak{D}$ denotes the controlled gDTSG when $\mathfrak{D}$ is controlled by Markov policies ρ for Player I and λ for Player II. Analogously, consider a control strategy for Player I, denoted by $\mathbf{C}_\rho$, and a control strategy for Player II, denoted by $\mathbf{C}_\lambda$. The corresponding controlled gDTSG is denoted by $(\mathbf{C}_\rho,\mathbf{C}_\lambda)\times\mathfrak{D}$. With this notation, $\mathbb{P}_{(\rho,\lambda)\times\mathfrak{D}}$ (resp. $\mathbb{P}_{(\mathbf{C}_\rho,\mathbf{C}_\lambda)\times\mathfrak{D}}$) denotes the probability measure over the space of output sequences of the controlled gDTSG $(\rho,\lambda)\times\mathfrak{D}$ (resp. $(\mathbf{C}_\rho,\mathbf{C}_\lambda)\times\mathfrak{D}$).

In this chapter, I focus on synthesizing controller enforcing those logical properties modeled by deterministic finite automata (DFA), as introduced in Definition 2.5.1. To synthesize such controllers, the following definition is required for connecting the gDTSG $\mathfrak{D}$ as in (2.4.3) to a DFA $\mathcal{A}$ using a measurable labeling function.

Definition 3.1.5. (Labelling Function) *Consider a gDTSG $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$, a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$, and a finite output sequence $y_{\omega(H-1)} = (y(0), y(1), \dots, y(H-1)) \in Y^H$ of $\mathfrak{D}$ with some $H \in \mathbb{N}_{>0}$. The trace of $y_{\omega(H-1)}$ over Π is $\sigma = L_H(y_{\omega(H-1)}) = (\sigma_0, \sigma_1, \dots, \sigma_{H-1})$ with $\sigma_k = L(y(k))$ for all $k \in [0, H-1]$, where $L : Y \rightarrow \Pi$ is a measurable labeling function and $L_H : Y^H \rightarrow \Pi^H$ is a measurable function. Moreover, $y_{\omega(H-1)}$ is accepted by $\mathcal{A}$, denoted by $y_{\omega(H-1)} \models \mathcal{A}$, if $L_H(y_{\omega(H-1)}) \in \mathcal{L}(\mathcal{A})$.*

Throughout this chapter, $(\mathcal{A}, H)$ represent the property of interest, with $\mathcal{A}$ being a DFA and H being the finite time horizon over which the property should be satisfied. Accordingly, the satisfaction of a gDTSG $\mathfrak{D}$ with respect to this property is evaluated in terms of $\mathbb{P}_{\mathfrak{D}}\{y_{\omega(H-1)} \models \mathcal{A}\}$ within a bounded-time horizon, where $y_{\omega(H-1)}$ is the output sequences generated by $\mathfrak{D}$. For this purpose, one needs to construct a product gDTSG based on $\mathfrak{D}$ and $\mathcal{A}$, as defined below.

Definition 3.1.6. (Product gDTSG) *Consider a gDTSG $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$, a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$, and a labeling function $L : Y \rightarrow \Pi$ as in Definition 3.1.5. The product of $\mathfrak{D}$ and $\mathcal{A}$ is a gDTSG defined as $\mathfrak{D} \otimes \mathcal{A} = \{\bar{X}, \bar{U}, \bar{W}, \bar{X}_0, \bar{T}, \bar{Y}, \bar{h}\}$, where*

$\bar{X} := X \times Q$ is the state set; $\bar{U} := U$ is the input set for Player I; $\bar{W} := W$ is Player II's input set; $\bar{X}_0$ is the initial state set, with $\bar{x}_0 := (x_0, \bar{q}_0) \in \bar{X}_0$, $x_0 \in X_0$ and

$$\bar{q}_0 = \tau(q_0, L \circ h(x_0)); \tag{3.1.2}$$

$\bar{T}(\mathsf{d}x' \times \{q'\}|x, q, u, w)$ is the stochastic kernel that assigns for any $(x, q) \in \bar{X}$, $u \in \bar{U}$, and $w \in \bar{W}$ the probability $\bar{T}(\mathsf{d}x' \times \{q'\}|x, q, u, w) = T(\mathsf{d}x'|x, u, w)$ when $q' = \tau(q, L \circ h(x'))$, and $\bar{T}(\mathsf{d}x' \times \{q'\}|x, q, w, u) = 0$, otherwise; $\bar{Y} := Y$ is the output set and $\bar{h}(x, q) := h(x)$ is the output map.

With all definitions above, I am ready to formally define the problems of interest in this chapter. For some properties, *e.g.,* co-safe-LTL$_F$ [63], all infinite output sequences satisfying them have a finite good prefix [107, Section 2.2]. In this case, such properties are modeled with DFAs that accept all good prefixes. Accordingly, the lower bound of satisfaction probability is of interest, which yields a *problem of robust satisfaction.*

Problem 3.1.7. (Robust Satisfaction) *Consider a gDTSG $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ and the desired property $(\mathcal{A}, H)$. The* problem of robust satisfaction *is to design a control strategy $\mathbf{C}_\rho$ for Player I such that for any control strategy $\mathbf{C}_\lambda$ for Player II, one has*

$$\mathbb{P}_{(\mathbf{C}_\rho, \mathbf{C}_\lambda) \times \mathfrak{D}}\Big\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\Big\} \geq \mathbf{s}, \tag{3.1.3}$$

where $\mathbf{s}$ is the robust satisfaction probability *guaranteed by $\mathbf{C}_\rho$.*

Meanwhile, for some other logical properties, *e.g.,* safe-LTL$_F$ [166], all infinite output sequences that violate these properties have a finite bad prefix [107, Section 2.2]. Thus, such properties are modeled with DFAs that accept all bad prefixes, and an upper bound of the violation probability is of interest. This results in a *problem of worst-case violation* as defined below.

Problem 3.1.8. (Worst-case Violation) *Consider a gDTSG $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ and a property $(\mathcal{A}, H)$. The* problem of worst-case violation *is to design a control strategy $\mathbf{C}_\rho$ for Player I such that for any control strategy $\mathbf{C}_\lambda$ for Player II, one has*

$$\mathbb{P}_{(\mathbf{C}_\rho, \mathbf{C}_\lambda) \times \mathfrak{D}}\Big\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\Big\} \leq \mathbf{v}, \tag{3.1.4}$$

with $\mathbf{v}$ being the worst-case violation probability *ensured by $\mathbf{C}_\rho$.*

For a better illustration of the theoretical results in this chapter, the following running example is deployed throughout this chapter.

Running example. Consider the following gDTSG

$$\mathfrak{D}: \begin{cases} x(k+1) = Ax(k) + Bu(k) + E\sin(Fx(k)) + Dw(k) + R\varsigma(k), \\ y(k) = Cx(k), \qquad k \in \mathbb{N}, \end{cases}$$

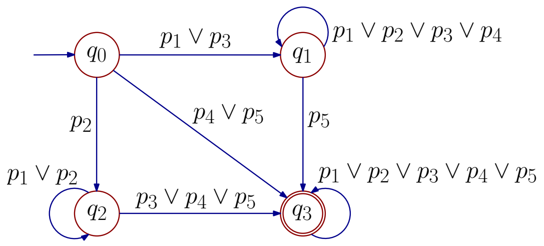


Figure 3.1: DFA for modeling ψ, with accepting state q_3, alphabet $\Pi = \{p_1, p_2, p_3, p_4, p_5\}$, and labeling function $L : Y \to \Pi$, where $L(y) = p_1$ when $y \in [0, 1.8]$, $L(y) = p_2$ when $y \in [-1.8, 0)$, $L(y) = p_3$ when $y \in (1.8, 2]$, $L(y) = p_4$ when $y \in [-2, -1.8)$, and $L(y) = p_5$ when $y \in (-\infty, -2) \cup (2, +\infty)$.

with

$$A=\begin{bmatrix}0.9204 & 0.4512 & 0.9491\\ 0.7865 & 0.8269 & 1.074\\ 0.6681 & 0.3393 & 0.5110\end{bmatrix}, \quad B=\begin{bmatrix}9.001 & 1.611 & 3.663\\ 1.404 & 11.76 & 2.386\\ 5.568 & 4.560 & 5.156\end{bmatrix},$$

$E = [0.6740; 0.6367; 0.7030]$, $D = [0.6; 0.4; 0.6]$, $R = [0.5110;\ 0.3347; 0.5336]$, $F = [0.5439; 0.9578; 0.2493]^\top$, and $C = [0.1;\ 0.1; 0.1]^\top$, where $x(t) = [x_1(k); x_2(k); x_3(k)]$ is the state, $u(k) \in [-2.5, 2.5]^3$ denotes the control input of Player I, $w(k) \in [-0.5, 0.5]$ denotes the adversarial input of Player II, $\varsigma(k)$ is a sequence of standard Gaussian random variables, and $y(k)$ is the output. Here, the following property ψ is considered: within 20 time steps (i.e., $H = 20$), if the output of the system starts from $[0, 2]$, it should stay within $[-2, 2]$; if it instead starts from $[-1.8, 0]$, it should then stay within $[-1.8, 1.8]$. The DFA for modeling ψ is shown in Figure 3.1. Accordingly, the problem of worst-case violation corresponding to this DFA is of interest.

3.2 Abstraction Synthesis for a Class of Nonlinear gDTSGs

In this section, a particular class of nonlinear gDTSG is considered, which is used to model many physical applications, such as fixed-joint robot [62], magnetic bearing [11], etc. This class of systems can be modeled as:

$$\mathfrak{D}: \begin{cases} x(k+1) = Ax(k) + Bu(k) + E\varphi(Fx(k)) + Dw(k) + R\varsigma(k), \\ y(k) = Cx(k), \qquad k \in \mathbb{N}, \end{cases} \tag{3.2.1}$$

with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $E \in \mathbb{R}^{n\times 1}$, $F \in \mathbb{R}^{1\times n}$, $D \in \mathbb{R}^{n\times p}$, $R \in \mathbb{R}^{n\times d}$, and $C \in \mathbb{R}^{q\times n}$. Here, it is assumed that the stochasticity $\varsigma : \mathbb{N} \to \mathbb{R}^d$ in (3.2.1) is a sequence of independent random vectors with multivariate standard normal distributions. Moreover, the nonlinearity $\varphi : \mathbb{R} \to \mathbb{R}$ satisfies

$$\underline{b} \leq \frac{\varphi(c) - \varphi(d)}{c - d} \leq \bar{b}, \tag{3.2.2}$$

for all $c, d \in \mathbb{R}$, $c \neq d$, with some $\underline{b}, \bar{b} \in \mathbb{R}$ where $\underline{b} \leq \bar{b}$. In the remainder of this chapter, the tuple

$$\mathfrak{D} = (A, B, C, D, E, F, R, \varphi), \tag{3.2.3}$$

is used to denote the systems as in (3.2.1). Next, in the following part of this section, I first discuss the construction of finite abstractions for this class of systems. Then, I propose how to establish an (ϵ, δ)-approximate probabilistic relation between original models and their corresponding abstractions.

Remark 3.2.1. *Note that although the main focus of this section is on establishing (ϵ, δ)-approximate probabilistic relations for a particular class of nonlinear gDTSGs as in* (3.2.1)*, the proposed results in 3.3 on synthesizing controllers are independent of the form of dynamics and are applicable to the general setting of gDTSGs.*

3.2.1 Construction of Finite Abstractions

Consider a gDTSG $\mathfrak{D}$ as in (3.2.3). I first introduce the construction of a reduced-order version of $\mathfrak{D}$, denoted by $\widehat{\mathfrak{D}}_{\mathsf{r}} = (\hat{A}_{\mathsf{r}}, \hat{B}_{\mathsf{r}}, \hat{C}_{\mathsf{r}}, \hat{D}_{\mathsf{r}}, \hat{E}_{\mathsf{r}}, \hat{F}_{\mathsf{r}}, \hat{R}_{\mathsf{r}}, \varphi)$, where the index r signifies the reduced-order version of the original game throughout this chapter. Then, I discuss how to build a finite abstraction, denoted by $\widehat{\mathfrak{D}}$, for $\widehat{\mathfrak{D}}_{\mathsf{r}}$. Note that the reduced-order gDTSG $\widehat{\mathfrak{D}}_{\mathsf{r}}$ is a simplified model of $\mathfrak{D}$, whose state and input sets are still continuous but with lower dimensions [167]. As a result, synthesizing controllers over reduced-order systems is more tractable than the original ones due to having less computational complexity (cf. Remark 3.2.3).

To construct the reduced-order model $\widehat{\mathfrak{D}}_{\mathsf{r}}$ for the gDTSG $\mathfrak{D}$ as in (3.2.3), one first needs to select an abstraction matrix $P \in \mathbb{R}^{n\times\hat{n}}$ that maps the states of the abstraction to the $\mathfrak{D}$ as follows

$$x = P\hat{x}. \tag{3.2.4}$$

Here, $\hat{n}$ denotes the dimension of the state space for $\widehat{\mathfrak{D}}_{\mathsf{r}}$, $x \in X \subseteq \mathbb{R}^n$ is the state of $\mathfrak{D}$, and $\hat{x} \in \hat{X} \subseteq \mathbb{R}^{\hat{n}}$ is the state of $\widehat{\mathfrak{D}}_{\mathsf{r}}$. With abstraction matrix P, the reduced-order game $\widehat{\mathfrak{D}}_{\mathsf{r}}$ can be constructed as long as the following equations hold for some matrices G, Q, and S with appropriate dimensions:

$$\hat{C}_{\mathsf{r}} = CP, \tag{3.2.5}$$

$$\hat{F}_{\mathsf{r}} = FP, \tag{3.2.6}$$

$$E = P\hat{E}_{\mathsf{r}} - BG, \tag{3.2.7}$$

$$AP = P\hat{A}_{\mathsf{r}} - BQ, \tag{3.2.8}$$

$$D = P\hat{D}_{\mathsf{r}} - BS. \tag{3.2.9}$$

Conditions (3.2.5) to (3.2.9) are similar to [121, conditions (5.5b) to (5.5f)]. I will discuss later (cf. (3.2.36)) how to select $\hat{R}_{\mathsf{r}}$ such that it is easier to establish an approximate probabilistic relation between the original game and its abstraction. Additionally, there is no restriction on the choice of $\hat{B}_{\mathsf{r}}$. For instant, one can choose $\hat{B}_{\mathsf{r}} = \mathbf{I}_{\hat{n}}$ so that $\widehat{\mathfrak{D}}_{\mathsf{r}}$ is fully actuated, and, hence, solving the synthesis problem over it get easier.

Remark 3.2.2. *Note that considering matrices A, E, B, and P. There exist matrices $\hat{A}_r$, $\hat{E}_r$, G, Q, and S satisfying* (3.2.7) *to* (3.2.9) *if and only if [208, Lemma 5.10 and Lemma 5.12]* $\operatorname{im} AP \subseteq \operatorname{im} P + \operatorname{im} B$*,* $\operatorname{im} D \subseteq \operatorname{im} P + \operatorname{im} B$*, and* $\operatorname{im} E \subseteq \operatorname{im} P + \operatorname{im} B$*.*

Next, I proceed with discussing the construction of a finite abstraction $\widehat{\mathfrak{D}}$ of $\widehat{\mathfrak{D}}_{\mathsf{r}}$. To this end, one needs the notion of *region of interest*, denoted by $\hat{X}_{rs}$, which is a compact subset of $\hat{X}_{\mathsf{r}}$. Note that this is usually the case for physical systems in practice, where variables evolve in a bounded domain. Accordingly, it is assumed that $\widehat{\mathfrak{D}}_{\mathsf{r}}$ will not come back to $\hat{X}_{rs}$ once it leaves $\hat{X}_{rs}$. Instead, it will stay in a single absorbing state, denoted by ϕ. With these notions, one first partitions $\hat{X}_{\mathsf{r}}$ with $\hat{X}_{\mathsf{r}} = \cup_{i\in\mathbb{N}} X_i$ and correspondingly selects representative points $\hat{x}_i \in X_i$ for each cell, where X_i are bounded cells. Then, the set $\hat{X} = \{\Pi'(X_i) \mid X_i \cap (\hat{X}_{\mathsf{r}} \backslash \hat{X}_{rs}) = \emptyset\} \cup \{\phi\}$ is used as the state set of $\widehat{\mathfrak{D}}$, with Π' being a function that maps X_i to its representative points, and ϕ represents the aggregation of representative points in the set

$$\big\{\Pi'(X_i) \mid X_i \cap (\hat{X}_{\mathsf{r}} \backslash \hat{X}_{rs}) \neq \emptyset\big\}. \tag{3.2.10}$$

For the sake of succinctness, I use $\hat{X} = \{\hat{x}_i\}_{i=1}^{n_x} \cup \{\phi\}$ with n_x being the number of representative points in the set $\{\Pi'(X_i) \mid X_i \cap (\hat{X}_{\mathsf{r}} \backslash \hat{X}_{rs}) = \emptyset\}$. Additionally, $\tilde{\Pi}_x$ is defined to maps any $\hat{x}_{\mathsf{r}} \in \hat{X}_{\mathsf{r}}$ to $\hat{x}_i = \Pi'(X_i)$ with $\hat{x}_{\mathsf{r}} \in X_i$, based on the set

$$\Delta := \big\{\tilde{\Pi}_x(\hat{x}_{\mathsf{r}}) - \hat{x}_{\mathsf{r}} \,|\, \hat{x}_{\mathsf{r}} \in \hat{X}_{\mathsf{r}}\big\}. \tag{3.2.11}$$

is defined. Since the partitions of $\hat{X}_{\mathsf{r}}$ are bounded, Δ is also bounded, namely, there exists $\delta' \in (-\infty, +\infty)$ such that $\forall \beta \in \Delta, |\beta| \leq \delta'$.

Following the same idea for constructing the finite state set, the finite input set of Player I and Player II are constructed by first selecting bounded partitions $\hat{U}_{\mathsf{r}} = \cup_{n_u} U_i$ and $\hat{W}_{\mathsf{r}} = \cup_{n_w} W_i$, and then choosing representative points $\hat{u}_i \in U_i$ and $\hat{w}_i \in W_i$. Accordingly, one gets $\hat{U} = \{\hat{u}_m\}_{m=1}^{n_u}$ being the input set for Player I and $\hat{W} = \{\hat{w}_l\}_{l=1}^{n_w}$ being the input set for Player II. Similar to $\tilde{\Pi}_x$ as in (3.2.11), a function $\Pi_w : \hat{W}_{\mathsf{r}} \to \hat{W}$ is also defined to map any $\hat{w}_{\mathsf{r}} \in \hat{W}_{\mathsf{r}}$ to its representative point $\hat{w} \in \hat{W}$ of the partition that contains $\hat{w}_{\mathsf{r}}$. Accordingly, one can define a bounded set

$$\Delta_w := \big\{\Pi_w(\hat{w}_{\mathsf{r}}) - \hat{w}_{\mathsf{r}} \,|\, \hat{w}_{\mathsf{r}} \in \hat{W}_{\mathsf{r}}\big\}. \tag{3.2.12}$$

The dynamic of $\widehat{\mathfrak{D}}$ is constructed according to the dynamic of $\widehat{\mathfrak{D}}_{\mathsf{r}}$ and the characteristic of ϕ, *i.e.*, $\hat{x}_{\mathsf{r}}(k+1) = \hat{f}_{\mathsf{r}}(\hat{x}_{\mathsf{r}}(k), \hat{u}_{\mathsf{r}}(k), \hat{w}_{\mathsf{r}}(k), \varsigma(k)) = A_{\mathsf{r}}\hat{x}_{\mathsf{r}}(k) + E_{\mathsf{r}}\varphi(F_{\mathsf{r}}\hat{x}_{\mathsf{r}}(k)) + D_{\mathsf{r}}\hat{w}_{\mathsf{r}}(k) + B_{\mathsf{r}}\hat{u}_{\mathsf{r}}(k) + R_{\mathsf{r}}\varsigma(k)$ when $\hat{x}_{\mathsf{r}}(k) \in \hat{X}_{rs}$ and $\hat{x}_{\mathsf{r}}(k+1) = \hat{f}_{\mathsf{r}}(\hat{x}_{\mathsf{r}}(k), \hat{u}_{\mathsf{r}}(k), \hat{w}_{\mathsf{r}}(k), \varsigma(k)) = \phi$ when $\hat{x}_{\mathsf{r}}(k) = \phi$. Concretely, the dynamic of $\widehat{\mathfrak{D}}$ is given by

$$\hat{f}(\hat{x}(k), \hat{u}(k), \hat{w}(k), \varsigma(k)) := \Pi_x(\hat{f}_{\mathsf{r}}(\hat{x}_{\mathsf{r}}(k), \hat{u}_{\mathsf{r}}(k), \hat{w}_{\mathsf{r}}(k), \varsigma(k)), \tag{3.2.13}$$

where $\Pi_x : \hat{X}_{\mathsf{r}} \to \hat{X}$ is the map that assigns to any $\hat{x}_{\mathsf{r}} \in \hat{X}_{\mathsf{rs}}$ the representative point $\hat{x} \in \hat{X}$ of the corresponding partition set containing $\hat{x}_{\mathsf{r}}$, and assigns any $\hat{x}_{\mathsf{r}} \in \hat{X}_{\mathsf{rs}}^c$ to ϕ. The output map is $\hat{y} = \hat{C}_{\mathsf{r}}\hat{x}$ when $\hat{x} \neq \phi$, and $\hat{y} = \phi_y$ when $\hat{x} = \phi$, where ϕ_y represents the output when $\hat{x} = \phi$. Then, one can rewrite (3.2.13) as

$$\hat{f}(\hat{x}(k), \hat{u}(k), \hat{w}(k), \varsigma(k)) := \hat{f}_{\mathsf{r}}(\hat{x}_{\mathsf{r}}(k), \hat{u}_{\mathsf{r}}(k), \hat{w}_{\mathsf{r}}(k), \varsigma(k)) + \beta,\ \beta \in \Delta.$$

Finally, the initial state set of $\widehat{\mathfrak{D}}$ is defined as $\hat{X}_0 := \{\hat{x}_0 \in \hat{X} \mid \hat{x}_0 = \Pi_x(\hat{x}_{\mathsf{r}0}), \hat{x}_{\mathsf{r}0} \in \hat{X}_{\mathsf{r}0}\}$, where $\hat{X}_{\mathsf{r}0}$ is the initial state set of $\widehat{\mathfrak{D}}_{\mathsf{r}}$, and the stochastic kernel $\hat{T}$ is computed as

$$\hat{T}(\hat{x}_h \mid \hat{x}_{h'}, \hat{u}_m, \hat{w}_l) = \begin{cases} T(X_h|\hat{x}_{h'}, \hat{u}_m, \hat{w}_l), & \text{if } \hat{x}_{h'}, \hat{x}_h \in \{\hat{x}_i\}_{i=1}^{n_x}, \\ T(\hat{X}_{rs}^c|\hat{x}_{h'}, \hat{u}_m, \hat{w}_l), & \text{if } \hat{x}_{h'} \in \{\hat{x}_i\}_{i=1}^{n_x}, \hat{x}_h = \phi, \\ 1, & \text{if } \hat{x}_{h'}, \hat{x}_h = \phi, \\ 0, & \text{if } \hat{x}_{h'} = \phi, \hat{x}_h \in \{\hat{x}_i\}_{i=1}^{n_x}, \end{cases} \tag{3.2.14}$$

with $h, h' \in [1, n_x]$, $\hat{x}_h = \Pi'(X_h)$, $\hat{u}_m \in \hat{U}$, and $\hat{w}_l \in \hat{W}$.

Remark 3.2.3. *If the finite abstraction is directly constructed from original gDTSG, the size of $\hat{T}$ grows exponentially with the dimension of original state and input sets. As a promising alternative, by constructing a reduced-order version of original gDTSG, the finite abstraction can be built based on gDTSG with a lower dimension, which alleviates the encountered computational complexity (cf. Section 3.4.1). One can also apply compositional techniques proposed in [121] for constructing finite abstractions of large-scale gDTSGs via abstractions of smaller subsystems, and utilize the techniques proposed in [114, Section 4.2] to further reduce the memory usage required for storing the stochastic kernel of finite abstractions.*

3.2.2 Conditions for Establishing Approximate Probabilistic Relations

Since the formal safety guarantees provided by the abstraction-based controller synthesis methodologies rely on an (ϵ, δ) approximate probabilistic relations between the original gDTSG $\mathfrak{D}$ and its finite abstraction $\widehat{\mathfrak{D}}$. In this subsection, I show under which conditions $\widehat{\mathfrak{D}}$ is (ϵ, δ)-stochastically simulated by $\mathfrak{D}$, denoted by $\widehat{\mathfrak{D}} \preceq_{\epsilon}^{\delta} \mathfrak{D}$ (cf. Definition 2.4.6), w.r.t. relations $\mathscr{R}$ and $\mathscr{R}_w$ defined as

$$\mathscr{R} = \big\{(x, \hat{x}) \,|\, (x - P\hat{x})^\top M (x - P\hat{x}) \le \epsilon^2\big\}, \tag{3.2.15}$$

$$\mathscr{R}_w = \big\{(w, \hat{w}) \,|\, (w - \hat{w})^\top \tilde{M} (w - \hat{w}) \le \tilde{\epsilon}^2\big\}, \tag{3.2.16}$$

where M and $\tilde{M}$ are positive-definite matrices with appropriate dimensions, and ϵ, $\tilde{\epsilon} \in \mathbb{R}_{>0}$. Prior to proposing the required conditions, I raise the following definition.

Definition 3.2.4. *Consider a gDTSG $\mathfrak{D} = (A, B, C, D, E, F, R, \varphi)$ as in* (3.2.3)*, its reduced-order version $\widehat{\mathfrak{D}}_r = (\hat{A}_r, \hat{B}_r, \hat{C}_r, \hat{D}_r, \hat{E}_r, \hat{F}_r, \hat{R}_r, \varphi)$ with the same additive noise, a finite abstraction $\widehat{\mathfrak{D}}$ constructed from $\widehat{\mathfrak{D}}_r$, and relations $\mathscr{R}$ and $\mathscr{R}_w$ as in* (3.2.15) *and* (3.2.16)*, respectively. For any $\kappa \in \mathbb{R}_{\ge 0}$, and matrices $K, L \in \mathbb{R}^{m \times n}$, consider the following conditions:*

$$M \succeq C^\top C, \tag{3.2.17}$$

$$(A + BK)^\top M (A + BK) \preceq \kappa M, \tag{3.2.18}$$

$$\bar{A}^\top M \bar{A} \preceq \kappa M, \tag{3.2.19}$$

$$\underline{A}^\top M \underline{A} \preceq \kappa M, \tag{3.2.20}$$

$$\sqrt{\kappa} \leq 1 - \tilde{\gamma}/\epsilon \leq 1, \tag{3.2.21}$$

in which $\bar{A} := A + BK + \bar{b}(BL + EF)$ *and* $\underline{A} := A + BK + \underline{b}(BL + EF)$ *with* $\underline{b}$ *and* $\bar{b}$ *as appeared in* (3.2.2)*, respectively, and* $\tilde{\gamma} := \gamma_0 + \gamma_1 + \gamma_2 + \gamma_3 + \gamma_4$ *with*

$$\gamma_0 := \underset{\bar{w},\, \|\bar{w}\|_{\tilde{M}} \leq \tilde{\epsilon}}{\arg\max} \|D\bar{w}\|_M, \tag{3.2.22}$$

$$\gamma_1 := \underset{\hat{u} \in \hat{U}'}{\arg\max} \|(B\tilde{R} - P\hat{B}_r)\hat{u}\|_M, \tag{3.2.23}$$

$$\gamma_2 := \underset{\varsigma,\, \|\varsigma\| \leq \chi_d^{-1}(1-\delta)}{\arg\max} \|(R - P\hat{R}_r)\varsigma\|_M, \tag{3.2.24}$$

$$\gamma_3 := \underset{\beta \in \Delta}{\arg\max} \|P\beta\|_M, \tag{3.2.25}$$

$$\gamma_4 := \underset{\hat{w} \in \hat{W}}{\arg\max} \|BS\hat{w}\|_M. \tag{3.2.26}$$

In (3.2.22)-(3.2.26)*,* $\chi_d^{-1} : [0,1] \to \mathbb{R}$ *is the chi-square inverse cumulative distribution function with* d *degrees of freedom [22],* $\|\bar{x}\|_M := \sqrt{\bar{x}^\top M \bar{x}}$*,* Δ *is as in* (3.2.11)*, and* $\hat{U}' \subseteq \hat{U}$ *is the input set for* $\widehat{\mathfrak{D}}$*.*

With Definition 3.2.4, the required conditions under which one has $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$ with respect to the relations as in (3.2.15) and (3.2.16) are introduced as below.

Theorem 3.2.5. *Consider a gDTSG* $\mathfrak{D}$ *and its finite abstraction* $\widehat{\mathfrak{D}}$ *constructed from* $\widehat{\mathfrak{D}}_r$*. For any* $x_0 \in X_0$ *and* $\hat{x}_0 \in \hat{X}_0$ *with* $(x_0, \hat{x}_0) \in \mathscr{R}$*,* $\widehat{\mathfrak{D}}$ *is* (ϵ, δ)*-stochastically simulated by* $\mathfrak{D}$ *(i.e.,* $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$*) with respect to the relations as in* (3.2.15) *and* (3.2.16)*, if*

- (**Cd.1**) *there exist* $\kappa \in \mathbb{R}_{\geq 0}$ *and* $K, L \in \mathbb{R}^{m \times n}$*, such that conditions in* (3.2.17)-(3.2.21) *holds;*
- (**Cd.2**) *the associated interface function is*

$$\nu(x, \hat{x}, \hat{u}) := (K + b(x, \hat{x})L)(x - P\hat{x}) + Q\hat{x} + \tilde{R}\hat{u} + G\varphi(FP\hat{x}), \tag{3.2.27}$$

  with P*,* Q*, and* G *being as in* (3.2.4)*,* (3.2.8)*, and* (3.2.7)*, respectively,* K *and* L *being as in* (**Cd.1**) *above,* $\tilde{R}$ *being a matrix with an appropriate dimension,*

$$b(x, \hat{x}) = \frac{\varphi(Fx) - \varphi(FP\hat{x})}{F(x - P\hat{x})} \in [\underline{b}, \bar{b}],$$

  if $x \neq P\hat{x}$*, with* $\underline{b}$ *and* $\bar{b}$ *appeared in* (3.2.2)*, and* $b(x, \hat{x}) = 0$ *otherwise;*
- (**Cd.3**) $\hat{U}'$ *in* (3.2.23) *is constructed such that* $\forall \hat{u} \in \hat{U}'$ *and* $\forall (x, \hat{x}) \in \mathscr{R}$*, one has* $\nu(x, \hat{x}, \hat{u}) \in U$*.*

The proof of Theorem 3.2.5 is provided in Section 3.6.1.

Remark 3.2.6. *Given an initial state x_0 of $\mathfrak{D}$, if there exists $\hat{x}_0 \in \hat{X}$ such that $(x_0, \hat{x}_0) \in \mathscr{R}$, one can choose $\hat{x}_0 = \Pi_x((P^\top MP)^{-1}P^\top Mx_0)$, which minimizes $\|x_0 - P\hat{x}_0\|_M$. Moreover, there is no restriction on $\tilde{R}$ in* (3.2.27) *in general. However, $\tilde{R} = (B^\top B)^{-1}B^\top P\hat{B}_r$ is recommended to obtain a smaller γ_1 as in* (3.2.23)*. Then, it gets easier to find $\tilde{\gamma}$ and ϵ such that an approximate probabilistic relation exists (cf.* (3.2.35)*).*

The next result shows that under which conditions, (3.2.17)-(3.2.21) hold.

Corollary 3.2.7. *Consider a gDTSG $\mathfrak{D} = (A, B, C, D, E, F, R, \varphi)$. There exist M, K, L, ϵ, and $\tilde{\gamma}$ such that* (3.2.17)-(3.2.21) *hold if and only if for all $b' \in \{\underline{b}, \bar{b}, 0\}$, the pair $(A + b'EF, B)$ is stabilizable, where $\underline{b}$ and $\bar{b}$ appeared in* (3.2.2)*.*

The proof of Corollary 3.2.7 is given in Section 3.6.1. So far, I have introduced conditions under which there exists an approximate probabilistic relation between a gDTSG and its finite abstraction. Next, an algorithmic procedure is proposed to establish such a relation based on those conditions.

3.2.3 Algorithmic Procedure for Establishing Approximate Probabilistic Relations

In this subsection, an algorithmic procedure is proposed to search for M, K, L, and ϵ in Definition 3.2.4 given the following items s.t. (**Cd.1**)-(**Cd.3**) in Theorem 3.2.5 hold:

1. δ in the approximate probabilistic relation;
2. a tolerable range for ϵ, denoted by $[\epsilon_{min}, \epsilon_{max}]$;
3. the finite abstraction constructed as in Section 3.2.1;
4. the set $\hat{U}'$ as in Definition 3.2.4 for synthesizing the controller over the finite abstraction.

Here, I first discuss how to accommodate $\hat{U}'$ when searching for M, K, L, and ϵ so that (**Cd.3**) holds. Then, I will investigate how to jointly compute M, K, and L given candidates $\epsilon \in [\epsilon_{min}, \epsilon_{max}]$ and $\kappa \in [0, 1]$, with κ appeared in Definition 3.2.4. Finally, the algorithmic procedure for establishing the approximate probabilistic relations will be formally proposed.

Accommodating $\hat{\mathbf{U}}'$. Here, it is assumed that all $\hat{u} \in \hat{U}$ are within a polytope defined by a matrix inequality

$$A_u\hat{u} \le b_u, \tag{3.2.28}$$

where $A_u \in \mathbb{R}^{r\times m}$ and $b_u \in \mathbb{R}^{r\times 1}$. Note that the input set of the form of (3.2.28) is appropriate for many physical systems. Next, by substituting the interface function as in (3.2.27) into (3.2.28), one can rewrite (3.2.28) as

$$A_u\bar{u} \le \tilde{b}_u(\hat{x}, \hat{u}), \tag{3.2.29}$$

with $\tilde{b}_u(\hat{x}, \hat{u}) = b_u - A_u(Q\hat{x} + \tilde{R}\hat{u} + G\varphi(FP\hat{x}))$ and $\bar{u} = (K + bL)\bar{x}$, where $\bar{x} = x - P\hat{x}$. One can readily see that every pair $(\hat{x}, \hat{u})$ corresponds to a polytope for $\bar{u}$ specified by A_u and $\tilde{b}_u(\hat{x},\hat{u})$, with $\hat{x}\in\hat{X}$ and $\hat{u}\in\hat{U}'$. Here, A denotes the set of all possible polytopes of the form of (3.2.29) given $\hat{X}$ and $\hat{U}'$, and

$$\tilde{A}\bar{u}\le\tilde{b}, \tag{3.2.30}$$

denotes a polytope $\tilde{\mathsf{A}}:=\cap_{r=1}^{a}\mathsf{A}_r$, in which $\tilde{A}\in\mathbb{R}^{r\times m}$, $\tilde{b}\in\mathbb{R}^{r\times 1}$, and a is the number of polytopes within A. This polytope can be computed by multi-parametric toolbox `MPT` [78]. Now, one can rewrite the polytope in (3.2.30) as:

$$\alpha_i\bar{u}\le 1,\quad i\in\{1,\dots,r\}, \tag{3.2.31}$$

with $\alpha_i=\frac{1}{\tilde{b}_i}\tilde{A}_i$, where $\tilde{A}_i$ and $\tilde{b}_i$ are the i-th row of $\tilde{A}$ and $\tilde{b}$, respectively. Now, I am ready to introduce Theorem 3.2.8, which accommodates (3.2.31) in the search for M, K, L, and ϵ.

Theorem 3.2.8. *Consider a series of constraints as in* (3.2.31) *for* $\bar{u}:=(K+bL)\bar{x}$, *with* $\bar{x}\in\mathbb{R}^n$. *For all* $\bar{x}\in E_x$ *with* $E_x:=\{\bar{x}\mid\bar{x}^\top M\bar{x}\le\epsilon^2\}$, $M\in\mathbb{R}^{n\times n}$, *and* $\epsilon\in\mathbb{R}_{>0}$, *constraints as in* (3.2.31) *are satisfied for all* $b\in[\underline{b},\bar{b}]\cup\{0\}$ *iff*

$$\alpha_i K\bar{M}K^\top\alpha_i^\top\le 1/\epsilon^2, \tag{3.2.32}$$
$$\alpha_i(K+\underline{b}L)\bar{M}(K+\underline{b}L)^\top\alpha_i^\top\le 1/\epsilon^2, \tag{3.2.33}$$
$$\alpha_i(K+\bar{b}L)\bar{M}(K+\bar{b}L)^\top\alpha_i^\top\le 1/\epsilon^2, \tag{3.2.34}$$

with $\underline{b}$ *and* $\bar{b}$ *appeared in* (3.2.2), $\bar{M}=M^{-1}$, *and* $i\in\{1,\dots,r\}$.

The proof of Theorem 3.2.8 is provided in Section 3.6.1. Next, I proceed with studying how to apply Theorem 3.2.8 when searching for M, K, and L.

Jointly Computing M, K, and L. Consider (3.2.17)-(3.2.20) and (3.2.32)-(3.2.34). When ϵ and κ are fixed, M, K, and L can be computed (if existing) by solving a semidefinite (SDP) programming problem [185]. Accordingly, one can first uniformly select samples from $[\epsilon_{min},\epsilon_{max}]$ and $[0,1]$ as candidates for ϵ and κ, respectively, and then try to compute M, K, and L for each (ϵ, κ) sample pairs. The next corollary shows how to compute M, K, and L jointly, given δ and a sample pair (ϵ, κ).

Corollary 3.2.9. *Consider a gDTSG* $\mathfrak{D}=(A,B,C,D,E,F,R,\varphi)$, *input constraints as in* (3.2.31), δ *as in the approximate probabilistic relation, candidates* $\epsilon\in[\epsilon_{min},\epsilon_{max}]$, *and* $\kappa\in[0,1]$. *Matrix* M *as in* (3.2.15) *as well as* K *and* L *as in* (3.2.27) *can be computed jointly by solving the convex optimization problem:*

$$\begin{aligned}\min_{\bar{M}}\quad & -\log(\det(\bar{M}))\\ s.t.\quad & \bar{M}\succ 0;\\ & \begin{bmatrix}\bar{M} & \bar{M}C^\top\\ C\bar{M} & \mathbf{I}_q\end{bmatrix}\succeq 0;\begin{bmatrix}\bar{M} & \bar{A}_b\\ \bar{A}_b^\top & \kappa\bar{M}\end{bmatrix}\succeq 0,\ b\in\{\underline{b},\bar{b},0\};\end{aligned}$$

$$\begin{bmatrix} 1/\epsilon^2 & \alpha_i(\bar{K}+b\bar{L}) \\ (\bar{K}+b\bar{L})^\top\alpha_i^\top & \bar{M} \end{bmatrix} \succeq 0, \ \textit{with } i \in \{1,\dots,r\} \textit{ and } b \in \{\underline{b}, \bar{b}, 0\};$$

where $\bar{A}_b = (A + bEF)\bar{M} + B(\bar{K} + b\bar{L})$ *and* $\det(\bar{M})$ *is the determinate of* $\bar{M}$*, with* $\bar{M}$*,* $\bar{K}$*, and* $\bar{L}$ *being matrices with appropriate dimensions. If there is a solution for this optimization problem, one can compute* M*,* K*, and* L *as* $M = \bar{M}^{-1}$*,* $K = \bar{K}M$*, and* $L = \bar{L}M$*, respectively, and one has* $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$*, if*

$$\tilde{\gamma} \leq \epsilon(1 - \sqrt{\kappa}), \tag{3.2.35}$$

with $\tilde{\gamma}$ *being computed as in Definition 3.2.4.*

Corollary 3.2.9 is a direct result of Theorems 3.2.5 and 3.2.8 with Schur complement [35]. Additionally, one can design $\hat{R}_r$ as

$$\hat{R}_r = (P^\top MP)^{-1}P^\top MR, \tag{3.2.36}$$

to minimize γ_2 for the selected δ. Finally, Algorithm 1 summarizes the solution to systematically establish an approximate probabilistic relation.

Algorithm 1: Establishing an approximate probabilistic relation between a stochastic game and its abstraction

```
1 Select matrix $P$ as in Remark 3.2.2, compute $\hat{C}_{\mathsf{r}}$, $\hat{F}_{\mathsf{r}}$, $\hat{E}_{\mathsf{r}}$, $\hat{A}_{\mathsf{r}}$, and $\hat{D}_{\mathsf{r}}$ following
    (3.2.5)-(3.2.9), and choose $\hat{B}_{\mathsf{r}}$ freely;
2 Discretize the state set as well as the input sets of Player I and Player II, and
    then select $\tilde{M}$ and $\tilde{\epsilon}$ in (3.2.16) according to the discretization of $\hat{W}_{\mathsf{r}}$;
3 Select $\hat{U}' \subseteq \hat{U}$ for synthesizing controllers over the finite abstraction, compute
    $\tilde{R}$ in (3.2.27) according to Remark 3.2.6, and compute constraints in (3.2.31);
4 Select $\delta$ and appropriate interval $[\epsilon_{min}, \epsilon_{max}]$. Then, uniformly select samples
    of $\epsilon$ within $[\epsilon_{min}, \epsilon_{max}]$ and $\kappa$ within [0,1]. For each $(\epsilon, \kappa)$,
    (a) Compute $M$, $K$, and $L$ as in Corollary 3.2.9;
    (b) If there are solutions in Step 4(i), compute $\hat{R}_r$ as in (3.2.36);
    (c) Compute $\tilde{\gamma}$ as in Definition 3.2.4 and check (3.2.35) accordingly;
    (d) If (3.2.35) in Step 4(c) holds, solutions for $M$,$K$,$L$, and $\epsilon$ are founded for
        establishing the relation.
```

Remark 3.2.10. *The number of constraints in the optimization problem in Corollary 3.2.9 grows linearly with the dimension of the system and* r *as in* (3.2.31)*. In practice, this problem can be solved efficiently with existing SDP solvers such as* `SDPT3` *[185].*

Running example (continued). For constructing the finite abstraction, $P = [0.6199; 0.4443; 0.6219]$ is chosen, and the reduced-order game is constructed with $\hat{A}_{\mathsf{r}} = 0.55$, $\hat{B}_{\mathsf{r}} = 1$, $\hat{D}_{\mathsf{r}} = 1$, $\hat{E}_r = 0.32$, $\hat{F}_r = 0.7957$, and $\hat{C}_r = 0.1686$. One therefore has $G = [-0.0334; -0.0311; -0.0342]$, $Q = [-0.1617; -0.1269; 0.1877]$, and $S =$

$[0.0021; 0.0038; -0.0014]$ as in (3.2.7) to (3.2.9). The finite abstraction for the reduced-order game is constructed as in Table 3.2, with $[-1.5, 1.5]$ and $[-0.5, 0.5]$ being selected as the input set of Player I and II respectively. Based on the discretization of the Player II's input set, $\tilde{M} = 1$ and $\tilde{\epsilon} = 0.05$ are chosen. As for establishing the (ϵ,δ)-approximate probabilistic relation, I choose $\hat{U}' = \hat{U}$, $\delta = 0.001$, and the set for ϵ as $[0.05, 1]$. Then, by applying Algorithm 1, the finite abstraction is (ϵ, δ)-stochastically simulated by the original model with $\epsilon = 0.1509$,

$$M = \begin{bmatrix} 0.0132 & 0.0082 & 0.0146 \\ 0.0082 & 0.0110 & 0.0074 \\ 0.0146 & 0.0074 & 0.0188 \end{bmatrix},$$

$\hat{R}_{\mathsf{r}} = 0.8256$, and the interface function as in (3.2.27) with

$$K = \begin{bmatrix} -0.1163 & -0.0355 & -0.0999 \\ -0.0367 & -0.0499 & -0.0514 \\ 0.0222 & -0.0215 & 0.0125 \end{bmatrix}, \quad L = \begin{bmatrix} -0.0450 & -0.0824 & -0.0200 \\ -0.0682 & -0.0761 & -0.0573 \\ 0.0524 & 0.0666 & 0.0378 \end{bmatrix},$$

and $\tilde{R} = [0.0422; 0.0213; 0.0562]$.

3.3 Controller Synthesis Problem

In this section, I focus on elaborating the synthesis of controller $\tilde{\mathbf{C}}_\rho$ for a gDTSG $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ for Problems 3.1.7 and 3.1.8, given a finite abstraction $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ of $\mathfrak{D}$ with $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$, and a property $(\mathcal{A}, H)$, with $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$. The general idea of the proposed methods is depicted in Figure 3.2 and summarized as

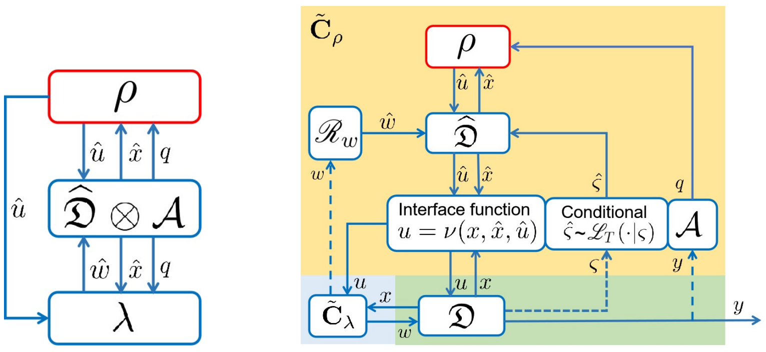

ρ
$\hat{u}$
$\hat{x}$
q
$\widehat{\mathfrak{D}} \otimes \mathcal{A}$
$\hat{w}$
λ
$\tilde{\mathbf{C}}_\rho$
$\mathscr{R}_w$
$\widehat{\mathfrak{D}}$
w
$\hat{\varsigma}$
Interface function $u = \nu(x, \hat{x}, \hat{u})$
Conditional $\hat{\varsigma} \sim \mathscr{L}_T(\cdot|\varsigma)$
$\mathcal{A}$
ς
y
u
x
$\tilde{\mathbf{C}}_\lambda$
$\mathfrak{D}$

Figure 3.2: Left: Synthesizing Markov policy for $\widehat{\mathfrak{D}} \otimes \mathcal{A}$. **Right:** Construction of $\tilde{\mathbf{C}}_\rho$ (yellow region).

follows:

- As shown in Figure 3.2 (left), one first synthesizes a Markov policy ρ for Player I of the gDTSG $\widehat{\mathfrak{D}} \otimes \mathcal{A}$, assuming that Player II of the gDTSG selects its actions in a rational fashion against the choice of Player I. The outcomes are the Markov policy ρ and the robust satisfaction probability $\mathbf{s}$ for Problem 3.1.7 (resp. worst-case violation probability $\mathbf{v}$ for Problem 3.1.8);

- One then constructs $\tilde{\mathbf{C}}_\rho$ based on ρ (cf. Definition 3.3.1) as depicted in Figure 3.2 (right). At runtime, when a state x of $\mathfrak{D}$ is fed to $\tilde{\mathbf{C}}_\rho$:
    1. State $\hat{x}$ of $\widehat{\mathfrak{D}}$ is first updated according to x, the conditional stochastic kernel $\mathscr{L}_T$, and the action w of $\tilde{\mathbf{C}}_\lambda$ in the previous time instant. Then, the state q of $\mathcal{A}$ are updated according to the output function $h(x)$ of $\mathfrak{D}$ and the transition function τ of $\mathcal{A}$;
    2. Afterwards, a $\hat{u}$ is provided by ρ based on $\hat{x}$ and q, and refined to $\mathfrak{D}$ by virtue of the interface function ν;
    3. $\tilde{\mathbf{C}}_\lambda$ selects w according to x and u, and feeds w to $\mathfrak{D}$.

Here, the construction of $\tilde{\mathbf{C}}_\rho$ is formally presented as follows.

Definition 3.3.1. (Construction of $\tilde{\mathbf{C}}_\rho$) *Consider gDTSGs* $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ *and* $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ *with* $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$*. Given a Markov policy* $\rho = (\rho_0, \rho_1, \ldots, \rho_{H-1})$ *for Player I of* $\widehat{\mathfrak{D}} \otimes \mathcal{A}$*,* $\tilde{\mathbf{C}}_\rho = (\tilde{\mathsf{M}}, \tilde{\mathsf{U}}, \tilde{\mathsf{Y}}, \tilde{\mathsf{H}}, \tilde{\mathsf{M}}_0, \tilde{\pi}_{\mathsf{M}}, \tilde{\pi}_{\mathsf{Y}})$ *is constructed for Player I of* $\mathfrak{D}$ *with* $\tilde{\mathsf{M}} = X \times \hat{X} \times Q \times W \times \hat{W}$, $\tilde{\mathsf{U}} = X \times W$, $\tilde{\mathsf{Y}} = U$, $\tilde{\mathsf{H}} = [0, H-1]$,

- $\tilde{\mathsf{m}}_0 = \big(\tilde{\mathsf{m}}_X(0), \tilde{\mathsf{m}}_{\hat{X}}(0), \tilde{\mathsf{m}}_Q(0), \tilde{\mathsf{m}}_W(0), \tilde{\mathsf{m}}_{\hat{W}}(0)\big) \in \tilde{\mathsf{M}}_0$*, with* $\tilde{\mathsf{m}}_X(0) = x_0$*, where* $x_0 \in X_0$*;* $\tilde{\mathsf{m}}_{\hat{X}}(0) = \hat{x}_0$ *such that* $(x_0, \hat{x}_0) \in \mathscr{R}$*, where* $\mathscr{R}$ *is as in* (3.2.15)*;* $\tilde{\mathsf{m}}_Q(0) = \tau\big(q_0, L \circ h(\tilde{\mathsf{m}}_X(0))\big)$*;* $\tilde{\mathsf{m}}_W(0)$ *is initialized as* $\tilde{\mathsf{m}}_W(0) = w(0)$ *after Player II of* $\mathfrak{D}$ *has chosen* $w(0)$*, and* $\tilde{\mathsf{m}}_{\hat{W}}(0)$ *is accordingly initialized as* $\tilde{\mathsf{m}}_{\hat{W}}(0) = \Pi_w(w(0))$ *with* Π_w *as in* (2.4.6)*;*
- $\tilde{\pi}_{\mathsf{M}}$ *updates* $\big(\tilde{\mathsf{m}}_X(k), \tilde{\mathsf{m}}_{\hat{X}}(k), \tilde{\mathsf{m}}_Q(k), \tilde{\mathsf{m}}_W(k), \tilde{\mathsf{m}}_{\hat{W}}(k)\big) \in \tilde{\mathsf{M}}$ *at all time instants* $k \in \tilde{\mathsf{H}} \backslash \{0\}$*, with the following steps:*
    1. *update* $\tilde{\mathsf{m}}_{\hat{X}}(k)$ *according to the conditional kernel:*
    $$\mathscr{L}_T\big(d\hat{x} | \tilde{\mathsf{m}}_{\hat{X}}(k-1), \tilde{\mathsf{m}}_X(k-1), x(k), \hat{u}(k-1), \tilde{\mathsf{m}}_W(k-1)\big),$$
    where $x(k)$ *is the state of* $\mathfrak{D}$*,* $\hat{u}(k-1) = \tilde{\rho}_k(\tilde{\mathsf{m}}_X(k-1), \tilde{\mathsf{m}}_{\hat{X}}(k-1), \tilde{\mathsf{m}}_Q(k-1))$*, and* $\mathscr{L}_T(\cdot)$ *as in* (2.4.7)*;*
    2. *update* $\tilde{\mathsf{m}}_X(k)$ *with* $\tilde{\mathsf{m}}_X(k) = x(k)$*;*
    3. *update* $\tilde{\mathsf{m}}_Q(k)$ *with* $\tilde{\mathsf{m}}_Q(k) = \tau\big(\tilde{\mathsf{m}}_Q(k-1), L \circ h(\tilde{\mathsf{m}}_X(k))\big)$*;*
    4. *update* $\tilde{\mathsf{m}}_W(k)$ *with* $\tilde{\mathsf{m}}_W(k) = w(k)$ *after Player II of* $\mathfrak{D}$ *has selected* $w(k)$*, and accordingly update* $\tilde{\mathsf{m}}_{\hat{W}}(k)$ *as* $\tilde{\mathsf{m}}_{\hat{W}}(k) = \Pi_w(w(k))$ *with* Π_w *as in* (2.4.6)*;*
- $\tilde{\pi}_{\mathsf{Y}}$ *updates* $\mathsf{y}(k) \in \tilde{\mathsf{Y}}$ *at the time instant* $k \in \tilde{\mathsf{H}}$ *with*
$$\mathsf{y}(k) = \nu\big(\tilde{\mathsf{m}}_X(k), \tilde{\mathsf{m}}_{\hat{X}}(k), \rho_k(\tilde{\mathsf{m}}_{\hat{X}}(k), \tilde{\mathsf{m}}_Q(k))\big),$$
where ν *is the interface function associated with the (*ϵ*,*δ*)-approximate probabilistic relation.*

The remaining problem is how to synthesize the Markov policy ρ for $\widehat{\mathfrak{D}} \otimes \mathcal{A}$. In Sections 3.3.1 and 3.3.2, new Bellman operators will be proposed to synthesize ρ for Problems 3.1.7 and 3.1.8, respectively. Prior to introducing these operators, I would like to point out that these operators require the following assumption.

Assumption 3.3.2. *Consider gDTSGs $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ and $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$ regarding relations $\mathscr{R}$ and $\mathscr{R}_w$ as in Definition 2.4.8. For all $\hat{w} \in \hat{W}$ and $w \in W$ with $(w, \hat{w}) \in \mathscr{R}_w$, it is assumed that*

$$\int_{\bar{\mathscr{R}}_{\hat{x}'}} \mathscr{L}_T(\mathsf{d}x'|x, \hat{x}, \hat{x}', \nu(x, \hat{x}, \hat{u}), w, \hat{w}) \geq 1 - \delta,$$

holds $\forall \hat{x}, \hat{x}' \in \hat{X}$, with $\bar{\mathscr{R}}_{\hat{x}'} = \{x' \in X | (x', \hat{x}') \in \mathscr{R}\}$ and $\mathscr{L}_T(\mathsf{d}x'|x,\ \hat{x}, \hat{x}', \nu(x, \hat{x}, \hat{u}), w, \hat{w})$ as the conditional probability of $x' \in X$ given $x \in X$, $\hat{x}$, $\hat{x}'$, w, $\hat{w}$, and the interface function $\nu(x, \hat{x}, \hat{u})$.

Remark 3.3.3. *Assumption 3.3.2 presumes that all states of $\widehat{\mathfrak{D}}$ are coupled into the δ-lifted relation, and at every time instant k, $\mathbb{P}\{(x', \hat{x}') \in \mathscr{R} | (x, \hat{x}) \in \mathscr{R}, \forall (w, \hat{w}) \in \mathscr{R}_w\} \geq 1 - \delta$ holds for all $\hat{x} \in \hat{X}$ via the interface function used in controller refinement, with $(x, \hat{x})$ and $(x', \hat{x}')$ being the state pairs at time instants k and $k+1$, respectively. Given the existing results on (ϵ,δ)-approximate probabilistic relations [74, 121, 187], Assumption 3.3.2 does not introduce extra subtlety in practice. In fact, although the results in [74, 121, 187] do not explicitly require such an assumption, the existence of an (ϵ,δ) approximate probabilistic relation is guaranteed by enforcing Assumption 3.3.2 (cf. [74, Condition A3], [121, Theorem 5.5] and [187, Theorem 3]).*

3.3.1 Robust Satisfaction Problem

Here, I first start with discussing how to synthesize the Markov policy ρ for the problem of robust satisfaction as in Problem 3.1.7. Consider a gDTSG $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ and its finite abstraction $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$, a property $(\mathcal{A}, H)$ with $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$, and the product gDTSG $\widehat{\mathfrak{D}} \otimes \mathcal{A} = \{\bar{X}, \bar{U}, \bar{W}, \bar{X}_0, \bar{T}, \bar{Y}, \bar{h}\}$ as in Definition 3.1.6. Given a Markov policy $\rho = (\rho_0, \rho_1, \ldots, \rho_{H-1})$ for Player I and $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{H-1})$ for Player II of $\widehat{\mathfrak{D}} \otimes \mathcal{A}$, a cost-to-go function $\bar{V}^{\rho,\lambda}_n : \hat{X} \times Q \to [0, 1]$ is defined, which assigns a real number to states of $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ at the time instant $H - n$. Then, $\bar{V}^{\rho,\lambda}_{n+1}(\hat{x}, q)$ is initialized with $\bar{V}^{\rho,\lambda}_0(\hat{x}, q) = 1$ when $q \in F$ and $\bar{V}^{\rho,\lambda}_0(\hat{x}, q) = 0$, otherwise, and it can be recursively computed as

$$\bar{V}^{\rho,\lambda}_{n+1}(\hat{x}, q) = \mathfrak{P}(\bar{V}^{\rho,\lambda}_n)(\hat{x}, q). \tag{3.3.1}$$

Here, $\mathfrak{P}$ is a Bellman operator defined as

$$\mathfrak{P}(\bar{V}^{\rho,\lambda}_n)(\hat{x}, q) := \begin{cases} (1-\delta) \sum\limits_{\hat{x}' \in \hat{X}} \bar{V}^{\rho,\lambda}_n(\hat{x}', \underline{q}(\hat{x}', q)) \hat{T}(\hat{x}' | \hat{x}, \hat{u}, \hat{w}), & \text{if } q \notin F; \\ 1, & \text{if } q \in F, \end{cases} \tag{3.3.2}$$

with $\hat{u} = \rho_{H-n-1}(\hat{x}, q)$, $\hat{w} = \lambda_{H-n-1}(\hat{x}, q, \hat{u})$, and

$$\underline{q}(\hat{x}', q) = \arg\min_{q' \in Q'_{\epsilon}(\hat{x}')} \bar{V}^{\rho,\lambda}_n(\hat{x}', q'), \tag{3.3.3}$$

where

$$Q'_\epsilon(\hat{x}') := \Big\{ q' \in Q \mid \exists x \in X, q' = \tau(q, L \circ h(x)), \text{with } h(x) \in N_\epsilon(\hat{h}(\hat{x}')) \Big\}, \tag{3.3.4}$$

and $\mathcal{N}_\epsilon(\hat{y}) := \{y \in Y \mid \|y - \hat{y}\| \leq \epsilon\}$. Moreover, given a Markov policy ρ for Player I, the corresponding worst-case adversarial policy $\lambda_*(\rho)$ for Player II can be computed as

$$\lambda_{*_{H-n-1}}(\rho) \in \min_{\lambda'_{H-n-1} \in \Lambda} (1-\delta) \sum_{\hat{x}' \in \hat{X}} \bar{V}_n^{\rho, \lambda_*(\rho)}(\hat{x}', \underline{q}(\hat{x}', q)) \hat{T}(\hat{x}' | \hat{x}, \hat{u}, \hat{w}), \tag{3.3.5}$$

for all $n \in [0, H-1]$, with $\hat{u} = \rho_{H-n-1}(\hat{x}, q)$ and $\hat{w} = \lambda'_{H-n-1}(\hat{x},\, q, \hat{u})$. Now, I am ready to propose one of the main results for the problem of robust satisfaction.

Theorem 3.3.4. *Consider gDTSGs $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ and $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$, and a property $(\mathcal{A}, H)$ with $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$. Given a Markov policy ρ designed for Player I of $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ and a control strategy $\tilde{\mathbf{C}}_\rho$ for Player I of $\mathfrak{D}$ that is constructed based on ρ as in Definition 3.3.1, for any control strategy $\tilde{\mathbf{C}}_\lambda$ for Player II of $\mathfrak{D}$, one has*

$$\mathbb{P}_{(\tilde{\mathbf{C}}_\rho, \tilde{\mathbf{C}}_\lambda) \times \mathfrak{D}} \big\{ \exists k \leq H, y_{\omega k} \models \mathcal{A} \big\} \geq \bar{V}_H^{\rho, \lambda_*(\rho)}(\hat{x}_0, \bar{q}_0), \tag{3.3.6}$$

where $\hat{x}_0 \in \hat{X}_0$ and $x_0 \in X_0$, with $(x_0, \hat{x}_0) \in \mathscr{R}$ and $\mathscr{R}$ as in (3.2.15), $\bar{V}_H^{\rho, \lambda_*(\rho)}(\hat{x}_0, \bar{q}_0)$ is computed as in (3.3.1), with $\lambda_*(\rho)$ as in (3.3.5) and $\bar{q}_0 = \tau(q_0, L \circ h(x_0))$.

The proof of Theorem 3.3.4 is provided in Section 3.6.2.1. In practice, it is of particular interest to construct a ρ that maximizes the robust satisfaction probability, *i.e.*, $\bar{V}_H^{\rho, \lambda_*(\rho)}(\hat{x}_0, \bar{q}_0)$ as in (3.3.6). One can leverage the following proposition to synthesize such a policy.

Proposition 3.3.5. *Consider gDTSGs $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ and $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$, and a property $(\mathcal{A}, H)$ with $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$. Considering that Player II minimizes $\bar{V}_{n+1}^{\rho, \lambda}(\hat{x}, q)$ according to ρ, the Markov policy $\rho^* = (\rho_0^*, \rho_1^*, \ldots, \rho_{H-1}^*)$ for Player I maximizes $\bar{V}_{n+1}^{\rho, \lambda}(\hat{x}, q)$, with*

$$\rho_{H-n-1}^* \in \arg \max_{\rho_{H-n-1} \in \mathcal{P}} \min_{\lambda_{H-n-1} \in \Lambda} (1-\delta) \sum_{\hat{x}' \in \hat{X}} \bar{V}_n^*(\hat{x}', \underline{q}^*(\hat{x}', q)) \hat{T}(\hat{x}' | \hat{x}, \hat{u}, \hat{w}), \tag{3.3.7}$$

for all $n \in [0, H-1]$, where $\hat{u} = \rho_{H-n-1}(\hat{x}, q)$ and $\hat{w} = \lambda_{H-n-1}(\hat{x}, q, \hat{u})$. Here,

$$\bar{V}_H^*(\hat{x}, q) := \max_{\rho \in \mathcal{P}^H} \min_{\lambda \in \Lambda^H} \bar{V}_H^{\rho, \lambda}(\hat{x}, q), \tag{3.3.8}$$

denotes the cost-to-go function associated with ρ^*.

Similar to (3.3.1), by initializing $\bar{V}_0^*(\hat{x},q)=1$ when $q\in F$, and $\bar{V}_0^*(\hat{x},q)=0$ otherwise, $\bar{V}_n^*(x,q)$ in (3.3.8) can be recursively computed as

$$\bar{V}_{n+1}^*(\hat{x},q)=\mathfrak{P}^*(\bar{V}_n^*)(\hat{x},q), \tag{3.3.9}$$

with $\mathfrak{P}^*$ being a Bellman operator defined as

$$\mathfrak{P}^*(\bar{V}_n^*)(\hat{x},q):=\begin{cases}\max\limits_{\rho_{H-n-1}\in\mathcal{P}}\min\limits_{\lambda_{H-n-1}\in\Lambda}(1-\delta)\sum\limits_{\hat{x}'\in\hat{X}}\bar{V}_n^*(\hat{x}',\underline{q}^*(\hat{x}',q))\hat{T}(\hat{x}'|\hat{x},\hat{u},\hat{w}), & \text{if } q\notin F;\\ 1, & \text{if } q\in F,\end{cases} \tag{3.3.10}$$

where $\hat{u}=\rho_{H-n-1}(\hat{x},q)$, $\hat{w}=\lambda_{H-n-1}(\hat{x},q,\hat{u})$, and

$$\underline{q}^*(\hat{x}',q)=\arg\min_{q'\in Q'_\epsilon(\hat{x}')}\bar{V}_n^*(\hat{x}',q'), \tag{3.3.11}$$

with $Q'_\epsilon(\hat{x}')$ being the set as in (3.3.4). With these notions, the following corollary associates ρ^* as in (3.3.7) with its corresponding robust satisfaction probability.

Corollary 3.3.6. *Consider gDTSGs $\mathfrak{D}=(X,U,W,X_0,T,Y,h)$ and $\widehat{\mathfrak{D}}=(\hat{X},\hat{U},\hat{W},\hat{X}_0,\hat{T},Y,\hat{h})$ with $\widehat{\mathfrak{D}}\preceq_\epsilon^\delta\mathfrak{D}$, and the desired property $(\mathcal{A},H)$ with $\mathcal{A}=(Q,q_0,\Pi,\tau,F)$. Given a Markov policy ρ^* synthesized for Player I of $\widehat{\mathfrak{D}}\otimes\mathcal{A}$ as in* (3.3.7)*, and a control strategy $\tilde{\mathbf{C}}_{\rho^*}$ for Player I of $\mathfrak{D}$ that is constructed based on ρ^* as in Definition 3.3.1, for any control strategy $\tilde{\mathbf{C}}_\lambda$ for Player II of $\mathfrak{D}$, one has*

$$\mathbb{P}_{(\tilde{\mathbf{C}}_{\rho^*},\tilde{\mathbf{C}}_\lambda)\times\mathfrak{D}}\Big\{\exists k\leq H, y_{\omega k}\models\mathcal{A}\Big\}\geq\bar{V}_H^*(\hat{x}_0,\bar{q}_0), \tag{3.3.12}$$

where $\hat{x}_0\in\hat{X}_0$ and $x_0\in X_0$, with $(x_0,\hat{x}_0)\in\mathscr{R}$ and $\mathscr{R}$ as in (3.2.15)*, $\bar{V}_H^*(\hat{x}_0,\bar{q}_0)$ is as in* (3.3.9) *with $\bar{q}_0=\tau(q_0,L\circ h(x_0))$.*

Note that Corollary 3.3.6 holds since Theorem 3.3.4 is valid for any arbitrary Markov policy ρ for Player I of $\widehat{\mathfrak{D}}\otimes\mathcal{A}$. Therefore, the probabilistic guarantee associated with ρ^* as in (3.3.7) can also be preserved for $\mathfrak{D}$.

Remark 3.3.7. *Given the zero-sum Stackelberg game setting with Player I as leader, Markovian stochastic kernel of $\mathfrak{D}\otimes\mathcal{A}$ as in Definition 3.1.6, and sum-multiplicative utility function as in* (3.3.10)*, there always exists a deterministic [38, Section 5.1] and Markovian [162, Section 4] policy as in* (3.3.7)*. In particular, considering Markov policy is sufficient here thanks to the sum-multiplicative utility function as constructed in* (3.3.10) *and the Markovian stochastic kernel T of $\mathfrak{D}$, which results in a Markovian stochastic kernel for the product $\widehat{\mathfrak{D}}\otimes\mathcal{A}$. Note that a similar deduction can also be applied to the corresponding policy for the worst-case violation problem, which is introduced later (cf.* (3.3.23) *and* (3.3.20)*).*

Finally, it is also worth noting that operators in (3.3.2) and (3.3.10) can readily be applied to synthesis problems for stochastic systems *without rational adversarial inputs*. In this case, thanks to Assumption 3.3.2, one can consider all states of finite abstraction $\widehat{\mathfrak{D}}$ in the proposed Bellman operators (instead of only a part of these states as the setting in [74]). Accordingly, the operator in (3.3.2) provides less conservative probabilistic guarantees than the one proposed in [74, equation (41)], which is formally shown with the following lemma.

Lemma 3.3.8. *Consider a property* $(\mathcal{A}, H)$ *in which* $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$, *gDTSGs* $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ *and* $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ *with* $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$ *and* $W = \hat{W} = \{\mathbf{0}_p\}$. *Given a Markov policy* ρ *designed for Player I of* $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ *and a control strategy* $\tilde{\mathbf{C}}_\rho$ *for Player I of* $\mathfrak{D}$ *that is constructed based on* ρ *as in Definition 3.3.1, one has*

$$\mathbb{P}_{(\tilde{\mathbf{C}}_\rho, \tilde{\mathbf{C}}_\lambda) \times \mathfrak{D}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\} \geq \bar{V}_H^{\rho,\lambda}(\hat{x}_0, \bar{q}_0) \geq \mathcal{S}(\hat{x}_0), \tag{3.3.13}$$

where $\hat{x}_0 \in \hat{X}_0$ *and* $x_0 \in X_0$, *with* $(x_0, \hat{x}_0) \in \mathscr{R}$ *and* $\mathscr{R}$ *as in* (3.2.15), $\tilde{\mathbf{C}}_\lambda(\mathsf{m}) \equiv \mathbf{0}_p$ *with* m *as the memory state of* $\tilde{\mathbf{C}}$, $\bar{V}_H^{\rho,\lambda}(\hat{x}_0, \bar{q}_0)$ *is as in* (3.3.1) *with* $\lambda(\hat{x}, \hat{u}) \equiv 0$, $\bar{q}_0 = \tau(q_0, L \circ h(x_0))$, *and* $\mathcal{S}(\hat{x}_0)$ *is the probabilistic guarantee provided by the operator in [74, equation (41)].*

The proof of Lemma 3.3.8 is provided in in Section 3.6.2.2. Similarly, the following corollary shows that the operator in (3.3.10) also provides less conservative probabilistic guarantees than the one proposed in [74, equation (42)].

Corollary 3.3.9. *Given a Markov policy* ρ^* *synthesized for Player I of* $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ *as in* (3.3.7), *and a control strategy* $\tilde{\mathbf{C}}_{\rho^*}$ *for Player I of* $\mathfrak{D}$ *that is constructed based on* ρ^* *as in Definition 3.3.1, one has*

$$\mathbb{P}_{(\tilde{\mathbf{C}}_{\rho^*}, \tilde{\mathbf{C}}_\lambda) \times \mathfrak{D}}\Big\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\Big\} \geq \bar{V}_H^*(\hat{x}_0, \bar{q}_0) \geq \mathcal{S}^*(\hat{x}_0), \tag{3.3.14}$$

where $\hat{x}_0 \in \hat{X}_0$ *and* $x_0 \in X_0$, *with* $(x_0, \hat{x}_0) \in \mathscr{R}$ *and* $\mathscr{R}$ *as in* (3.2.15), $\tilde{\mathbf{C}}_\lambda(\mathsf{m}) \equiv \mathbf{0}_p$ *with* m *as the memory state of* $\tilde{\mathbf{C}}$, $\bar{V}_H^*(\hat{x}_0, \bar{q}_0)$ *is as in* (3.3.9) *with* $\lambda(\hat{x}, \hat{u}) \equiv 0$, $\bar{q}_0 = \tau(q_0, L \circ h(x_0))$, *and* $\mathcal{S}^*(\hat{x}_0)$ *is the probabilistic guarantee provided by the operator in [74, equation (42)].*

The proof of Corollary 3.3.9 is similar to that of Lemma 3.3.8. The results in Corollary 3.3.9 will later be illustrated with an example in Section 3.4.3. Next, I proceed with proposing the results for the problem of worst-case violation.

3.3.2 Worst-case Violation Problem

In this subsection, the controller synthesis regarding Problem 3.1.8 is discussed. Consider a gDTSG $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ and its finite abstraction $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$, a property $(\mathcal{A}, H)$, and a product gDTSG $\widehat{\mathfrak{D}} \otimes \mathcal{A} = \{\bar{X}, \bar{U}, \bar{W}, \bar{X}_0, \bar{T}, \bar{Y}, \bar{h}\}$. Given a Markov policy $\rho = (\rho_0, \rho_1, \ldots, \rho_{H-1})$ for Player I and $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{H-1})$ for Player II of $\widehat{\mathfrak{D}} \otimes \mathcal{A}$, a cost-to go function $\underline{V}_n^{\rho,\lambda} : \hat{X} \times Q \to [0, 1]$ is defined, which maps each state of $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ at the time instant $H - n$ to a real number. Then, $\underline{V}_{n+1}^{\rho,\lambda}(\hat{x}, q)$ is recursively computed as

$$\underline{V}_{n+1}^{\rho,\lambda}(\hat{x}, q) = \mathfrak{T}(\underline{V}_n^{\rho,\lambda})(\hat{x}, q), \tag{3.3.15}$$

initialized by $\underline{V}_0^{\rho,\lambda}(\hat{x}, q) = 1$ when $q \in F$, and $\underline{V}_0^{\rho,\lambda}(\hat{x}, q) = 0$, otherwise. Here, $\mathfrak{T}$ is a Bellman operator defined as

$$\mathfrak{T}(\underline{V}_n^{\rho,\lambda})(\hat{x}, q) := \begin{cases} (1-\delta) \sum\limits_{\hat{x}' \in \hat{X}} \underline{V}_n^{\rho,\lambda}(\hat{x}', \bar{q}(\hat{x}', q)) \hat{T}(\hat{x}' | \hat{x}, \hat{u}, \hat{w}) + \delta, & \text{if } q \notin F; \\ 1, & \text{if } q \in F, \end{cases} \tag{3.3.16}$$

where $\hat{u} = \rho_{H-n-1}(\hat{x}, q)$, $\hat{w} = \lambda_{H-n-1}(\hat{x}, q, \hat{u})$, and

$$\overline{q}(\hat{x}', q) = \arg\max_{q' \in Q'_\epsilon(\hat{x}')} \underline{V}_n^{\rho,\lambda}(\hat{x}', q'), \tag{3.3.17}$$

with $Q'_\epsilon(\hat{x}')$ as in (3.3.4). Additionally, one can compute the worst-case adversarial policy $\lambda^*(\rho)$ for Player II with respect to the Markov policy ρ for Player I as

$$\lambda^*_{H-n-1}(\rho) \in \max_{\lambda_{H-n-1} \in \Lambda} \Big((1-\delta) \sum_{\hat{x}' \in \hat{X}} \underline{V}_n^{\rho,\lambda^*(\rho)}(\hat{x}', \overline{q}(\hat{x}', q)) \hat{T}(\hat{x}' | \hat{x}, \hat{u}, \hat{w}) + \delta\Big), \tag{3.3.18}$$

for all $n \in [0, H-1]$, with $\hat{u} = \rho_{H-n-1}(\hat{x}, q)$, and $\hat{w} = \lambda_{H-n-1}(\hat{x}, q, \hat{u})$. Now, I am ready to propose in the next theorem the main result corresponding to the problem of worst-case violation.

Theorem 3.3.10. *Consider gDTSGs $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ and $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$, and a property $(\mathcal{A}, H)$ in which $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$. Given a Markov policy ρ for Player I of $\widehat{\mathfrak{D}} \otimes \mathcal{A}$, and a control strategy $\tilde{\mathbf{C}}_\rho$ for Player I of $\mathfrak{D}$ that is constructed based on ρ as in Definition 3.3.1, for any control strategy $\tilde{\mathbf{C}}_\lambda$ for Player II of $\mathfrak{D}$, one has*

$$\mathbb{P}_{(\tilde{\mathbf{C}}_\rho, \tilde{\mathbf{C}}_\lambda) \times \mathfrak{D}} \Big\{ \exists k \leq H, y_{\omega k} \models \mathcal{A} \Big\} \leq \underline{V}_H^{\rho,\lambda^*(\rho)}(\hat{x}_0, \bar{q}_0), \tag{3.3.19}$$

where $\hat{x}_0 \in \hat{X}_0$ and $x_0 \in X_0$, with $(x_0, \hat{x}_0) \in \mathscr{R}$ and $\mathscr{R}$ as in (3.2.15), $\underline{V}_H^{\rho,\lambda_*(\rho)}(\hat{x}_0, \bar{q}_0)$ is computed as in (3.3.15), with $\lambda^*(\rho)$ as in (3.3.18) and $\bar{q}_0 = \tau(q_0, L \circ h(x_0))$.

The proof of Theorem 3.3.10 is provided in Section 3.6.2.3. In practice, synthesizing a ρ that minimizes the worst-case violation probability, *i.e.*, $\underline{V}_H^{\rho,\lambda^*(\rho)}(\hat{x}_0, \bar{q}_0)$ as in (3.3.19), is of particular interest. The following proposition shows how such a Markov policy can be synthesized.

Proposition 3.3.11. *Consider gDTSGs* $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ *and* $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ *with* $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$*, and a property* $(\mathcal{A}, H)$ *in which* $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$*. Consider that Player II is assumed to be able to maximize* $\underline{V}_{n+1}^{\rho,\lambda}(\hat{x}, q)$ *according to* ρ*. The Markov policy* $\rho_* = (\rho_{*0}, \rho_{*1}, \ldots, \rho_{*H-1})$ *for Player I minimizes* $\underline{V}_{n+1}^{\rho,\lambda}(\hat{x}, q)$*, with*

$$\rho_{*H-n-1} \in \arg \min_{\rho_{H-n-1}\in\mathcal{P}} \max_{\lambda_{H-n-1}\in\Lambda} \Big((1-\delta)\sum_{\hat{x}'\in\hat{X}} \underline{V}_{*,n}(\hat{x}', \overline{q}_*(\hat{x}', q))\hat{T}(\hat{x}'|\hat{x}, \hat{u}, \hat{w}) + \delta\Big), \tag{3.3.20}$$

for all $n \in [0, H-1]$*, where* $\hat{u} = \rho_{H-n-1}(\hat{x}, q)$ *and* $\hat{w} = \lambda_{H-n-1}(\hat{x}, q, \hat{u})$*. Here,*

$$\underline{V}_{*,H}(\hat{x}, q) := \min_{\rho\in\mathcal{P}^H} \max_{\lambda\in\Lambda^H} \underline{V}_H^{\rho,\lambda}(\hat{x}, q), \tag{3.3.21}$$

denotes the cost-to-go function associated with ρ_**.*

Analogous to (3.3.15), $\underline{V}_{*,n}(\hat{x}, q)$ is initialized with $\underline{V}_{*,0}(\hat{x}, q) = 1$ when $q \in F$, and $\underline{V}_{*,0}(\hat{x}, q) = 0$ when $q \notin F$. Then, $\underline{V}_{*,n}(\hat{x}, q)$ can be recursively computed as

$$\underline{V}_{*,n+1}(\hat{x}, q) = \mathfrak{T}_*(\underline{V}_{*,n})(\hat{x}, q), \tag{3.3.22}$$

where $\mathfrak{T}_*$ is a Bellman operator defined as

$$\mathfrak{T}_*(\underline{V}_{*,n})(\hat{x}, q) := \begin{cases} \min\limits_{\rho_{H-n-1}\in\mathcal{P}} \max\limits_{\lambda_{H-n-1}\in\Lambda} \Big((1-\delta)\sum\limits_{\hat{x}'\in\hat{X}} \underline{V}_{*,n}(\hat{x}', \overline{q}_*(\hat{x}', q))\hat{T}(\hat{x}'|\hat{x}, \hat{u}, \hat{w}) + \delta\Big), & \text{if } q \notin F; \\ 1, & \text{if } q \in F, \end{cases} \tag{3.3.23}$$

with $\hat{u} = \rho_{H-n-1}(\hat{x}, q)$, $\hat{w} = \lambda_{H-n-1}(\hat{x}, q, \hat{u})$,

$$\overline{q}_*(\hat{x}', q) = \arg\max_{q'\in Q'_\epsilon(\hat{x}')} \underline{V}_{*,n}(\hat{x}', q'), \tag{3.3.24}$$

and $Q'_\epsilon(\hat{x}')$ as in (3.3.4). Note that Theorem 3.3.10 holds for any arbitrary Markov policy ρ for Player I of $\widehat{\mathfrak{D}} \otimes \mathcal{A}$. Thus, the probabilistic guarantee associated with ρ_* as in (3.3.20) can also be preserved for $\mathfrak{D}$. This preservation is formally proposed in the following corollary.

Corollary 3.3.12. *Consider gDTSGs* $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ *and* $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ *with* $\widehat{\mathfrak{D}} \preceq_{\epsilon}^{\delta} \mathfrak{D}$, *and a property* $(\mathcal{A}, H)$ *with* $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$. *Given a Markov policy* ρ_* *synthesized for Player I of* $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ *as in* (3.3.20), *and a control strategy* $\tilde{\mathbf{C}}_{\rho_*}$ *for Player I of* $\mathfrak{D}$ *that is constructed based on* ρ_* *as in Definition 3.3.1, for any control strategy* $\tilde{\mathbf{C}}_{\lambda}$ *for Player II of* $\mathfrak{D}$, *one has*

$$\mathbb{P}_{(\tilde{\mathbf{C}}_{\rho_*}, \tilde{\mathbf{C}}_{\lambda}) \times \mathfrak{D}} \{\exists k \leq H, y_{\omega k} \models \mathcal{A}\} \leq \underline{V}_{*,H}(\hat{x}_0, \bar{q}_0), \tag{3.3.25}$$

where $\hat{x}_0 \in \hat{X}_0$ *and* $x_0 \in X_0$, *with* $(x_0, \hat{x}_0) \in \mathscr{R}$ *and* $\mathscr{R}$ *as in* (3.2.15), $\underline{V}_{*,H}(\hat{x}_0, \bar{q}_0)$ *is as in* (3.3.22) *and* $\bar{q}_0 = \tau(q_0, L \circ h(x_0))$.

Finally, the controller synthesis procedure is summarized as follows:

- For the problem of robust satisfaction, one first synthesize a Markov policy ρ^* as in (3.3.7). Then, the controller $\tilde{\mathbf{C}}_{\rho^*}$ as in Definition 3.3.1 is constructed based on ρ^*.
- As for the problem of worst-case violation, one constructs a control strategy $\tilde{\mathbf{C}}_{\rho_*}$ as in Definition 3.3.1 based on a Markov policy ρ_* synthesized as in (3.3.20).

Remark 3.3.13. *Note that given the product gDTSG* $\widehat{\mathfrak{D}} \otimes \mathcal{A} = \{\bar{X}, \bar{U}, \bar{W}, \bar{X}_0, \bar{T}, \bar{Y}, \bar{h}\}$, *both* ρ^* *as in* (3.3.7) *and* ρ_* *as in* (3.3.20) *are (offline) look-up tables, whose sizes grow linearly with the time horizon* H *and the cardinality of* $\bar{X}$. *Moreover, the number of operations required for computing* (3.3.7) *and* (3.3.20) *is proportional to* H *and the cardinality of* $\bar{X}$, $\bar{U}$, *and* $\bar{W}$. *It is also worth noting that, for all* $n \in [0, H-1]$, *the computations of* $\rho_{*n}(\hat{x}, q)$ *and* $\rho_n^*(\hat{x}, q)$ *for all* $(\hat{x}, q) \in \bar{X}$ *are independent from each other and can be done in a parallel fashion.*

3.4 Case Studies

In this section, the proposed approaches in this chapter is applied to two case studies, including the running example and a control problem for a Quadrotor helicopter. Each case study is simulated with 1.0×10^5 different realizations of noise, in which inputs of Player II are randomly selected from their input sets following a uniform distribution. Here, Player II does not select adversarial inputs rationally since it is challenging to obtain closed-form solutions for such case. Meanwhile, the probabilistic guarantees provided by the results in this chapter are still valid regardless of how Player II chooses inputs. To show the applicability of the proposed results, in all case studies, I summarize the required memory[1] for storing stochastic kernels and synthesized controllers, and report the average execution time of these controllers. All experiments are performed via `MATLAB 2019b`, on a machine with Ubuntu 20.04 (Intel(R) Xeon(R) Gold 6254 CPU (3.1 GHz) and 378 GB of RAM).

[1]In this section, 4 bytes are allocated for each entry of matrices to be stored as a single-precision floating-point.

3.4.1 Running Example

Here, the controller of the running example is synthesized following (3.3.20)-(3.3.24). The simulation setting and results are summarized in Table 3.1 and depicted in Figure 3.3. One can readily observe that the probabilistic guarantee of satisfaction is respected. Additionally, I also show how the reduced-order game improves the scalability issue (cf. Remark 3.2.3) via the running example. To do so, the finite abstraction of the original game without performing any model order reduction is built by considering $[-12, 12]^3$ as the region of interest. Then, this region is uniformly partitioned with girds whose sizes are $(0.24, 0.24, 0.24)$ for a fair comparison with the reduced-order model setting (cf. Table 3.2). For the same reason, U and W are uniformly divided with grids whose sizes are $(0.06, 0.06)$ and 0.1, respectively. Under this setting, when a reduced-order game is built, only around 19 MB is required to store the stochastic kernel. On the other hand, without constructing a reduced-order game, the finite abstraction contains 10^6 states, 1.25×10^5 inputs for Player I, and 10 inputs for Player II. As a result, one needs 4.65×10^9 GB to store the stochastic kernel, which is not practical.

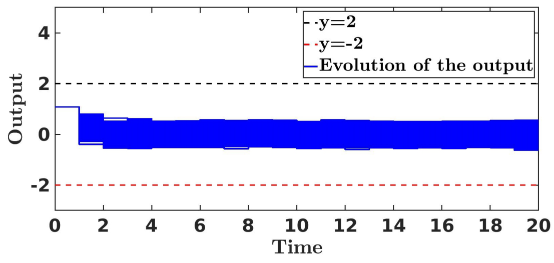


Figure 3.3: Simulation of the running example with respect to ψ.

3.4.2 Quadrotor

Here, the proposed results are applied to a quadrotor tracking a moving vehicle on the ground. Consider a quadrotor moving on a 2-dimensional planar. As discussed in [93], the control of a quadrotor can be decoupled into the control on different dimensions. Hence, I borrow the model from [93] which models the relative motion between the quadrotor and the ground vehicle:

$$\mathfrak{D}: \begin{cases} x(k+1) = Ax(k) + Bu(k) + Dw(k) + R\varsigma(k), \\ y(k) = Cx(k), \qquad\qquad\qquad k \in \mathbb{N}, \end{cases}$$

where $A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$, $R = \begin{bmatrix} 0.4\Delta t & 0 \\ 0 & 0.4\Delta t \end{bmatrix}$, $B = [\frac{\Delta t^2 g}{2}; \Delta t g]$, $D = [\frac{\Delta t^2}{2}; \Delta t]$, and $C = [1; 0]^\top$, with $\Delta t = 0.05s$ being the sampling time and $g = 9.8m/s$ being the gravitational constant. Here, $x(k) = [x_1(k); x_2(k)]$ with $x_1(k)$ and $x_2(k)$ being the relative position and velocity between the quadrotor and the vehicle, respectively; $u(k) \in [-0.25, 0.25](m/s^2)$ denotes the acceleration of the quadrotor as the control input; $w(k) \in [-0.6, 0.6](m/s^2)$ denotes the acceleration that can be chosen by the vehicle in the worst case against the control inputs; $\varsigma(k)$ is a standard Gaussian random variable; and y is the output of the system. Here, the following properties would be considered:

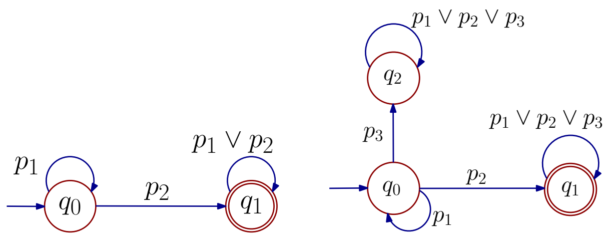


Figure 3.4: Left: DFA for modeling ψ_1, with accepting state q_1, alphabet $\Pi = \{p_1, p_2\}$, and labeling function $L : Y \to \Pi$ with $L(y) = p_1$ when $y \in [-0.7, 0.7]$, and $L(y) = p_2$ when $y \in (-\infty, -0.7) \cup (0.7, +\infty)$. **Right:** DFA for modeling ψ_2, with accepting state q_1, alphabet $\Pi = \{p_1, p_2, p_3\}$, and labeling function $L : Y \to \Pi$ with $L(y) = p_1$ when $y \in [-1, -0.5) \cup (0.5, 1]$, $L(y) = p_2$ when $y \in [-0.5, 0.5]$, and $L(y) = p_3$ when $y \in (-\infty, -1) \cup (1, +\infty)$.

1. ψ_1: y should stay in $[-0.7, 0.7]$ for 1 minute (*i.e.*, time horizon $H = 1200$). The DFA for modeling ψ_1 is shown in Figure 3.4 (left), and the problem of worst-case violation concerning this DFA is of interest.

2. ψ_2: starting from $[-1.0, 1.0]$, y should reach $[-0.5, 0.5]$ within 5 seconds (*i.e.*, time horizon $H = 100$). Here, a DFA as in Figure 3.4 (right) is constructed for characterizing ψ_2. Accordingly, the problem of robust satisfaction regarding this DFA should be considered.

3. ψ_3: within 2 seconds (*i.e.*, time horizon $H{=}40$), (1) y should reach $[-0.45, 0.45]$ and then stay within $[-0.45, 0.45]$ for 3 time instants after it reaches $[-0.45, 0.45]$; (2) if it reaches $[-0.1, 0.1]$, it only needs to stay within $[-0.45, 0.45]$ for 1 time instant after it reaches $[-0.1, 0.1]$; (3) y is not allowed to leave $[-0.8, 0.8]$. The DFA for modeling ψ_3 is depicted in Figure 3.5 and I focus on the problem of robust satisfaction accordingly.

First, the finite abstraction of the model is constructed. Since model order reduction is not applied to this model, one can select $P = \mathbf{I}_2$. Therefore, one gets $\hat{A}_{\mathsf{r}} = A$, $\hat{D}_{\mathsf{r}} = D$, $\hat{R}_{\mathsf{r}} = R$, $\hat{C}_{\mathsf{r}} = C$ and $Q = S = \mathbf{0}_{2\times 1}$. The finite abstraction is constructed as in Table 3.2. Accordingly, one can select $\tilde{M} = 1$ and $\tilde{\epsilon} = 0.05$. As for establishing the relation between the constructed abstraction and the original game, I set $\hat{U}' = \{\hat{u} \in \hat{U} \mid -0.12 \leq \hat{u} \leq 0.12\}$, $\delta = 0$, and the tolerable range of ϵ as $[0.05, 0.4]$. By applying Algorithm 1, the finite abstraction is (ϵ, δ)-stochastically simulated by the original model with $\delta = 0$, $M = \begin{bmatrix} 1.7699 & 0.5494 \\ 0.5494 & 0.3920 \end{bmatrix}$, and $\epsilon = 0.2911$, when the interface function in (3.2.27) is applied with $\tilde{R} = 1$ and $K = [-0.4294; -0.2773]$. Now I am ready to synthesize a controller enforcing ψ_1 following (3.3.20)-(3.3.24), and controllers enforcing ψ_2 and ψ_3 following (3.3.7)-(3.3.11). The setting and results of the simulation

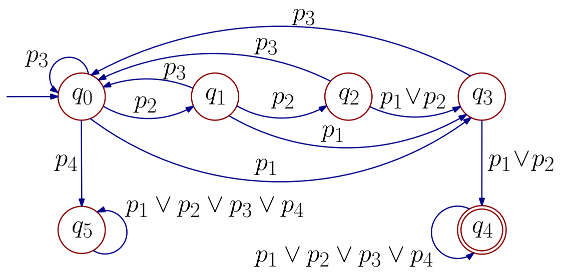


Figure 3.5: DFA for modeling ψ_3 with accepting state q_4, alphabet $\Pi = \{p_1, p_2, p_3, p_4\}$, and labeling function $L : Y \to \Pi$ with $L(y)=p_1$ when $y \in [-0.1, 0.1]$; $L(y) = p_2$ when $y \in [-0.45, -0.1) \cup (0.1, 0.45]$; $L(y) = p_3$ when $y \in [-0.8, -0.45) \cup (0.45, 0.8]$, and $L(y) = p_4$ when $y \in (-\infty, -0.8) \cup (0.8, +\infty)$. Transitions $q_5 = \tau(q_j, p_4)$, with $j \in \{1, 2, 3\}$, are omitted to keep the figure less crowded.

	x_0	P_f	P_e	Execution time (ms)
ψ	$[3.8; 4.1; 2.9]$	$\geq 99.90\%$	100%	0.0755
ψ_1	$[0.2; 0.2]$	$\geq 99.26\%$	100%	0.0684
ψ_2	$[0.6; 0.1]$	$\geq 94.77\%$	100%	0.0683
ψ_3	$[-0.48; 0.45]$	$\geq 98.75\%$	100%	0.0766

Table 3.1: Simulation results with respect to properties ψ, ψ_1, ψ_2, ψ_3, with P_f denoting the formal probabilistic guarantees, and P_e being the empirical satisfaction probability.

for ψ_1, ψ_2, and ψ_3 are summarized in Table 3.1 and depicted in Figure 3.6. In all case studies, the probabilistic guarantees of satisfaction are well respected.

3.4.3 Comparison with Existing Results

By virtue of the grid-based approximation framework introduced in [1], results in [93, 52] can be applied to the synthesis problem for (nonlinear) stochastic games with continuous state and input sets. In this subsection, the methodologies in this chapter are compared with these results in the sense of the conservativeness of probabilistic guarantees associated with the synthesized controllers. Note that providing less conservative probabilistic guarantees are crucial in correct-by-construction synthesis techniques. The ultimate goal for employing these techniques is to obtain formal (probabilistic) guarantees for satisfying the desired properties, instead of performing exhaustive testing, which is heuristic, costly, and time-consuming.

Under the grid-based approximation framework in [1], the probabilistic guarantee for a desired property is provided in terms of a *probabilistic closeness*, denoted by $\mathbf{e}$, between the finite abstraction and the original system, with:

$$|p - \hat{p}| \leq \mathbf{e}, \tag{3.4.1}$$

where $\hat{p}$ and p denote the probabilities of satisfaction over the finite abstraction and the original system, respectively. Moreover, [175] shows that $\mathbf{e}$ is proportional to the size of

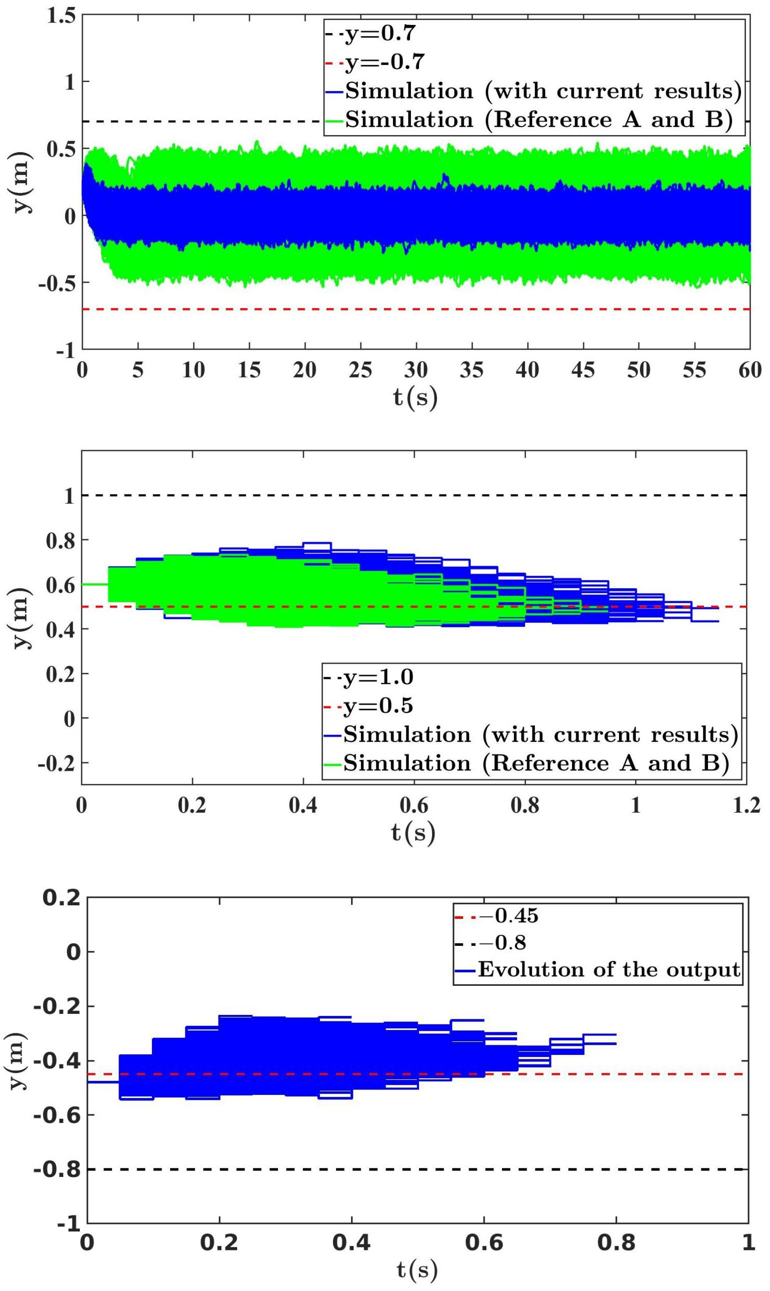


Figure 3.6: Simulation results for ψ_1 (top), ψ_2 (middle), and ψ_3 (bottom). (Reference A: [93]; Reference B: [52])

	X_{rs}	Grids' size			Number of Elements				Time	Required Memory (GB)	
		X_{rs}	U	W	$\hat{X}$	$\hat{U}$	$\hat{W}$	Q	horizon	$\hat{T}$	ρ^* (or ρ_*)
ψ	$[-12, 12]$	0.24	0.06	0.1	101	50	10	4	20	1.9×10^{-2}	3.01×10^{-5}
ψ_1	$[-0.7, 0.7] \times [-0.5, 0.5]$				3501			2	1200	7.13	3.13×10^{-2}
ψ_2	$[-1, 1] \times [-0.75, 0.75]$	(0.02, 0.02)	0.02	0.1	7501	25	12	3	100	32.70	8.38×10^{-3}
ψ_3	$[-0.8, 0.8] \times [-0.55, 0.55]$				4401			6	40	11.26	3.93×10^{-3}

Table 3.2: Construction of finite abstractions against properties ψ, ψ_1, ψ_2, ψ_3, where X_{rs} and $\hat{X}$ denote the region of interest (see Section 3.2.1 for its definition) and its corresponding finite state set, respectively; $\hat{T}$ denotes the stochastic kernel of the finite abstraction; ρ^* (or ρ_*) denotes the look-up tables for constructing the desired controllers (cf. Remark 3.3.13); U (resp. W) denotes the input set of Player I (resp. Player II) to be partitioned, and $\hat{U}$ (resp. $\hat{W}$) denotes its corresponding finite sets.

discretization parameters, denoted by η_i, $i = \{1, \dots, s\}$, with s being the dimension of the state set. Roughly speaking, the quantity η_i is the maximum diameter of partition cells along with the i^{th} dimension of the state set. Here, the interested readers are referred to [175, Theorem 9] for the formal definition. By employing the results in [175, Section 5], one has $\mathbf{e} = 3.586 \times 10^4$, $\mathbf{e} = 4.356 \times 10^3$, and $\mathbf{e} = 1.345 \times 10^3$ for ψ_1, ψ_2, and ψ_3[2], respectively, when grid-size parameters are $(\eta_1, \eta_2) = (0.02, 0.02)$ (as the discretization setting in Table 3.2).

In all cases, $\mathbf{e}$ is significantly larger than 1. Notably, the results in [175, Section 5] only consider the effect of state set's discretization on $\mathbf{e}$. According to results in [174, 184], the discretization of input sets would make $\mathbf{e}$ even larger. Since probability should be a real number between 0 and 1, the probabilistic guarantees for the original system are very conservative in all cases. To show this, I first synthesize controllers with the results in [93, 52] enforcing ψ_1 and ψ_2[3]. By deploying these controllers, one gets formally that the probabilities of satisfying ψ_1 and ψ_2 will be within $[-3.586 \times 10^4, 3.586 \times 10^4]$ and $[-4.356 \times 10^3, 4.356 \times 10^3]$, respectively, which are very conservative. Then, starting from the same initial states as in Table 3.1, both cases are simulated with 10^5 different noise realizations. In both cases, as depicted in Figure 3.6, trajectories under different noise realizations satisfy the desired properties with probability 1 in the experiments. Hence, the formal probabilistic guarantees associated with both controllers are very conservative considering the empirical results. In comparison, as shown in Table 3.1, my controllers empirically perform as good as those controllers synthesized with the results in [93, 52]. On the other hand, the results in this chapter provide formal probabilistic guarantees which are much less conservative. Note that one may select smaller η_i such that $\mathbf{e}$ becomes smaller. Here, I summarize in Table 3.3 the required (η_1, η_2) and the corresponding memory for storing the stochastic kernels of finite abstractions such that one has reasonable $\mathbf{e}$. In terms of required memory, it is computationally expensive to provide a reasonable guarantee under the grid-based approximation framework proposed in [1].

Properties	Required (η_1, η_2) $(\times 10^{-6})$	Required memory (GB)
ψ_1	$(0.492, 0.644)$	2.184×10^{19}
ψ_2	$(4.132, 5.166)$	2.207×10^{16}
ψ_3	$(12.915, 17.512)$	6.768×10^{13}

Table 3.3: Required (η_1, η_2) and corresponding required memory for different properties when applying the results in [93, 52].

[2]Although results in [93, 52] only solve the reachability problem over continuous sets, enforcing DFA properties can be cast as a reachability problem over state set of the product system between the DFA and the original system. Therefore, results in [175, Section 5] can readily be used to compute $\mathbf{e}$ for ψ_3.

[3]The results in [93, 52] cannot be used to synthesize controllers enforcing ψ_3 since they do not provide any operator that handle general DFA properties like ψ_3.

Next, I compare the proposed results with operators in [74] by showing Corollary 3.3.9 with an example. To this end, the following system is considered:

$$\mathfrak{D}: \begin{cases} x(k+1) = Ax(k) + Bu(k) + R\varsigma(k), \\ y(k) = Cx(k), \qquad\qquad k \in \mathbb{N}, \end{cases}$$

where

$$A = \begin{bmatrix} 0.91 & 0.47 & 0.76 \\ 0.65 & 0.71 & 0.93 \\ 0.69 & 0.28 & 0.53 \end{bmatrix}, \quad B = \begin{bmatrix} 9.5 & 0.6 & 4.1 \\ 2.4 & 12.4 & 2.9 \\ 5.7 & 5.4 & 5.8 \end{bmatrix},$$

$R = [0.63; 0.28; 0.48]$, and $C = [0.1; 0.1; 0.1]^\top$, in which $x(t) = [x_1(k); x_2(k); x_3(k)]$ and $u(k) \in [-3, 3]^3$. Here, a co-safe linear temporal logic property [63] ψ' is considered, which can be handled by the operators proposed in [74]: starting from $[-2, 2]$, the output of the system should reach $[-0.3, 0.3]$ while avoiding $(-\infty, -2) \cup (2, +\infty)$ within 90 time steps (i.e., $H = 90$). Accordingly, the controller is synthesized by solving the problem of robust satisfaction corresponding to the DFA in Figure 3.7.

For constructing the finite abstraction, $P = [0.5; 0.4; 0.5]$ is chosen and a reduced-order model is constructed accordingly with $\hat{A}_\mathsf{r} = 0.62$, $\hat{B}_\mathsf{r} = 1$, $\hat{C}_\mathsf{r} = 0.14$, $\hat{R}_\mathsf{r} = 0.9939$, and $Q = [-0.1179; -0.0694; 0.1094]$ as proposed in (3.2.5)-(3.2.8). The finite abstraction is then constructed by uniformly dividing the region of interest, i.e. $[-15, 15]$, of the reduced-order model's state set into partitions whose lengths are 0.15, and partitioning the input set, i.e. $[-3, 3]$, for the reduced-order model uniformly with 48 cells. Here, $\hat{U}' = \hat{U}$ is set, and the finite abstraction is (ϵ, δ)-stochastically simulated by original model with $\delta = 0.1$, $\epsilon = 0.2466$, with the associated $\nu(x, \hat{x}, \hat{u}) := K(x - P\hat{x}) + Q\hat{x} + \tilde{R}\hat{u}$ with $\tilde{R} = [0.0369; 0.0172; 0.0340]$,

$$M = \begin{bmatrix} 0.0107 & 0.0106 & 0.0108 \\ 0.0106 & 0.0105 & 0.0106 \\ 0.0108 & 0.0106 & 0.0108 \end{bmatrix}, K = \begin{bmatrix} -0.3225 & -0.1899 & -0.3033 \\ 0.2199 & 0.2355 & 0.2094 \\ -0.0441 & -0.1894 & -0.0532 \end{bmatrix}.$$

Then, I synthesize controllers with the operator in (3.3.10) and the one proposed in [74, equation (42)]. As an example, Figure 3.8 demonstrates the lower bounds for the probability of satisfaction associated with both controllers when the original system's initial state is $x = [5; 5; \tilde{x}]$ where $\tilde{x} \in [-5, 5]$ (correspondingly, original system's output $y \in [0.5, 1.5]$). One can readily observe that operator proposed in this chapter provides a less conservative lower bound than the one proposed in [74].

3.5 Summary

In this chapter, a notion of (ϵ,δ)-approximate probabilistic relations is considered to quantify the similarity between two stochastic games. Based on this notion, new Bellman operators are proposed to synthesize controllers for stochastic games enforcing complex logical properties modeled by deterministic finite automata. To do so, a

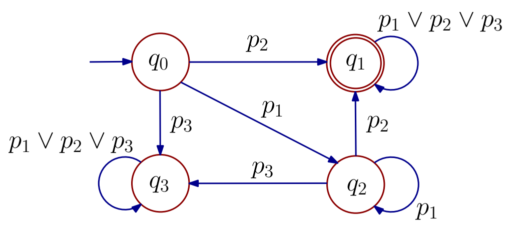


Figure 3.7: DFA for modelling ψ', with accepting state q_1, alphabet $\Pi = \{p_1, p_2, p_3\}$, and labeling function $L : Y \to \Pi$ with $L(y) = p_1$ when $y \in [-2, -0.3) \cup (0.3, 2]$, $L(y) = p_2$ when $y \in [-0.3, 0.3]$, and $L(y) = p_3$ when $y \in (-\infty, -2) \cup (2, +\infty)$.

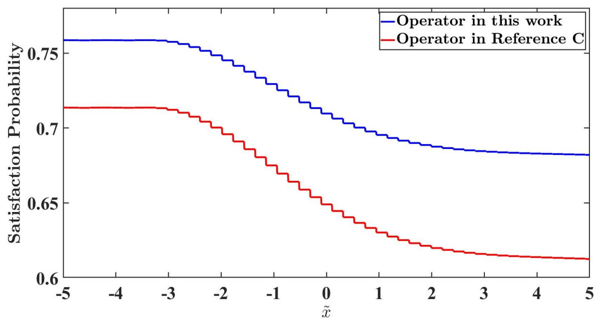


Figure 3.8: Comparison of probabilistic guarantees between the operator in (3.3.23) and the one proposed in [74] (Reference C).

controller is first synthesized based on a finite abstraction that is (ϵ, δ)-stochastically simulated by the original game. Then, this controller is refined to the original game based on the approximate probabilistic relation between the original game and its finite abstraction, which is the key to providing probabilistic guarantees. Moreover, a systematic algorithm is proposed to establish such a relation for a particular class of nonlinear stochastic games with slope restrictions on the nonlinearity. The empirical results show that the newly proposed method is less conservative than the existing methods in the literature.

3.6 Proof of Statements in Chapter 3

3.6.1 Proof of Statements: Section 3.2

The following proposition is required to show the results of Section 3.2.

Proposition 3.6.1. *Consider a positive (semi)definite matrix $M_0 \in \mathbb{R}^{n\times n}$. Given $a, b \in \mathbb{R}$ with $a \leq b$, and a matrix $M \in \mathbb{R}^{n\times n}$, if $M_0 + aM$ and $M_0 + bM$ are positive (semi)definite, then for all $t \in [a, b]$, $M_0 + tM$ is positive (semi)definite.*

Proof: For any $t \in [a, b]$,

- If $a \leq t < 0$, one has

$$M_0 + tM = (1 - \frac{t}{a})M_0 + \frac{t}{a}M_0 + tM = (1 - \frac{t}{a})M_0 + \frac{t}{a}(M_0 + aM).$$

Since $1 - \frac{t}{a} \geq 0$ and $\frac{t}{a} \geq 0$, both $(1 - \frac{t}{a})M_0$ and $\frac{t}{a}(M_0 + aM)$ are positive (semi)definite, so that $M_0 + tM$ is also positive (semi)definite.

- If $0 < t \leq b$, one has

$$M_0 + tM = (1 - \frac{t}{b})M_0 + \frac{t}{b}M_0 + tM = (1 - \frac{t}{b})M_0 + \frac{t}{b}(M_0 + bM).$$

Since $1 - \frac{t}{b} \geq 0$ and $\frac{t}{b} \geq 0$, both $(1 - \frac{t}{b})M_0$ and $\frac{t}{b}(M_0 + bM)$ are positive (semi)definite, so that $M_0 + tM$ is also positive (semi)definite.

Additionally, $M_0 + tM$ is positive (semi)definite when $t = 0$, which completes the proof. ■

Now I am ready to show the results of Section 3.2.

Proof of Theorem 3.2.5: Since $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$ are affected by the same additive noise $\varsigma \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$, one can readily define an lifting $\mathscr{L}_T$ based on $\varsigma \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$ for the approximation probabilistic relation. Now, one needs to check the conditions in Definition 2.4.8. Note that the third condition in Definition 2.4.8 holds trivially since only those initial states $x_0 \in X_0$ and $\hat{x}_0 \in \hat{X}_0$ such that $(x_0, \hat{x}_0) \in \mathscr{R}$ are considered. Firstly, with (3.2.5) and (3.2.17), one has

$$\|y - \hat{y}\|^2 = \|Cx - \hat{C}_{\mathsf{r}}\hat{x}\|^2 = (x - P\hat{x})^\top C^\top C(x - P\hat{x}) \leq (x - P\hat{x})^\top M(x - P\hat{x}) \leq \epsilon^2,$$

for any $(x, \hat{x}) \in \mathscr{R}$. Therefore, the first condition holds for all $(x, \hat{x}) \in \mathscr{R}$. The second condition requires that $\forall (x, \hat{x}) \in \mathscr{R}$, $\forall \hat{u} \in \hat{U}$, $\exists u \in U$ s.t. $\forall w \in W$, $\exists \hat{w} \in \hat{W}$ with $(w, \hat{w}) \in \mathscr{R}_w$ s.t. the next state $(x', \hat{x}')$ is also in the relation $\mathscr{R}$ with a probability of at least $1 - \delta$. According to Assumption 3.3.2, the following should hold:

$$\mathbb{P}\{(x' - P\hat{x}')^\top M(x' - P\hat{x}') \leq \epsilon^2\} \geq 1 - \delta. \tag{3.6.1}$$

From the slope restriction of φ as in (3.2.2), one gets

$$\varphi(Fx) - \varphi(FP\hat{x}) = b(Fx - FP\hat{x}) = bF(x - P\hat{x}), \tag{3.6.2}$$

with $b \in [\underline{b}, \bar{b}]$ if $x \neq P\hat{x}$, and $b = 0$ otherwise. Then, by applying the dynamics of $\mathfrak{D}$ as in (3.2.1) and $\widehat{\mathfrak{D}}$ as in (3.2.13), one has

$$\begin{aligned} x' - P\hat{x}' = \; & Ax + E\varphi(Fx) + Dw + B\nu(x, \hat{x}, \hat{u}) + R\varsigma \\ & - P(\hat{A}_{\mathsf{r}}\hat{x} + \hat{E}_{\mathsf{r}}\varphi(\hat{F}_{\mathsf{r}}\hat{x}) + \hat{D}_{\mathsf{r}}\hat{w} + \hat{B}_{\mathsf{r}}\hat{u} + \hat{R}_{\mathsf{r}}\varsigma) + P\beta. \end{aligned} \tag{3.6.3}$$

Additionally, one can simplify (3.6.3) to

$$\begin{aligned} & \big(A + BK + b(BL + EF)\big)(x - P\hat{x}) + (B\tilde{R} - P\hat{B}_{\mathsf{r}})\hat{u} \\ & \qquad + D(w - \hat{w}) + (R - P\hat{R}_{\mathsf{r}})\varsigma + P\beta - BS\hat{w}, \end{aligned}$$

by employing (3.2.6)-(3.2.9), (3.6.2) and (3.2.27). Note that b is used here to denote $b(x,\hat{x})$ as in (3.2.27) for succinctness, and it is clear from the context. Therefore, (3.6.1) is fulfilled when

$$\begin{aligned}\|\big(A+BK+b(BL+EF)\big)(x-P\hat{x})+(B\tilde{R}-P\hat{B}_{\mathsf{r}})\hat{u}&\\ +D(w-\hat{w})+(R-P\hat{R}_{\mathsf{r}})\varsigma+P\beta-BS\hat{w}\|_M\leq\epsilon&\end{aligned}\quad(3.6.4)$$

holds for all $b\in[\underline{b},\bar{b}]\cup\{0\}$, for all $\beta\in\Delta$ as in (3.2.11), and for all ς s.t. $\mathbb{P}\{\varsigma^\top\varsigma\leq c_\varsigma^2\}\geq 1-\delta$ with $c_\varsigma=\chi_d^{-1}(1-\delta)$, since $\varsigma\sim(\mathbf{0}_d,\mathbf{I}_d)$ so that $\varsigma^\top\varsigma$ has chi-square distribution with d degrees of freedom. Considering the left-hand side of (3.6.4), one has

$$\begin{aligned}&\|\big(A+BK+b(BL+EF)\big)(x-P\hat{x})+(B\tilde{R}-P\hat{B}_{\mathsf{r}})\hat{u}+D(w-\hat{w})+(R-P\hat{R}_{\mathsf{r}})\varsigma+P\beta-BS\hat{w}\|_M\\
&\leq\|\big(A+BK+b(BL+EF)\big)(x-P\hat{x})\|_M+\|D(w-\hat{w})\|_M+\|(B\tilde{R}-P\hat{B}_{\mathsf{r}})\hat{u}\|_M\\
&\quad+\|(R-P\hat{R}_{\mathsf{r}})\varsigma\|_M+\|P\beta\|_M+\|BS\hat{w}\|_M\\
&\leq\|\big(A+BK+b(BL+EF)\big)(x-P\hat{x})\|_M+\gamma_0+\gamma_1+\gamma_2+\gamma_3+\gamma_4\\
&=\|\big(A+BK+b(BL+EF)\big)(x-P\hat{x})\|_M+\tilde{\gamma},\end{aligned}\quad(3.6.5)$$

with $\tilde{\gamma}$ as in (3.2.21), γ_0 as in (3.2.22), γ_1 as in (3.2.23), γ_2 as in (3.2.24), γ_3 as in (3.2.25), and γ_4 as in (3.2.26). According to S-procedure [35], for all $\|x-P\hat{x}\|_M\leq\epsilon$, $\|\big(A+BK+b(BL+EF)\big)(x-P\hat{x})\|_M+\tilde{\gamma}\leq\epsilon$ holds for all $b\in[\underline{b},\bar{b}]\cup\{0\}$ if and only if there exists a $\kappa\geq 0$ such that

$$\begin{bmatrix}A_b^\top MA_b & \mathbf{0}_n\\ \mathbf{0}_n^\top & -(\epsilon-\tilde{\gamma})^2\end{bmatrix}\preceq\kappa\begin{bmatrix}M & \mathbf{0}_n\\ \mathbf{0}_n^\top & -\epsilon^2\end{bmatrix}\quad(3.6.6)$$

holds for all $b\in[\underline{b},\bar{b}]\cup\{0\}$, with $A_b=A+BK+b(BL+EF)$. Note that (3.6.6) holds if and only if $\kappa M-A_b^\top MA_b$ is positive semidefinite and $-\kappa\epsilon^2+(\epsilon-\tilde{\gamma})^2\geq 0$. Therefore, one has (3.6.6) holds for all $b\in[\underline{b},\bar{b}]\cup\{0\}$ if and only if $\forall b\in[\underline{b},\bar{b}]\cup\{0\}$, there exists a $\kappa\in[0,(\epsilon-\tilde{\gamma})^2/\epsilon^2]$ such that

$$A_b^\top MA_b\preceq\kappa M.\quad(3.6.7)$$

Using Schur complement [35], one can rewrite (3.6.7) as

$$\underbrace{\begin{bmatrix}\bar{M} & A\bar{M}+B\bar{K}\\ \bar{M}^\top A^\top+\bar{K}^\top B^\top & \kappa\bar{M}\end{bmatrix}}_{M_0}+b\underbrace{\begin{bmatrix}\mathbf{0}_{n\times n} & B\bar{L}+EF\bar{M}\\ \bar{L}^\top B^\top+\bar{M}^\top F^\top E^\top & \mathbf{0}_{n\times n}\end{bmatrix}}_{M'}\succeq 0,$$

with $\bar{M}=M^{-1}$, $\bar{K}=K\bar{M}$, and $\bar{L}=L\bar{M}$. According to (3.2.18), M_0 is positive semidefinite. Furthermore, (3.2.19), (3.2.20), and (3.2.21) ensure that there exists a κ with $0\leq\kappa\leq(\epsilon-\tilde{\gamma})^2/\epsilon^2$ such that (3.6.7) holds for $b=\{\bar{b},\underline{b}\}$. As a result, according to Proposition 3.6.1, there exists a $\kappa\in[0,(\epsilon-\tilde{\gamma})^2/\epsilon^2]$ such that M_0+bM' is positive semidefinite for all $b\in[\underline{b},\bar{b}]\cup\{0\}$. Therefore, the second condition also holds, which completes the proof. ■

Next, I show the results of Corollary 3.2.7.

Proof of Corollary 3.2.7 According to [150, Theorems 5,6], for all $b' \in \{\underline{b}, \bar{b}, 0\}$, the pair $(A + b'EF, B)$ is stabilizable if and only if there exist positive-definite matrix M and $\kappa \in [0, 1]$ such that (3.2.18)-(3.2.20) hold. Next, I show that if there exist M' and κ' such that (3.2.18)-(3.2.20) hold, then:

- (**C1**) There exist M and κ such that (3.2.17)-(3.2.20);
- (**C2**) There exist $\tilde{\gamma}$ and ϵ so that (3.2.21) holds.

Let's start by showing (**C1**). Suppose one has M' and κ' such that $(A+BK)^\top M'(A+BK) \preceq \kappa' M'$, $\bar{A}^\top M' \bar{A} \preceq \kappa' M'$, and $\underline{A}^\top M' \underline{A} \preceq \kappa' M'$ (*i.e.*, (3.2.18)-(3.2.20) hold). Then, for any C as in (3.2.17), there exists $m' \in \mathbb{R}_{>0}$ such that $m'M' \succeq C^\top C$, since M' is positive definite. Meanwhile, one can readily verify that (3.2.18)-(3.2.20) still hold with $M = m'M'$ and $\kappa = \kappa'$. Therefore, (**C1**) holds. As for (**C2**), suppose one has M and κ such that (3.2.18)-(3.2.20) hold, one can verify that (3.2.18)-(3.2.20) also hold with M and any κ'' such that

$$\sqrt{\kappa''} \geq \max_{b \in [\underline{b}, \bar{b}] \cup \{0\}} \|N A_b N^{-1}\|, \tag{3.6.8}$$

where $A_b = A + BK + b(BL + EF)$, and $N \in \mathbb{R}^{n \times n}$ is a positive-definite matrix such that $N^\top N = M$. Thus, if one has $1 - \tilde{\gamma}/\epsilon \in [\max_{b \in [\underline{b}, \bar{b}] \cup \{0\}} \|N A_b N^{-1}\|, 1]$, then (3.2.21) holds. In fact, for any $\epsilon \in \mathbb{R}_{>0}$, one has $\tilde{\gamma}$ such that $1 - \tilde{\gamma}/\epsilon \in [\max_{b \in [\underline{b}, \bar{b}] \cup \{0\}} \|N A_b N^{-1}\|, 1]$, when the finite abstraction is properly constructed. On one hand, one always has $\gamma_1 = \gamma_2 = \gamma_4 = 0$ when there is no model order reduction involving in the abstraction since, in this case, one has $P = \mathbf{I}_n$. Then, one can select $\hat{B}_{\mathsf{r}} = B$ and $\tilde{R} = \mathbf{I}_m$ in (3.2.23), $R = \hat{R}_{\mathsf{r}}$ in (3.2.24), and $S = \mathbf{0}_{m \times p}$ in (3.2.26), so that one has $B\tilde{R} - P\hat{B}_{\mathsf{r}} = \mathbf{0}_{n \times m}$, $R - P\hat{R}_{\mathsf{r}} = \mathbf{0}_{n \times d}$ and $BS = \mathbf{0}_{n \times p}$. On the other hand, γ_0 and γ_3 are proportional to the cardinality of Δ_w in (3.2.12) and Δ in (3.2.11), respectively. Therefore, one has (**C2**) also holds, which completes the proof. ■

To show the results of Theorem 3.2.8, the following proposition is required.

Proposition 3.6.2. *Consider a constraint $c\bar{x} \leq 1$ with $\bar{x} \in \mathbb{R}^n$ and a set $E_{\bar{x}} = \{\bar{x} \mid \bar{x}^\top M \bar{x} \leq \epsilon^2\}$ with $M \in \mathbb{R}^{n \times n}$ and $\epsilon \in \mathbb{R}_{>0}$. Then, $c\bar{x} \leq 1$ holds for all $\bar{x} \in E_{\bar{x}}$ if and only if $cM^{-1}c^\top \leq 1/\epsilon^2$.*

Proof: For all $\bar{x} \in E_{\bar{x}}$, $c\bar{x} \leq 1$ if and only if $\max_{\bar{x} \in E_{\bar{x}}} c\bar{x} \leq 1$. Let $\bar{x}^* = \arg\max_{\bar{x} \in E_{\bar{x}}} c\bar{x}$. Then, $\bar{x}^*$ satisfies the Karush-Kuhn-Tucker conditions [35]:

$$\lambda(\epsilon^2 - \bar{x}^{*\top} M \bar{x}^*) = 0, \tag{3.6.9}$$

$$c - 2\lambda M \bar{x}^* = 0, \tag{3.6.10}$$

with $\lambda \geq 0$. Solving (3.6.9) and (3.6.10), one has

$$\bar{x}^* = \epsilon \frac{M^{-1}c^\top}{\sqrt{cM^{-1}c^\top}}.$$

Therefore, $\max_{\bar{x}\in E_{\bar{x}}} c\bar{x} = c\bar{x}^* = \epsilon\sqrt{cM^{-1}c^\top} \leq 1$ if and only if $cM^{-1}c^\top \leq 1/\epsilon^2$, which concludes the proof. ■

Now I am ready to show the results of Theorem 3.2.8.

Proof of Theorem 3.2.8: For any $i \in \{1,\ldots,r\}$, $\alpha_i\bar{u} \leq 1$ implies that $\alpha_i(K + bL)\bar{x} \leq 1$ for all $b \in [\underline{b},\bar{b}]\cup\{0\}$. According to Proposition 3.6.2, $\forall \bar{x} \in E_{\bar{x}}$, $\alpha_i(K+bL)\bar{x} \leq 1$ is fulfilled for all $b \in [\underline{b},\bar{b}] \cup \{0\}$ if and only if

$$\alpha_i(K + bL)M^{-1}(K + bL)^\top\alpha_i^\top \leq 1/\epsilon^2 \tag{3.6.11}$$

holds for all $b \in [\underline{b},\bar{b}] \cup \{0\}$. Using Schur complement [35], (3.6.11) can be rewritten as

$$\underbrace{\begin{bmatrix} 1/\epsilon^2 & \alpha_i\bar{K} \\ \bar{K}^\top\alpha_i^\top & \bar{M} \end{bmatrix}}_{M_0} + b\underbrace{\begin{bmatrix} 0 & \alpha_i\bar{L} \\ \bar{L}^\top\alpha_i^\top & 0 \end{bmatrix}}_{M'} \succeq 0, \tag{3.6.12}$$

with $\bar{M} = M^{-1}$, $\bar{K} = K\bar{M}$, and $\bar{L} = L\bar{M}$. Note that M_0 is positive semidefinite according to (3.2.32). Moreover, (3.2.33) and (3.2.34) ensure that $M_0 + \underline{b}M'$ and $M_0 + \bar{b}M'$ are both positive semidefinite. Then, according to Proposition 3.6.1, (3.2.32) to (3.2.34) guarantee that (3.6.12) holds for all $b \in [\underline{b},\bar{b}] \cup \{0\}$, which completes the proof. ■

3.6.2 Proof of Statements: Section 3.3

To show the results of Section 3.3, some additional definitions and lemmas are required for the product gDTSG $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ between the original gDTSG $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ and its finite abstraction $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ as in Definition 2.4.7. Given a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ that models the desired property, the reachability over the set F of the gDTSG $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$ within the time horizon $[0, H]$ can be characterized by a value function defined as

$$\begin{aligned}\tilde{V}_n^{\rho,\lambda}(x,\hat{x},q) =&\mathbb{E}[\max_{H-n\leq t\leq H} \mathbf{1}_F(q(t))|x(H-n) = x, \hat{x}(H-n) = \hat{x}, q(H-n) = q] \\ =&\mathbb{P}_{(\rho,\lambda)\times(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}}\{\exists k \in [H-n, H], q(k) \in F\},\end{aligned} \tag{3.6.13}$$

for all $n \in [0, H]$, with $\rho \in \mathcal{P}^H$ and $\lambda \in \Lambda^H$ being Markov policies for Players I and II of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$, respectively. Given any Markov policy $\rho = (\rho_0,\ldots,\rho_{H-1})$ and $\lambda = (\lambda_0,\ldots,\lambda_{H-1})$, one initializes (3.6.13) with $\tilde{V}_0^{\rho,\lambda}(x,\hat{x},q) = 1$ when $q \in F$, and $\tilde{V}_0^{\rho,\lambda}(x,\hat{x},q) = 0$ when $q \notin F$, and recursively calculate it as

$$\begin{aligned}\tilde{V}_{n+1}^{\rho,\lambda}(x,\hat{x},q) =&\sum_{q^+\in Q}\int_{X\times\hat{X}} \tilde{V}_n^{\rho,\lambda}(x',\hat{x}',q^+)\bar{T}(\mathsf{d}x'\times d\hat{x}'\times q^+|x,\hat{x},q,\hat{u},w) \\ =&\int_{X\times\hat{X}} \tilde{V}_n^{\rho,\lambda}(x',\hat{x}',q')\mathscr{L}_T(\mathsf{d}x'\times d\hat{x}'|x,\hat{x},\hat{u},w),\end{aligned} \tag{3.6.14}$$

where $\hat{u} = \rho_{H-n-1}(x, \hat{x}, q)$, $w = \lambda_{H-n-1}(x, \hat{x}, q, \hat{u})$, and $q' = \tau(q, L \circ h(x'))$. In the case that $\lambda = \lambda_r$ is a randomized Markov policy over $[0, H-1]$, (3.6.14) should be rewritten as

$$\tilde{V}_{n+1}^{\rho,\lambda_r}(x, \hat{x}, q) = \int_W \int_{X \times \hat{X}} \tilde{V}_n^{\rho,\lambda_r}(x', \hat{x}', q') \mathscr{L}_T(\mathsf{d}x' \times d\hat{x}' | x, \hat{x}, \hat{u}, w) \lambda_{r,H-n-1}(dw | x, \hat{x}, q, \hat{u}). \tag{3.6.15}$$

In both (3.6.14) and (3.6.15), one has

$$\mathbb{P}_{(\rho,\lambda)\times(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}}\{\exists k \le H, q(k) \in F\} = \tilde{V}_{n+1}^{\rho,\lambda}(x_0, \hat{x}_0, \bar{q}_0), \tag{3.6.16}$$

with $\bar{q}_0 = \tau(q_0, L \circ h(x_0))$, $x_0 \in X_0$, and $\hat{x}_0 \in \hat{X}_0$ with $(x_0, \hat{x}_0) \in \mathscr{R}$.

Lemma 3.6.3. *Consider a Markov policy ρ over the time horizon $[0, H-1]$ for Player I of the gDTSG $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$. For any randomized Markov policy $\lambda_r \in \Lambda^H$ for Player II of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$, one has*

$$\tilde{V}_n^{\rho,\lambda'}(x_0, \hat{x}_0, \bar{q}_0) \le \tilde{V}_n^{\rho,\lambda_r}(x_0, \hat{x}_0, \bar{q}_0) \tag{3.6.17}$$

and

$$\tilde{V}_n^{\rho,\lambda_r}(x_0, \hat{x}_0, \bar{q}_0) \le \tilde{V}_n^{\rho,\lambda''}(x_0, \hat{x}_0, \bar{q}_0) \tag{3.6.18}$$

for all $n \in [0, H]$, with $\bar{q}_0 = \tau(q_0, L \circ h(x_0))$, $x_0 \in X_0$, and $\hat{x}_0 \in \hat{X}_0$ with $(x_0, \hat{x}_0) \in \mathscr{R}$. Here, λ' and λ'' are nonrandomized Markov policies that are computed based on ρ, as

$$\lambda'_{H-n-1} \in \inf_{\lambda_{H-n-1} \in \Lambda} \int_{X \times \hat{X}} \tilde{V}_n^{\rho,\lambda}(x', \hat{x}', q') \mathscr{L}_T(\mathsf{d}x' \times d\hat{x}' | x, \hat{x}, \hat{u}, w), \tag{3.6.19}$$

and

$$\lambda''_{H-n-1} \in \sup_{\lambda_{H-n-1} \in \Lambda} \int_{X \times \hat{X}} \tilde{V}_n^{\rho,\lambda}(x', \hat{x}', q') \mathscr{L}_T(\mathsf{d}x' \times d\hat{x}' | x, \hat{x}, \hat{u}, w), \tag{3.6.20}$$

for all $n \in [0, H]$, with $\hat{u} = \rho_{H-n-1}(x, \hat{x}, q)$, and $w = \lambda_{H-n-1}(x, \hat{x}, q, \hat{u})$.

Proof: First, (3.6.17) in Lemma 3.6.3 is shown by induction. When $n = 0$, according to the initialization of $\tilde{V}_0^{\rho,\lambda'}(x, \hat{x}, q)$, one has $\tilde{V}_0^{\rho,\lambda'}(x, \hat{x}, q) = \tilde{V}_0^{\rho,\lambda_r}(x, \hat{x}, q)$ so that (3.6.17) holds. Suppose that (3.6.17) is met when $n = k$. Then, when $n = k+1$, one has

$$\begin{aligned}
\tilde{V}_{k+1}^{\rho,\lambda_r}(x, \hat{x}, q) &= \int_W \int_{X \times \hat{X}} \tilde{V}_k^{\rho,\lambda_r}(x', \hat{x}', q') \mathscr{L}_T(\mathsf{d}x' \times d\hat{x}' | x, \hat{x}, \hat{u}, w) \lambda_{r,H-k-1}(dw | x, \hat{x}, q, \hat{u}) \\
&\ge \int_W \int_{X \times \hat{X}} \tilde{V}_k^{\rho,\lambda'}(x', \hat{x}', q') \mathscr{L}_T(\mathsf{d}x' \times d\hat{x}' | x, \hat{x}, \hat{u}, w) \lambda_{r,H-k-1}(dw | x, \hat{x}, q, \hat{u}) \quad \text{(c1)} \\
&\ge \int_{X \times \hat{X}} \tilde{V}_k^{\rho,\lambda'}(x', \hat{x}', q') \mathscr{L}_T(\mathsf{d}x' \times d\hat{x}' | x, \hat{x}, \hat{u}, w') \int_W \lambda_{r,H-k-1}(dw | x, \hat{x}, q, \hat{u}) \quad \text{(c2)}
\end{aligned}$$

$$= \int_{X\times\hat{X}} \tilde{V}_k^{\rho,\lambda'}(x',\hat{x}',q')\mathscr{L}_T(\mathsf{d}x'\times d\hat{x}'|x,\hat{x},\hat{u},w') = \tilde{V}_{k+1}^{\rho,\lambda'}(x,\hat{x},q).$$

Note that (c1) holds since (3.6.17) is supposed to be met when $n=k$, and (c2) holds with $w'=\lambda'_{H-k-1}(x,\hat{x},q,\hat{u})$ according to (3.6.19). Thus, one has (3.6.17) also holds for $n=k+1$, which completes the proof for (3.6.17). The proof of (3.6.18) can be proceeded similar to (3.6.17), and is omitted here for the sake of brevity. ■

So far, I am ready to prove the results in Section 3.3.

3.6.2.1 Required Lemmas and the proof for Theorem 3.3.4

To show Theorem 3.3.4, Lemma 3.6.4, Lemma 3.6.6, and some additional definitions are needed as well.

Lemma 3.6.4. *Consider a gDTSG $\mathfrak{D}=(X,U,W,X_0,T,Y,h)$ and its finite abstraction $\widehat{\mathfrak{D}}=(\hat{X},\hat{U},\hat{W},\hat{X}_0,\hat{T},Y,\hat{h})$ with $\widehat{\mathfrak{D}}\preceq_\epsilon^\delta\mathfrak{D}$, and a DFA $\mathcal{A}=(Q,q_0,\Pi,\tau,F)$ modeling the desired property. Given a Markov policy ρ for Player I of the gDTSG $\widehat{\mathfrak{D}}\otimes\mathcal{A}$ over time horizon $[0,H-1]$, a Markov policy $\tilde{\rho}$ for Player I of the gDTSG $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}$ is constructed such that $\forall k\in[0,H-1]$, $\tilde{\rho}_k(x,\hat{x},q)=\rho_k(\hat{x},q)$. Then, for any Markov policy $\tilde{\lambda}$ for Player II of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}$, one has*

$$\bar{V}_n^{\rho,\lambda_*(\rho)}(\hat{x},q)\le\tilde{V}_n^{\tilde{\rho},\tilde{\lambda}}(x,\hat{x},q), \tag{3.6.21}$$

for all $n\in[0,H]$ and $(x,\hat{x})\in\mathscr{R}$ as in (3.2.15), with $\lambda_*(\rho)$ as in (3.3.5), $\bar{V}_n^{\rho,\lambda_*(\rho)}(\hat{x},q)$ computed as in (3.3.1), and $\tilde{V}_n^{\tilde{\rho},\tilde{\lambda}}(x,\hat{x},q)$ as in (3.6.14).

Proof: The proof of Lemma 3.6.4 is performed by induction. Here, λ_* is used to denote $\lambda_*(\rho)$ in the following discussion. Additionally, one only needs to focus on the cases in which $q\notin F$ since (3.6.21) holds trivially for all $n\in\mathbb{N}$ when $q\in F$. According to the initialization of $\bar{V}_0^{\rho,\lambda_*}(\hat{x},q)$ and $\tilde{V}_0^{\tilde{\rho},\tilde{\lambda}}(x,\hat{x},q)$, one has $\bar{V}_0^{\rho,\lambda_*}(\hat{x},q)=\tilde{V}_0^{\tilde{\rho},\tilde{\lambda}}(x,\hat{x},q)$. Therefore, (3.6.21) holds when $n=0$. Suppose that (3.6.21) holds when $n=k$. Then, for $n=k+1$, one has

$$\begin{aligned}
&\bar{V}_{k+1}^{\rho,\lambda_*}(\hat{x},q)\\
&= (1-\delta)\sum_{\hat{x}'\in\hat{X}}\bar{V}_k^{\rho,\lambda_*}(\hat{x}',\underline{q}(\hat{x}',q))\hat{T}(\hat{x}'|\hat{x},\hat{u},\hat{w})\\
&\quad\text{with } \hat{u}=\rho_{H-k-1}(\hat{x},q) \text{ and } \hat{w}=\lambda_{*H-k-1}(\hat{x},q,\hat{u})\\
&\le(1-\delta)\sum_{\hat{x}'\in\hat{X}}\bar{V}_k^{\rho,\lambda_*}(\hat{x}',\underline{q}(\hat{x}',q))\hat{T}(\hat{x}'|\hat{x},\hat{u},f_{\hat{W}}) &\text{(c1)}\\
&\le(1-\delta)\sum_{\hat{x}'\in\hat{X}}\bar{V}_k^{\rho,\lambda_*}(\hat{x}',\underline{q}(\hat{x}',q))\Big(\frac{1}{1-\delta}\int_{x'\in\bar{\mathscr{R}}_{\hat{x}'}}\mathscr{L}_T(\mathsf{d}x'|x,\hat{x},\hat{x}',\hat{u},w)\Big)\hat{T}(\hat{x}'|\hat{x},\hat{u},\Pi_w(w)), &\text{(c2)}
\end{aligned}$$

$$
\begin{aligned}
&= \int_{\mathscr{R}} \bar{V}_k^{\rho,\lambda_*}(\hat{x}', \underline{q}(\hat{x}', q)) \mathscr{L}_T(\mathsf{d}x'|x, \hat{x}, \hat{x}', \hat{u}, w)\hat{T}(\hat{x}'|\hat{x}, \hat{u}, \Pi_w(w)) \\
&= \int_{\mathscr{R}} \bar{V}_k^{\rho,\lambda_*}(\hat{x}', \underline{q}(\hat{x}', q)) \mathscr{L}_T(\mathsf{d}x' \times d\hat{x}'|x, \hat{x}, \hat{u}, w) && \text{(c3)} \\
&\leq \int_{\mathscr{R}} \bar{V}_k^{\rho,\lambda_*}(\hat{x}', q') \mathscr{L}_T(\mathsf{d}x' \times d\hat{x}'|\hat{x}, x, \hat{u}, w), && \text{(c4)} \\
&\leq \int_{\mathscr{R}} \tilde{V}_k^{\tilde{\rho},\tilde{\lambda}}(x', \hat{x}', q') \mathscr{L}_T(\mathsf{d}x' \times d\hat{x}'|x, \hat{x}, \hat{u}, w) \\
&\leq \int_{X \times \hat{X}} \tilde{V}_k^{\tilde{\rho},\tilde{\lambda}}(x', \hat{x}', q') \mathscr{L}_T(\mathsf{d}x' \times d\hat{x}'|x, \hat{x}, \hat{u}, w) = \tilde{V}_{k+1}^{\tilde{\rho},\tilde{\lambda}}(x, \hat{x}, q),
\end{aligned}
$$

where $f_{\hat{W}}$ is a functions that assigns a probability measure over $(\hat{W}, \mathcal{B}(\hat{W}))$, $\bar{\mathscr{R}}_{\hat{x}'} = \{x' \in X | (x', \hat{x}') \in \mathscr{R}\}$, $\Pi_w(w)$ is as in (2.4.6), and $\mathscr{L}_T(\mathsf{d}x'|x, \hat{x}, \hat{x}', \hat{u}, w)$ is the conditional probability of x' as in (2.4.7). In the chain of equations above, (c1) holds due to the computation of λ_* as in (3.3.5), (c2) holds with $w = \tilde{\lambda}_{H-k-1}(x, \hat{x}, q, \hat{u})$ according to Assumption 3.3.2, (c3) holds according to (2.4.7), and (c4) holds with $q' = \tau(q, L \circ h(x'))$, since $\bar{V}_k^{\rho,\lambda_*}(\hat{x}', q') \geq \bar{V}_k^{\rho,\lambda_*}(\hat{x}', \underline{q}(\hat{x}', q))$ according to the definition of $\underline{q}$ as in (3.3.3). Thus, (3.6.21) also holds when $n = k + 1$, which completes the proof.■

Before showing Lemma 3.6.6, I introduce how to construct a control strategy $\mathbf{C}_\rho$ for Player I of the gDTSG $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ given a Markov policy $\tilde{\rho}$ for Player I of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$.

Definition 3.6.5. (Construction of $\mathbf{C}_\rho$) *Consider a gDTSG* $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}} = (X \times \hat{X}, \hat{U}, W, X_{0||}, \mathscr{L}_T, Y, h_{||})$, *a DFA* $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$, *and a Markov policy* $\tilde{\rho} = (\tilde{\rho}_0, \tilde{\rho}_1, \ldots, \tilde{\rho}_{H-1})$ *for Player I of* $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A} = \{\bar{X}, \bar{U}, \bar{W}, \bar{X}_0, \bar{T}, \bar{Y}, \bar{h}\}$. *A control strategy* $\mathbf{C}_\rho = (\mathsf{M}, \mathsf{U}, \mathsf{Y}, \mathsf{H}, \mathsf{M}_0, \pi_\mathsf{M}, \pi_\mathsf{Y})$ *is constructed for Player I of* $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ *with* $\mathsf{M} = X \times \hat{X} \times Q$; $\mathsf{U} = X \times \hat{X}$; $\mathsf{Y} = \hat{U}$; $\mathsf{H} = [0, H-1]$; *and* $\mathsf{M}_0 = \bar{X}_0$. *Furthermore,* π_M *updates* $\mathsf{m}(k) = (\mathsf{m}_X(k), \mathsf{m}_{\hat{X}}(k), \mathsf{m}_Q(k)) \in \mathsf{M}$ *at the time instant* $k \in \mathsf{H}\backslash\{0\}$ *with* $(\mathsf{m}_X(k), \mathsf{m}_{\hat{X}}(k)) = (x(k), \hat{x}(k))$, *where* $x(k) \in X$, $\hat{x}(k) \in \hat{X}$, *and* $\mathsf{m}_Q(k) = \tau\big(\mathsf{m}_Q(k-1), L \circ h(\mathsf{m}_X(k))\big)$; π_Y *updates* $\mathsf{y}(k) \in \mathsf{Y}$ *at the time instant* $k \in \mathsf{H}$ *with* $\mathsf{y}(k) = \tilde{\rho}_k(\mathsf{m}_X(k), \mathsf{m}_{\hat{X}}(k), \mathsf{m}_Q(k))$.

In brief, $\mathbf{C}_\rho$ takes the state $(x(k), \hat{x}(k))$ of $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ and the state $q(k)$ of $\mathcal{A}$ as its memory state at the time instant k. At runtime, it provides input $\hat{u}(k)$ to $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ according to the Markov policy $\tilde{\rho}_k$ based on its memory state.

Lemma 3.6.6. *Consider a gDTSG* $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}} = (X \times \hat{X}, \hat{U}, W, X_{0||}, \mathscr{L}_T, Y, h_{||})$, *a DFA* $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$, *and their product gDTSG* $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$. *Given a Markov policy* $\tilde{\rho}$ *for Player I of* $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$, *for any control strategy* $\mathbf{C}_\lambda$ *for Player II of* $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$, *one has*

$$
\mathbb{P}_{(\tilde{\rho},\lambda') \times (\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}}\{\exists k \leq H, q(k) \in F\} \leq \mathbb{P}_{(\mathbf{C}_\rho, \mathbf{C}_\lambda) \times \mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\},
$$

with λ' *as in* (3.6.19), *and* $\mathbf{C}_\rho$ *being a control strategy for Player I of* $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ *constructed based on* $\tilde{\rho}$ *as in Definition 3.6.5.*

Proof: Given a path $\omega_k \in \Omega$ of $\mathfrak{D}$, the memory state of $\mathbf{C}_\rho$ is the same as the state of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$ according to the construction of $\mathbf{C}_\rho$ as in Definition 3.6.5. Therefore, the same input $u(k) \in U$ is provided by $\tilde{\rho}$ and $\mathbf{C}_\rho$ given the same path ω_k. Moreover, given $(\omega_k, u(k))$, it is considered, without loss of generality, that $\mathbf{C}_\lambda$ chooses its adversarial input $w \in W$ according to a measurable stochastic kernel $T_W(W|\omega_k, u(k))$ over $(W, \mathcal{B}(W))$. This kernel corresponds to a randomized Markov policy λ_r for Player II of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$ to select $w(k)$ given the same ω_k and $u(k)$, such that

$$\mathbb{P}_{(\tilde{\rho},\lambda_r)\times(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}}\{\exists k \leq H, q(k) \in F\} = \mathbb{P}_{(\mathbf{C}_\rho,\mathbf{C}_\lambda)\times\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\}. \tag{3.6.22}$$

According to (3.6.17) and (3.6.16), one has

$$\mathbb{P}_{(\tilde{\rho},\lambda')\times(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}}\{\exists k \leq H, q(k) \in F\} \leq \mathbb{P}_{(\tilde{\rho},\lambda_r)\times(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}}\{\exists k \leq H, q(k) \in F\}, \tag{3.6.23}$$

with synthesized λ' based on $\tilde{\rho}$ as in (3.6.19). The proof is then completed by combining (3.6.22) and (3.6.23). ■

Before showing the proof for Theorem 3.3.4, I present how to construct the control strategy $\tilde{\mathbf{C}}_\rho$ for Player I of $\mathfrak{D}$ given the control strategy $\mathbf{C}_\rho$ for Player I of $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$.

Definition 3.6.7. (Construction of $\tilde{\mathbf{C}}_\rho$) *Consider a gDTSG* $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ *and its finite abstraction* $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ *with* $\widehat{\mathfrak{D}} \preceq^\delta_\epsilon \mathfrak{D}$. *Given a control strategy* $\mathbf{C}_\rho = (\mathsf{M}, \mathsf{U}, \mathsf{Y}, \mathsf{H}, \mathsf{M}_0, \pi_{\mathsf{M}}, \pi_{\mathsf{Y}})$ *for Player I of* $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ *that is constructed based on* $\tilde{\rho}$ *as proposed in Definition 3.6.5, a control strategy* $\tilde{\mathbf{C}}_\rho = (\tilde{\mathsf{M}}, \tilde{\mathsf{U}}, \tilde{\mathsf{Y}}, \tilde{\mathsf{H}}, \tilde{\mathsf{M}}_0, \tilde{\pi}_{\mathsf{M}}, \tilde{\pi}_{\mathsf{Y}})$ *is constructed for Player I of* $\mathfrak{D}$*, in which*

- $\tilde{\mathsf{M}} := \mathsf{M} \times W \times \hat{W} = X \times \hat{X} \times Q \times W \times \hat{W}$;
- $\tilde{\mathsf{U}} := \mathsf{U}_X \times W = X \times W$;
- $\tilde{\mathsf{Y}} := U$;
- $\tilde{\mathsf{H}} := \mathsf{H}$;
- $\tilde{\mathsf{m}}_0 = (\tilde{\mathsf{m}}_X(0), \tilde{\mathsf{m}}_{\hat{X}}(0), \tilde{\mathsf{m}}_Q(0), \tilde{\mathsf{m}}_W(0), \tilde{\mathsf{m}}_{\hat{W}}(0)) \in \tilde{\mathsf{M}}_0$, *with* $\tilde{\mathsf{m}}_X(0) = x_0$, *where* $x_0 \in X_0$; $\tilde{\mathsf{m}}_{\hat{X}}(0) = \hat{x}_0$ *such that* $(x_0, \hat{x}_0) \in \mathscr{R}$, *where* $\mathscr{R}$ *is as in* (3.2.15); $\tilde{\mathsf{m}}_Q(0) = \tau(q_0, L \circ h(\tilde{\mathsf{m}}_X(0)))$; $\tilde{\mathsf{m}}_W(0)$ *is initialized as* $\tilde{\mathsf{m}}_W(0) = w(0)$ *after Player II of* $\mathfrak{D}$ *has chosen* $w(0)$, *and* $\tilde{\mathsf{m}}_{\hat{W}}(0)$ *is initialized as* $\tilde{\mathsf{m}}_{\hat{W}}(0) = \Pi_w(w(0))$ *with* Π_w *as in* (2.4.6);
- $\tilde{\pi}_{\mathsf{M}}$ *updates* $(\tilde{\mathsf{m}}_X(k), \tilde{\mathsf{m}}_{\hat{X}}(k), \tilde{\mathsf{m}}_Q(k), \tilde{\mathsf{m}}_W(k), \tilde{\mathsf{m}}_{\hat{W}}(k)) \in \tilde{\mathsf{M}}$ *at all time instant* $k \in \mathsf{H}\backslash\{0\}$, *with the following steps:*
  1. *update* $\tilde{\mathsf{m}}_{\hat{X}}(k)$ *with the conditional kernel*

     $$\mathscr{L}_T(d\hat{x}|\tilde{\mathsf{m}}_{\hat{X}}(k-1), \tilde{\mathsf{m}}_X(k-1), x(k), \hat{u}(k-1), \tilde{\mathsf{m}}_W(k-1))$$

     as in (2.4.7), *with* $x(k)$ *the state of* $\mathfrak{D}$ *and* $\hat{u}(k-1) = \tilde{\rho}_{k-1}(\tilde{\mathsf{m}}_X(k-1), \tilde{\mathsf{m}}_{\hat{X}}(k-1), \tilde{\mathsf{m}}_Q(k-1))$;

2. *update* $\tilde{\mathsf{m}}_X(k)$ *with* $\tilde{\mathsf{m}}_X(k) = x(k)$*;*
3. *update* $\tilde{\mathsf{m}}_Q(k)$ *with* $\tilde{\mathsf{m}}_Q(k) = \tau\big(\tilde{\mathsf{m}}_Q(k-1), L \circ h(\tilde{\mathsf{m}}_X(k))\big)$*;*
4. *update* $\tilde{\mathsf{m}}_W(k)$ *with* $\tilde{\mathsf{m}}_W(k) = w(k)$ *after Player II of* $\mathfrak{D}$ *has selected* $w(k)$ *and accordingly update* $\tilde{\mathsf{m}}_{\hat{W}}(k)$ *as* $\tilde{\mathsf{m}}_{\hat{W}}(k) = \Pi_w(w(k))$ *with* Π_w *as in* (2.4.6)*;*

- $\tilde{\pi}_{\mathsf{Y}}$ *updates* $\mathsf{y}(k) \in \mathsf{Y}$ *at the time instant* $k \in \mathsf{H}$ *with*

$$\mathsf{y}(k) = \nu\big(\tilde{\mathsf{m}}_X(k), \tilde{\mathsf{m}}_{\hat{X}}(k), \tilde{\rho}_k(\tilde{\mathsf{m}}_X(k), \tilde{\mathsf{m}}_{\hat{X}}(k), \tilde{\mathsf{m}}_Q(k))\big),$$

with ν *being the interface function associated with the approximate probabilistic relation.*

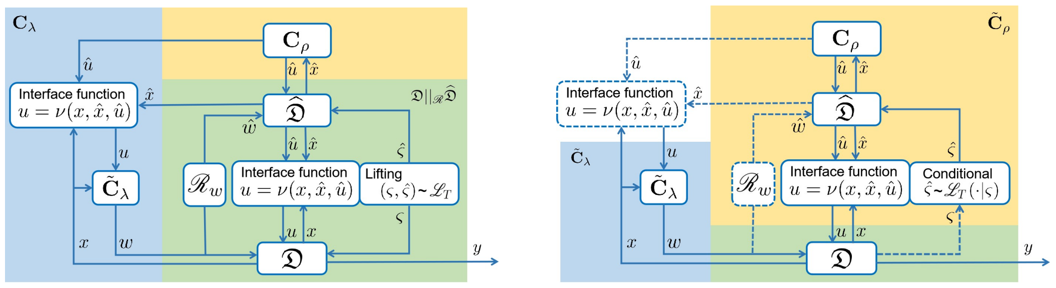

$\mathbf{C}_\lambda$
$\mathbf{C}_\rho$
$\hat{u}$
$\hat{x}$
Interface function
$u = \nu(x, \hat{x}, \hat{u})$
$\widehat{\mathfrak{D}}$
$\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$
$\hat{w}$
u
$\hat{\varsigma}$
$\tilde{\mathbf{C}}_\lambda$
$\mathscr{R}_w$
Lifting
$(\varsigma, \hat{\varsigma}) \sim \mathscr{L}_T$
ς
x
w
$\mathfrak{D}$
y
$\tilde{\mathbf{C}}_\rho$
Conditional
$\hat{\varsigma} \sim \mathscr{L}_T(\cdot|\varsigma)$

Figure 3.9: Left: Coupling gDTSG $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ (green region) controlled by $\mathbf{C}_\rho$ (yellow region) and $\mathbf{C}_\lambda$ (blue region). **Right:** A gDTSG $\mathfrak{D}$ (green region) controlled by $\tilde{\mathbf{C}}_\rho$ (yellow region) and $\tilde{\mathbf{C}}_\lambda$ (blue region).

Employing Definition 3.6.7, one can construct a control strategy $\tilde{\mathbf{C}}_\rho$ for Player I of the gDTSG $\mathfrak{D}$ given a control strategy $\mathbf{C}_\rho$ for Player I of $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$. Then, given any control strategy $\tilde{\mathbf{C}}_\lambda$ for Player II of $\mathfrak{D}$, the controlled gDTSG $(\tilde{\mathbf{C}}_\rho, \tilde{\mathbf{C}}_\lambda) \times \mathfrak{D}$ can be written as a controlled gDTSG $(\mathbf{C}_\rho, \mathbf{C}_\lambda) \times \mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ as depicted in Figure 3.9, where $\mathbf{C}_\lambda$ is constructed by combining $\tilde{\mathbf{C}}_\lambda$ with the interface function $\nu(x, \hat{x}, \hat{u})$. Accordingly, one has

$$\mathbb{P}_{(\tilde{\mathbf{C}}_\rho, \tilde{\mathbf{C}}_\lambda)\times\mathfrak{D}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\} = \mathbb{P}_{(\mathbf{C}_\rho, \mathbf{C}_\lambda)\times\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\}. \tag{3.6.24}$$

Now, I am ready to show the results of Theorem 3.3.4.

Proof of Theorem 3.3.4: Consider $x_0 \in X_0$ and $\hat{x}_0 \in \hat{X}_0$ with $(x_0, \hat{x}_0) \in \mathscr{R}$ and $\mathscr{R}$ as in (3.2.15). According to (3.6.16) and Lemma 3.6.4, for any Markov policy $\tilde{\lambda}$ for Player II of the gDTSG $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$, one has

$$\bar{V}_H^{\rho,\lambda_*(\rho)}(\hat{x}_0, \bar{q}_0) \leq \mathbb{P}_{(\tilde{\rho},\tilde{\lambda})\times(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}}\{\exists k \leq H, q(k) \in F\}, \tag{3.6.25}$$

with $\tilde{\rho}$ being a Markov policy for Player I of the gDTSG $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}$ that is constructed based on ρ as discussed in Lemma 3.6.4. Moreover, Lemma 3.6.6 indicates that given a Markov policy $\tilde{\rho}$ for Player I of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$ and a control strategy $\mathbf{C}_\rho$ for Player I of $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ that is constructed based on $\tilde{\rho}$ as in Definition 3.6.5, for any control strategy $\mathbf{C}_\lambda$ for Player II of $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$, one has

$$\mathbb{P}_{(\tilde{\rho},\lambda')\times(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}}\{\exists k \leq H, q(k) \in F\} \leq \mathbb{P}_{(\mathbf{C}_\rho,\mathbf{C}_\lambda)\times\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\}, \quad (3.6.26)$$

where λ' is a Markov policy for Player II of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$ computed as in (3.6.19). Since (3.6.25) holds for any Markov policy for Player II of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$, by combining (3.6.25) and (3.6.26), one has

$$\bar{V}_H^{\rho,\lambda_*(\rho)}(\hat{x}_0, \bar{q}_0) \leq \mathbb{P}_{(\mathbf{C}_\rho,\mathbf{C}_\lambda)\times\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\}. \quad (3.6.27)$$

Finally, considering (3.6.27) and (3.6.24), one has

$$\bar{V}_H^{\rho,\lambda_*(\rho)}(\hat{x}_0, \bar{q}_0) \leq \mathbb{P}_{(\tilde{\mathbf{C}}_\rho,\tilde{\mathbf{C}}_\lambda)\times\mathfrak{D}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\}, \quad (3.6.28)$$

where $\tilde{\mathbf{C}}_\rho$ is a control strategy for Player I of $\mathfrak{D}$ that is constructed based on $\mathbf{C}_\rho$ as in Definition 3.6.7. Considering the construction of $\tilde{\rho}$ as in Lemma 3.6.4 based on ρ, $\mathbf{C}_\rho$ as in Definition 3.6.5 based on $\tilde{\rho}$, and $\tilde{\mathbf{C}}_\rho$ as in Definition 3.6.7 based on $\mathbf{C}_\rho$, $\tilde{\mathbf{C}}_\rho$ in (3.6.28) can be constructed as in Definition 3.3.1 directly based on a Markov policy ρ for Player I of $\widehat{\mathfrak{D}} \otimes \mathcal{A}$, which completes the proof. ■

3.6.2.2 Proof for Lemma 3.3.8

In this subsection, $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{X}_0, \hat{T}, Y, \hat{h})$ denotes the finite abstraction for the stochastic systems without rational adversarial input, and by $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ a DFA modeling the desired property. Additionally, $\bar{V}_n^\rho(\hat{x}, q)$ is used to replace $\bar{V}_n^{\rho,\lambda}(\hat{x}, q)$ as in (3.3.1), since λ does not play a role in stochastic systems of interest here. Accordingly, initializing $\bar{V}_n^\rho(\hat{x}, q)$ with $\bar{V}_0^\rho(\hat{x}, q) = 1$ when $q \in F$ and $\bar{V}_0^\rho(\hat{x}, q) = 0$, otherwise, $\bar{V}_{n+1}^\rho(\hat{x}, q)$ is then recursively computed as

$$\bar{V}_{n+1}^\rho(\hat{x}, q) := (1-\delta) \sum_{\hat{x}' \in \hat{X}} \bar{V}_n^\rho(\hat{x}', \underline{q}(\hat{x}', q))\hat{T}(\hat{x}'|\hat{x}, \hat{u}),$$

when $q \notin F$, and $\bar{V}_{n+1}^\rho(\hat{x}, q) := 1$ otherwise. Furthermore, $\underline{q}$ as in (3.3.3) should accordingly be modified as

$$\underline{q}(\hat{x}', q) = \arg\min_{q' \in Q'_\epsilon(\hat{x}')} \bar{V}_n^\rho(\hat{x}', q'), \quad (3.6.29)$$

where $Q'_\epsilon(\hat{x}')$ is the set as in (3.3.4). Before showing the results for Lemma 3.3.8, I briefly introduce some results in [74] for the sake of completeness. Considering a Markov policy $\rho = (\rho_0, \rho_1, \ldots, \rho_{H-1})$ over time horizon $[0, H-1]$, a value function $V_n^\rho : \hat{X} \times Q \to [0,1]$ is defined in [74]. Initialized with $V_0^\rho(\hat{x}, q) = 0$, $V_n^\rho(\hat{x}, q)$ is then recursively computed as [74, equation (41)]:

$$V_{k+1}^\rho(\hat{x}, q) := \mathbf{L}\Big(\sum_{\hat{x}' \in \hat{X}} \min_{q' \in \bar{\tau}(q, \hat{x}')} \max\{\mathbf{1}_F(q'), V_k^\rho(\hat{x}', q')\}\hat{T}(d\hat{x}'|\hat{x}, \hat{u}) - \delta\Big), \quad (3.6.30)$$

with $\mathbf{L} : \mathbb{R} \to [0,1]$ being the truncation function $\mathbf{L}(\cdot) := \min(1, \max(0,\cdot))$; $\bar{\tau}(q, \hat{x}') := \{\tau(q,\alpha) \text{ with } \alpha \in L(\mathcal{N}_\epsilon(\hat{h}(\hat{x}')))\}$, where $\mathcal{N}_\epsilon(\hat{y}) := \{y \in Y \mid \|y - \hat{y}\| \leq \epsilon\}$; $\mathbf{1}_F(\cdot)$ being an indicator function for the set F, i.e., if $q' \in F$ then $\mathbf{1}_F(q') = 1$, otherwise $\mathbf{1}_F(q') = 0$; and $\hat{u} = \rho_{H-k-1}(\hat{x})$. With these notations, $\mathcal{S}(\hat{x}_0)$ as in (3.3.13) can be computed as [74, equation (43)]:

$$\mathcal{S}(\hat{x}_0) := \min_{\bar{q}_0 \in \bar{\tau}(q_0,\hat{x}_0)} \max(\mathbf{1}_F(\bar{q}_0), V^\rho_H(\hat{x}_0, \bar{q}_0)). \tag{3.6.31}$$

Moreover, Lemma 3.6.8 is required for proving Lemma 3.3.8.

Lemma 3.6.8. *If one has*

$$V^\rho_n(\hat{x}', q) \leq \bar{V}^\rho_n(\hat{x}', q), \tag{3.6.32}$$

for all $\hat{x}' \in \hat{X}$ and $q \in Q$, with $n \in \mathbb{N}$, then one has

$$\min_{q' \in \bar{\tau}(q,\hat{x}')} \max\{\mathbf{1}_F(q'), V^\rho_n(\hat{x}', q')\} \leq \bar{V}^\rho_n(\hat{x}', \underline{q}(\hat{x}', q)). \tag{3.6.33}$$

Proof of Lemma 3.6.8: One can prove Lemma 3.6.8 by showing two cases:

- (Case 1) If $\exists \hat{x} \in Q'_\epsilon(\hat{x}')$ such that $\tau(q,\hat{x}) \notin F$, then one has

  $$\min_{q' \in \bar{\tau}(q,\hat{x}')} \max\{\mathbf{1}_F(q'), V^\rho_n(\hat{x}', q')\} = \min_{q' \in \bar{\tau}(q,\hat{x}')} V^\rho_n(\hat{x}', q') = \min_{q' \in Q'_\epsilon(\hat{x}')} V^\rho_n(\hat{x}', q'). \tag{3.6.34}$$

  Meanwhile, according to the definition of $\underline{q}$ as in (3.6.29), one has

  $$\bar{V}^\rho_n(\hat{x}', \underline{q}(\hat{x}', q)) = \min_{q' \in Q'_\epsilon(\hat{x}')} \bar{V}^\rho_n(\hat{x}', q') \tag{3.6.35}$$

  Then, with (3.6.34), (3.6.35) and (3.6.32), one can readily verify that (3.6.33) holds in Case 1.

- (Case 2) If $\forall \hat{x} \in Q'_\epsilon(\hat{x}')$ such that $\tau(q,\hat{x}) \in F$, one has $\bar{V}^\rho_n(\hat{x}', \underline{q}(\hat{x}', q)) = 1$. Therefore, (3.6.33) holds trivially in Case 2.

Then, the proof for Lemma 3.6.8 is completed by combining Case 1 and Case 2. ∎

Now, I am ready to show the results for Lemma 3.3.8.

Proof of Lemma 3.3.8: First, one can show

$$V^\rho_n(\hat{x}, q) \leq \bar{V}^\rho_n(\hat{x}, q) \tag{3.6.36}$$

holds for all $n \in \mathbb{N}$ by induction. Note that one only needs to focus on the cases in which $q \notin F$ since $\bar{V}^\rho_n(\hat{x}, q) = 1$ when $q \in F$ so that (3.6.36) holds trivially. According to the initialization of $\bar{V}^\rho_0(\hat{x}, q)$ and $V_0(\hat{x}, q)$, one has $\bar{V}^\rho_0(\hat{x}, q) \geq V_0(\hat{x}, q)$. Therefore, (3.6.36) holds when $n = 0$. Suppose that (3.6.36) is met when $n = k$. Then, when $n = k+1$, one only needs to focus on the case in which

$$\sum_{\hat{x}' \in \hat{X}} \min_{q' \in \bar{\tau}(q,\hat{x}')} \max\{\mathbf{1}_F(q'), V^\rho_k(\hat{x}', q')\} \hat{T}(d\hat{x}' | \hat{x}, \hat{u}) - \delta \geq 0. \tag{3.6.37}$$

Otherwise, $V^{\rho}_{k+1}(\hat{x},q) \leq \bar{V}^{\rho}_{k+1}(\hat{x},q)$ holds trivially since one has $V^{\rho}_{k+1}(\hat{x},q)=0$ according to the definition of function $\mathbf{L}(\cdot)$ as in (3.6.30). When (3.6.37) holds, one has

$$\begin{aligned}
V^{\rho}_{k+1}(\hat{x},q) =& \sum_{\hat{x}'\in\hat{X}} \min_{q'\in\bar{\tau}(q,\hat{x}')} \max\{\mathbf{1}_F(q'), V^{\rho}_k(\hat{x}',q')\}\hat{T}(d\hat{x}'|\hat{x},\hat{u}) - \delta,\\
\leq& \sum_{\hat{x}'\in\hat{X}} \bar{V}^{\rho}_k(\hat{x}',\underline{q}(\hat{x}',q))\hat{T}(d\hat{x}'|\hat{x},\hat{u}) - \delta \qquad (3.6.38)\\
\leq& \sum_{\hat{x}'\in\hat{X}}\bar{V}^{\rho}_k(\hat{x}',\underline{q}(\hat{x}',q))\hat{T}(d\hat{x}'|\hat{x},\hat{u}) - \delta\Big(\sum_{\hat{x}'\in\hat{X}}\bar{V}^{\rho}_k(\hat{x}',\underline{q}(\hat{x}',q))\hat{T}(d\hat{x}'|\hat{x},\hat{u})\Big)\\
=&(1-\delta)\sum_{\hat{x}'\in\hat{X}} \bar{V}^{\rho}_k(\hat{x}',\underline{q}(\hat{x}',q))\hat{T}(\hat{x}'|\hat{x},\hat{u}) = \bar{V}^{\rho}_{k+1}(\hat{x},q).
\end{aligned}$$

Note that (3.6.38) holds according to Lemma 3.6.8. Therefore, one has (3.6.36) also hold for $n=k+1$, so that (3.6.36) holds for all $n\in\mathbb{N}$. Then, one can readily verify

$$\bar{V}^{\rho}_H(\hat{x}_0,\bar{q}_0) \geq \mathcal{S}(\hat{x}_0)$$

by considering (3.6.29), (3.6.31), (3.6.36), and Lemma 3.6.8, which completes the proof. ■

3.6.2.3 Required Lemmas and the proof for Theorem 3.3.10

In order to show the results of Theorem 3.3.10, Lemma 3.6.9 and 3.6.10 are required.

Lemma 3.6.9. *Consider a gDTSG* $\mathfrak{D}=(X,U,W,X_0,T,Y,h)$ *and its finite abstraction* $\widehat{\mathfrak{D}}=(\hat{X},\hat{U},\hat{W},\hat{X}_0,\hat{T},Y,\hat{h})$ *with* $\widehat{\mathfrak{D}}\preceq^{\delta}_{\epsilon}\mathfrak{D}$*, and a DFA* $\mathcal{A}=(Q,q_0,\Pi,\tau,F)$ *characterizing the desired property. Given a Markov policy* ρ *for Player I of the gDTSG* $\widehat{\mathfrak{D}}\otimes\mathcal{A}$ *over the time horizon* $[0,H-1]$*, construct a Markov policy* $\tilde{\rho}$ *for Player I of the gDTSG* $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}$ *such that* $\forall k\in[0,H-1]$, $\tilde{\rho}_k(x,\hat{x},q)=\rho_k(\hat{x},q)$*. Then, for any Markov policy* $\tilde{\lambda}$ *for Player II of* $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}$*, one has*

$$\underline{V}^{\rho,\lambda^*(\rho)}_n(\hat{x},q) \geq \tilde{V}^{\tilde{\rho},\tilde{\lambda}}_n(x,\hat{x},q), \qquad (3.6.39)$$

for all $n\in[0,H]$, $(x,\hat{x})\in\mathscr{R}$ *as in* (3.2.15)*, with* $\lambda^*(\rho)$ *computed as in* (3.3.18)*,* $\underline{V}^{\rho,\lambda^*(\rho)}_n(\hat{x},q)$ *as in* (3.3.15) *and* $\tilde{V}^{\tilde{\rho},\tilde{\lambda}}_n(x,\hat{x},q)$ *as in* (3.6.14)*.*

Proof: The proof is followed by induction. Here, $\lambda^*(\rho)$ is denoted by λ^* for the sake of clarity. Moreover, one only needs to focus on the cases in which $q\notin F$ since (3.6.39) holds trivially for all $n\in\mathbb{N}$ when $q\in F$. For $n=0$, one can readily verify that $\underline{V}^{\rho,\lambda^*}_0(\hat{x},q)=\tilde{V}^{\tilde{\rho},\tilde{\lambda}}_0(x,\hat{x},q)$ according to the initialization of $\underline{V}^{\rho,\lambda^*}_0(\hat{x},q)$ and $\tilde{V}^{\tilde{\rho},\tilde{\lambda}}_0(x,\hat{x},q)$. Thus, (3.6.39) holds for $n=0$. Suppose that (3.6.39) holds for $n=k$. Then, for

$n = k + 1$, one has

$$\begin{aligned}
1 - \underline{V}^{\rho,\lambda^*}_{k+1}(\hat{x}, q) &= (1-\delta) - (1-\delta)\sum_{\hat{x}'\in\hat{X}} \underline{V}^{\rho,\lambda^*}_{k}(\hat{x}', \bar{q}(\hat{x}', q))\hat{T}(\hat{x}'|\hat{x}, \hat{u}, \hat{w})\\
&\quad \text{with } \hat{u} = \rho_{H-k-1}(\hat{x}) \text{ and } \hat{w} = \lambda^*_{H-k-1}(\hat{x}, \hat{u})\\
&\le (1-\delta) - (1-\delta)\sum_{\hat{x}'\in\hat{X}} \underline{V}^{\rho,\lambda^*}_{k}(\hat{x}', \bar{q}(\hat{x}', q))\hat{T}(\hat{x}'|\hat{x}, \hat{u}, f_{\hat{W}})\\
&= (1-\delta)\sum_{\hat{x}'\in\hat{X}} \Big(1 - \underline{V}^{\rho,\lambda^*}_{k}(\hat{x}', \bar{q}(\hat{x}', q))\Big)\hat{T}(\hat{x}'|\hat{x}, \hat{u}, f_{\hat{W}})\\
&\le (1-\delta)\sum_{\hat{x}'\in\hat{X}} \Big(1 - \underline{V}^{\rho,\lambda^*}_{k}(\hat{x}', \bar{q}(\hat{x}', q))\Big)\Big(\frac{1}{1-\delta}\int_{x'\in\bar{\mathscr{R}}_{\hat{x}'}} \mathscr{L}_T(\mathsf{d}x'|x, \hat{x}, \hat{x}', \hat{u}, w)\Big)\hat{T}(\hat{x}'|\hat{x}, \hat{u}, \Pi_w(w)),\\
&\text{with } w = \tilde{\lambda}_{H-k-1}(x, \hat{x}, q, \hat{u})\\
&= \int_{\mathscr{R}} \Big(1 - \underline{V}^{\rho,\lambda^*}_{k}(\hat{x}', \bar{q}(\hat{x}', q))\Big)\mathscr{L}_T(\mathsf{d}x'|x, \hat{x}, \hat{x}', \hat{u}, w)\hat{T}(\hat{x}'|\hat{x}, \hat{u}, \Pi_w(w))\\
&= \int_{\mathscr{R}} \Big(1 - \underline{V}^{\rho,\lambda^*}_{k}(\hat{x}', \bar{q}(\hat{x}', q))\Big)\mathscr{L}_T(\mathsf{d}x' \times d\hat{x}'|x, \hat{x}, \hat{u}, w)\\
&\le \int_{\mathscr{R}} \Big(1 - \underline{V}^{\rho,\lambda^*}_{k}(\hat{x}', q')\Big)\mathscr{L}_T(\mathsf{d}x' \times d\hat{x}'|\hat{x}, x, \hat{u}, w), \text{ with } \; q' = \tau(q, L\circ h(x'))\\
&\le \int_{\mathscr{R}} \Big(1 - \tilde{V}^{\tilde{\rho},\tilde{\lambda}}_{k}(x', \hat{x}', q')\Big)\mathscr{L}_T(\mathsf{d}x' \times d\hat{x}'|x, \hat{x}, \hat{u}, w)\\
&\le \int_{X\times\hat{X}} \Big(1 - \tilde{V}^{\tilde{\rho},\tilde{\lambda}}_{k}(x', \hat{x}', q')\Big)\mathscr{L}_T(\mathsf{d}x' \times d\hat{x}'|x, \hat{x}, \hat{u}, w) = 1 - \tilde{V}^{\tilde{\rho},\tilde{\lambda}}_{k+1}(x, \hat{x}, q),
\end{aligned}$$

where $f_{\hat{W}}$ is a functions that assigns a probability measure over $(\hat{W}, \mathcal{B}(\hat{W}))$, $\bar{\mathscr{R}}_{\hat{x}'} = \{x' \in X|(x', \hat{x}') \in \mathscr{R}\}$, and $\mathscr{L}_T(\mathsf{d}x'|x, \hat{x}, \hat{x}', \hat{u}, w)$ is the conditional probability of x' as in (2.4.7). Note that the chain of equations above hold similarly to those in the proof of Lemma 3.6.4. Thus, one has $\underline{V}^{\rho,\lambda^*}_{k+1}(\hat{x}, q) \ge \tilde{V}^{\tilde{\rho},\tilde{\lambda}}_{k+1}(x, \hat{x}, q)$ so that (3.6.21) also holds for $n = k + 1$, which concludes the proof. ■

Lemma 3.6.10. *Consider a gDTSG* $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}} = (X \times \hat{X}, \hat{U}, W, X_{0||}, \mathscr{L}_T, Y, h_{||})$, *a DFA* $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$, *and their product gDTSG* $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$. *Given a Markov policy* $\tilde{\rho}$ *for Player I of* $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$, *for any control strategy* $\mathbf{C}_\lambda$ *for Player II of* $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$, *one has*

$$\mathbb{P}_{(\tilde{\rho},\lambda'')\times(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}}\{\exists k \le H, q(k) \in F\} \ge \mathbb{P}_{(\mathbf{C}_\rho,\mathbf{C}_\lambda)\times\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}}\{\exists k \le H, y_{\omega k} \models \mathcal{A}\},$$

with λ'' *as in* (3.6.20), *and* $\mathbf{C}_\rho$ *being a control strategy for Player I of* $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ *constructed based on* $\tilde{\rho}$ *as in Definition 3.6.5.*

Lemma 3.6.10 can be proved similar to that of Lemma 3.6.6 with the help of (3.6.18) and (3.6.20). Employing Lemmas 3.6.9 and 3.6.10, I show the results of Theorem 3.3.10 as follows.

Proof of Theorem 3.3.10: Consider $x_0 \in X_0$ and $\hat{x}_0 \in \hat{X}_0$ with $(x_0, \hat{x}_0) \in \mathscr{R}$ and $\mathscr{R}$ as in (3.2.15). According to (3.6.16) and Lemma 3.6.9, for any Markov policy $\tilde{\lambda}$ for Player II of the gDTSG $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$, one has

$$\underline{V}_H^{\rho,\lambda^*(\rho)}(\hat{x}_0, \bar{q}_0) \geq \mathbb{P}_{(\tilde{\rho},\tilde{\lambda})\times(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}}\{\exists k \leq H, q(k) \in F\}, \tag{3.6.40}$$

with $\tilde{\rho}$ being a Markov policy for Player I of the gDTSG $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}$ that is constructed based on ρ as in Lemma 3.6.9. Furthermore, according to Lemma 3.6.10, given a Markov policy $\tilde{\rho}$ for Player I of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$ and a control strategy $\mathbf{C}_\rho$ for Player I of $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$ constructed as in Definition 3.6.5 based on $\tilde{\rho}$, for any control strategy $\mathbf{C}_\lambda$ for Player II of $\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}$, one has

$$\mathbb{P}_{(\tilde{\rho},\lambda'')\times(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}}\{\exists k \leq H, q(k) \in F\} \geq \mathbb{P}_{(\mathbf{C}_\rho,\mathbf{C}_\lambda)\times\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\}, \tag{3.6.41}$$

where λ'' is a Markov policy for Player II of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}})\otimes\mathcal{A}$ computed as in (3.6.20). Note that (3.6.40) holds for any arbitrary Markov policy for Player II of $(\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}) \otimes \mathcal{A}$. By combining (3.6.40) and (3.6.41), one has

$$\underline{V}_H^{\rho,\lambda^*(\rho)}(\hat{x}_0, \bar{q}_0) \geq \mathbb{P}_{(\mathbf{C}_\rho,\mathbf{C}_\lambda)\times\mathfrak{D}||_{\mathscr{R}}\widehat{\mathfrak{D}}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\}. \tag{3.6.42}$$

Then, similar to the proof of Theorem 3.3.4, one can readily verify (3.3.19) considering (3.6.42) and (3.6.24), which completes the proof. ■

4 Abstraction-based Construction of Safe-visor Architecture

4.1 Introduction

In this chapter, the abstraction-based construction of the Safe-visor architecture is proposed based on those abstraction-based controller synthesis schemes proposed in Chapter 3. First, a detailed description for the related work regarding sandboxing-like techniques and the verification over (AI-based) unverified controllers will be given. Then, the abstraction-based construction scheme of Safe-visor architecture over general Markov decision processes (gMDP) and general discrete-time stochastic games (gDTSGs) and will be elaborated in Section 4.2 and 4.3, respectively. Finally, the construction methodologies will be applied to three case studies in Section 4.4.

4.1.1 Related Works

In the setting of discrete-space systems, *e.g.,* reactive systems, the work in [31, 83, 24, 6] proposed the notion of *shield* to enforce some safety properties at runtime. For systems with continuous state and input sets, plenty of results are applicable to deterministic systems concerning simple invariance properties, in which systems are expected to stay within a fixed safety set. Concretely, reachability analysis-based techniques [65, 84] can be leveraged to provide safety guarantees by checking the intersection between the unsafe and reachable sets of the systems. Alternatively, a Simplex architecture was proposed in [170, 48] that enables the application of unverified, high-performance controllers in the control loop by using a Lyapunov-function-based elliptic recovery region (*a.k.a.* safety invariant set). This region is associated with a verified, linear, state-feedback controller, which serves as a high-assurance controller. Later, this architecture is further developed in [195, 196, 205, 3] to handle uncertainty and bounded-time delay in the system dynamics, as well as undetectable cyber attacks.

Note that in the Simplex architecture, the Lyapunov-function-based elliptic recovery region is usually very conservative and, therefore, unnecessarily restricts the use of high-performance controllers [14, Figure 7]. However, high-performance controllers are expected to be applied as often as possible. To provide more flexibility for high-performance controllers, results in [15, 14, 4] employ reachability analysis to enlarge the recovery region. Although some of these results are still called “Simplex design”, they are quite different from the basic idea of Simplex architecture as in [170]. In the Simplex architecture, high-assurance and high-performance controllers are both designed for the same tasks and are different in terms of the performance of finishing those tasks. By enlarging the recoverable region as in [15, 14, 4], the high-assurance controller is no longer able to finish the same tasks as the high-performance one but only ensures that the system would not leave the desired safety set, which are similarly to the safety advisor in the Safe-visor architecture. As mentioned, however, these results only work for deterministic systems and simple invariance properties.

Here, it is also worth mentioning some works related to providing safety guarantees over AI-based unverified controllers. Results in [101, 133, 182, 66, 172] improve the robustness of systems concerning safety in which AI-based controllers are applied, but they do not provide any formal safety guarantee. Meanwhile, results in [45, 51, 204, 82, 112, 65, 193, 194, 145, 79] provides formal safety guarantees for AI-based controllers regarding simple invariance properties; a few recent results also handle complex logical properties for systems with continuous state and input sets, e.g., [123] for deterministic systems, and [115, 100] for stochastic systems. In these results, formal guarantees are achieved by adequately cooperating the desired properties in the reward functions and guiding the learning process. However, these results are only applicable when reward functions are easy to be designed, while reward functions for some control tasks are difficult to be obtained (e.g., [28]). For those AI-based controllers designed based on deep-neural-networks-based (DNNs-based) techniques, various progresses has been made for their verification. Nevertheless, verifying DNNs is very challenging and it is an NP-complete problem [98]. Concretely, the challenges are due to the non-linear activation functions which makes DNNs non-linear and non-convex [57, 98]. Recent satisfiability modulo theory (SMT)-based approaches [57, 98, 99] and mixed-integer linear program (MILP) optimizers-based approaches [55] can be applied to check if adversarial perturbations over the inputs of the DNNs can change the decisions of the DNNs [98]. However, these approaches are typically applied on linearized input sets for simple DNNs with a few layers and a few hundred neurons per layer [84]. To reduce the complexities in verifying larger DNNs, [58, 156] propose DNNs abstraction approaches, with which one can obtain formal guarantees for the original complex DNNs by verifying the simplified DNNs using existing verification tools [99, 186]. Unfortunately, it is still difficult to deploy these methods to verify complex DNNs with millions of parameters and complicated architectures.

4.1.2 Contributions

With the abstraction-based controllers synthesis approaches in Chapter 3, an abstraction-based construction of the Safe-visor architecture as in Figure 1.1 is introduced in this chapter. The main contributions of the chapter are summarized as follows:

1. Considering safety specifications that can be expressed by the accepting languages of deterministic finite automata (DFA), the design of history-based supervisors in the Safe-visor architecture is proposed.

2. By employing (ϵ,δ)-approximate probabilistic relations proposed in [76] while designing the history-based supervisor, formal guarantees are provided for probabilities of satisfying desired specifications.

3. The synthesis approaches here allow constructing a reduced-order model for the original system when synthesizing the safety advisor and supervisor. As a result, these methodologies could be applied to large-scale systems, which is crucial since the complexity of synthesizing a safety advisor via a finite abstraction grows exponentially with respect to the dimension of the system's state set.

Additionally, compared with those results that adapt shielding approach in the learning process to ensure safety (e.g. [6, 45, 100], as discussed above in the related works), the proposed approaches decouple the desired safety properties from the reward functions in the learning process so that the provided safety guarantee is valid for AI-based controllers trained with arbitrary learning methods. Particularly, the proposed methods are applicable when the reward functions for the control tasks are challenging to be designed (e.g., [28]), while satisfying some safety properties are still required. Compared with those results that also decouple the desired safety properties from the synthesis of the unverified controllers (e.g., Simplex architecture in [196, 4], see detailed discussion above in the related works), the proposed results are the first to deal with stochastic CPSs with continuous state and input sets while ensuring safety properties modeled by DFAs. Although [31, 83, 24] also enforce safety specifications that are modeled by automata, they are only applicable to discrete-space systems.

4.2 Design of Safe-visor Architecture for General Markov Decision Processes

In this section, the construction of Safe-visor architecture over stochastic CPSs that can be modeled as general Markov decision processes (gMDPs), as introduced in Definition 2.4.1, would be discussed. In particular, the safety specifications of interested in this section are modeled by the accepting languages of deterministic finite automata (DFA), as introduced in Definition 2.5.1.

4.2.1 Problem Formulation

To formulate the problem of interest in this section, one needs the following definition of Markov policies for controlling gMDPs, which determine the input at the time instant k only based on the state at the same time instant, *i.e.,* $x(k)$.

Definition 4.2.1. (Markov Policy [171]) *Consider a gMDP* $\mathfrak{D} = (X, U, x_0, T, Y, h)$. *A Markov policy* ρ *is a sequence* $\rho = (\rho_0, \rho_1, \ldots)$ *of universally measurable maps* $\rho_k \colon X \to U$ *with*

$$\rho_k(U \,|\, \omega_k) = \rho_k(U \,|\, \omega_{xk}(k)) = 1,$$

for all $\omega_k \in \Omega$ *with* $k \in \mathbb{N}$. *Additionally,* $\mathcal{P}$ *denotes the set of all Markov policies, and* $\mathcal{P}^H$ *denotes the set of all Markov policies within time horizon* $[0, H-1]$.

Next, a more general set of control strategies are introduced, with which inputs are provided based on paths of the gMDP via a memory state. The definition here is adapted from [74] by allowing the memory and output update map to be time dependent.

Definition 4.2.2. (Control Strategy) *A control strategy for a gMDP* $\mathfrak{D} = (X, U, x_0, T, Y, h)$ *is a tuple*

$$\mathbf{C} = (\mathsf{M}, \mathsf{U}, \mathsf{Y}, \mathsf{H}, \mathsf{m}_0, \pi_{\mathsf{M}}, \pi_{\mathsf{Y}}),$$

where,

- M *is a Borel set as the* memory state set*;*
- $\mathsf{U} \subseteq \mathbb{R}^n$ *is a Borel set as the* observation set*;*
- $\mathsf{Y} \subseteq U$ *is a Borel set as the* output set*;*
- $\mathsf{H} \subseteq \mathbb{N}$ *is the* time domain*;*
- $\mathsf{m}_0 \in \mathsf{M}$ *is the* initial memory state*;*
- $\pi_{\mathsf{M}} : \mathsf{M} \times \mathsf{U} \times \mathsf{H} \to \mathbf{P}(\mathsf{M}, \mathcal{B}(\mathsf{M}))$ *is a* memory update function*;*
- $\pi_{\mathsf{Y}} : \mathsf{M} \times \mathsf{H} \to \mathbf{P}(\mathsf{Y}, \mathcal{B}(\mathsf{Y}))$ *is an* output update function*.*

Having Definition 4.2.2, given a gMDP $\mathfrak{D}$ and a control strategy $\mathbf{C}$, the controlled gMDP is denoted by $\mathbf{C} \times \mathfrak{D}$. Additionally, $\mathbb{P}_{\mathfrak{D}}$ (resp. $\mathbb{P}_{\mathbf{C}\times\mathfrak{D}}$) denotes the probability measure over the space of output sequences of $\mathfrak{D}$ (resp. $\mathbf{C} \times \mathfrak{D}$). Next, a measurable labeling function is introduced to show that how a gMDP $\mathfrak{D}$ as in (2.4.3) can be connected to a DFA $\mathcal{A}$.

Definition 4.2.3. (Labeling Function) *Given a gMDP* $\mathfrak{D} = (X, U, x_0, T, Y, h)$ *and a DFA* $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$*, a measurable labeling function* $L : Y \to \Pi$ *and a function* $L_H : Y^H \to \Pi^H$ *are defined as follows. Consider a finite output sequence* $y_{\omega(H-1)} = \big(y(0), y(1), \ldots, y(H-1)\big) \in Y^H$ *of* $\mathfrak{D}$ *with some* $H \in \mathbb{N}_{\geq 1}$*. The trace of* $y_{\omega(H-1)}$ *over* Π *is* $\sigma = L_H(y_{\omega(H-1)}) = (\sigma_0, \sigma_1, \ldots, \sigma_{H-1})$*, where* $\sigma_k = L(y(k))$ *for all* $k \in [0, H-1]$*. Moreover,* $y_{\omega(H-1)}$ *is accepted by* $\mathcal{A}$*, denoted by* $y_{\omega(H-1)} \models \mathcal{A}$*, if* $L_H(y_{\omega(H-1)}) \in \mathcal{L}(\mathcal{A})$*.*

Since the desired safety properties are modeled with DFA $\mathcal{A}$, here, the performance of the gMDP $\mathfrak{D}$ concerning this property is evaluated in terms of $\mathbb{P}_{\mathfrak{D}}\{y_{\omega(H-1)} \models \mathcal{A}\}$ within a bounded-time horizon. To this end, a product gMDP based on $\mathfrak{D}$ and $\mathcal{A}$ needs to be constructed, which is defined as follows.

Definition 4.2.4. (Product gMDP [74]) *Given a gMDP* $\mathfrak{D} = (X, U, x_0, T, Y, h)$*, a DFA* $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$*, and a labeling function* $L : Y \to \Pi$*, the product of* $\mathfrak{D}$ *and* $\mathcal{A}$ *is a gMDP and defined as*

$$\mathfrak{D} \otimes \mathcal{A} = \{\bar{X}, \bar{U}, \bar{x}_0, \bar{T}, \bar{Y}, \bar{h}\},$$

where

- $\bar{X} := X \times Q$ *is the state set;*

- $\bar{U} := U$ *is the input set;*
- $\bar{x}_0 := (x_0, \bar{q}_0)$ *is the initial state with*

$$\bar{q}_0 = \tau\big(q_0, L \circ h(x_0)\big); \tag{4.2.1}$$

- $\bar{T}(dx' \times \{q'\} \,|\, x, q, u)$ *is the stochastic kernel that assigns for any* $(x, q) \in \bar{X}$ *and* $u \in \bar{U}$*, the probability* $\bar{T}(dx' \times \{q'\}|x, q, u) := \mathbf{1}_{\{q^+\}}(q')T(dx'|x, u)$*, with* $q^+ = \tau(q, L \circ h(x'))$*, where* $\mathbf{1}_{Q'}(\cdot)$ *is the indicator function for a set* $Q' \subseteq Q$*, i.e., if* $q' \in Q'$*, then* $\mathbf{1}_{Q'}(q') = 1$*, otherwise* $\mathbf{1}_{Q'}(q') = 0$*;*
- $\bar{Y} := Y$ *is the output set;*
- $\bar{h}(x, q) := h(x)$ *is the output map.*

Having definitions above, I am ready to propose the main problem in Section 4.2. Concretely, two different problems according to different types of safety specifications are of interest. In the following discussion, η denotes the *maximal tolerable probability of violating the safety specification.* For some safety specifications, *e.g.,* those expressed as co-safe-LTL$_F$ properties [63], all infinite output sequences that satisfy this type of specifications have a finite good prefix. In this case, a DFA $\mathcal{A}$ that accepts all good prefixes is built to model the safety specifications of this kind. Accordingly, a problem of *robust satisfaction*, as defined follows, would be of interested.

Problem 4.2.5. (Robust Satisfaction) *Consider a gMDP* $\mathfrak{D}$ *as in* (2.4.3). *The* problem of robust satisfaction *with respect to the parameter* η *is to design a Safe-visor architecture as in Figure 1.1 for* $\mathfrak{D}$ *such that*

$$\mathbb{P}_{\mathfrak{D}}\Big\{y_{\omega H} \models \mathcal{A}\Big\} \geq 1 - \eta, \tag{4.2.2}$$

where $\mathcal{A}$ *is a DFA that accepts all good prefixes of the desired safety specification.*

Meanwhile, for some other safety specifications, *e.g.,* those expressed as safe-LTL$_F$ properties [166], all infinite output sequences that violate these specifications have a finite bad prefix. In this case, the specifications are modeled with a DFA $\mathcal{A}$ that accepts all bad prefixes. Correspondingly, a problem of *worst-case violation* is of interest, as defined below.

Problem 4.2.6. (Worst-case Violation) *Consider a gMDP* $\mathfrak{D}$ *as in* (2.4.3). *The* Problem of worst-case violation *with respect to the parameter* η *is to design a Safe-visor architecture as in Figure 1.1 for* $\mathfrak{D}$ *such that*

$$\mathbb{P}_{\mathfrak{D}}\Big\{y_{\omega H} \models \mathcal{A}\Big\} \leq \eta, \tag{4.2.3}$$

where $\mathcal{A}$ *is a DFA that accepts all bad prefixes of the desired safety specification.*

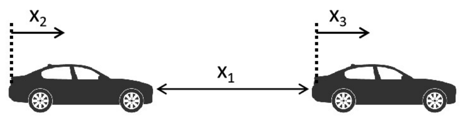


Figure 4.1: Running example: A system with two cars.

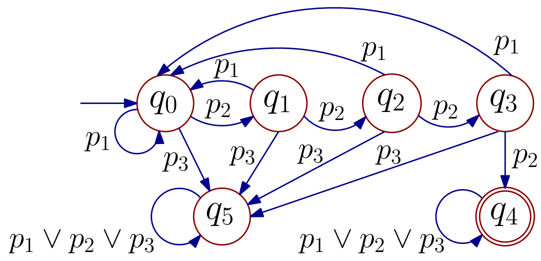


Figure 4.2: DFA modeling ψ, with accepting state q_4, alphabet $\Pi = \{p_1, p_2, p_3\}$ and labeling function $L : Y \to \Pi$, where $L(y) = p_1$ when $y \in (3, 10]$, $L(y) = p_2$ when $y \in [0, 3]$ and $L(y) = p_3$ when $y \in (-\inf, 0) \cup (10, +\inf)$.

It is also worth noting that a running example is provided here to demonstrate the theoretical results more intuitively.

Running example. In the running example, a system with two cars, which is adapted from [74], is deployed. This example is depicted in Figure 4.1 and can be modeled by the following set of stochastic difference equations:

$$\mathfrak{D} : \begin{cases} x(k+1) = Ax(k) + Bu(k) + R\varsigma(k), \\ y(k) = Cx(k), \end{cases} \qquad k \in \mathbb{N},$$

with

$$A = \begin{bmatrix} 1 & -0.15 & 0.15 \\ 0 & 0.6 & 0 \\ 0 & 0 & 0.6 \end{bmatrix}, \ B = [-0.03\,;1\,;0], \ R = [0.006\,;0\,;0.1], \ C = [1\,;0\,;0]^\top.$$

Here, $x(k) = [x_1(k); x_2(k); x_3(k)]$ is the state of the system, in which $x_1(k)$, $x_2(k)$, and $x_3(k)$ denote the distance between cars, and velocities of follower and leader cars, respectively. Input $u(k) \in [-8, 8]$ is the external actuation of the follower car. Besides, $\varsigma(k)$ is a sequence of standard Gaussian random variable that models the unpredictable changes in the leader car's velocity and in the distance between two cars, and $y(k)$ represents the output of the system. Here, the safety specification ψ requires: (i) within 8 time instances, the system output should reach $[0, 3]$ and then stay within $[0, 3]$ for 3 time instances after it reaches $[0, 3]$; (ii) the output should not exceed $[0, 10]$. The DFA modeling ψ is shown in Figure 4.2. Accordingly, the problem of robust satisfaction corresponding to this DFA should be considered.

4.2.2 Design of Safety Advisor

Consider a gMDP $\mathfrak{D}$ that models the original stochastic systems. To build the safety advisor, a finite abstraction of $\mathfrak{D}$, denoted by $\widehat{\mathfrak{D}}$, needs to be constructed using the results in Section 3.2.1. Accordingly, the (ϵ,δ)-approximate probabilistic relation in Definition 2.4.8 is used to quantify the similarity between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$, which is the key insight for providing a formal safety guarantee. The (ϵ,δ)-approximate probabilistic relation between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$ is established leveraging the results in Section 3.2.2 and 3.2.3.[1]

Remark 4.2.7. *In general, by reducing the quantization parameters (i.e. the size of the cells for constructing the finite abstraction, as discussed in Section 3.2.1) in partitioning the continuous state sets, a better safety guarantee can be provided by the safety advisor, but at the cost of increasing the execution time of the supervisor. This is shown with an example in Section 4.4.2.*

Running example (continued). By deploying the results in Section 3.2.1, a reduced-order version for the original model is first constructed with scalar state and input sets via balance truncation as implemented in `MATLAB`. This results in a reduced-order model

$$\widehat{\mathfrak{D}}_{\mathsf{r}}: \begin{cases} \hat{x}_{\mathsf{r}}(k+1) = \hat{A}_{\mathsf{r}}\hat{x}_{\mathsf{r}}(k) + \hat{B}_{\mathsf{r}}\hat{u}_{\mathsf{r}}(k) + \hat{R}_{\mathsf{r}}\varsigma(k), \\ \hat{y}_{\mathsf{r}}(k) = \hat{C}_{\mathsf{r}}\hat{x}_{\mathsf{r}}(k), \end{cases} \qquad k\in\mathbb{N},$$

where $\hat{A}_{\mathsf{r}} = 1$, $\hat{B}_{\mathsf{r}} = 0.3469$, and $\hat{C}_{\mathsf{r}} = 1$ (the index r signifies the reduced-order version of the original model), with a reduction matrix $P = [1\,;0.76\,;0]$. Additionally, $\hat{R}_{\mathsf{r}}$ would be selected later when establishing the approximate probabilistic relation. Then, a finite abstraction of the reduced-order model is built by considering $\hat{X}_{\mathsf{r}} = [0,10]$ as the region of interest for $\widehat{\mathfrak{D}}_{\mathsf{r}}$, and uniformly dividing this region into 100 partitions. Moreover, the input set is uniformly partitioned into cells whose lengths are 0.03. For establishing (ϵ,δ)-approximate probabilistic relation, $\hat{U}' = \{\hat{u}\in\hat{U} \mid -0.3 \le \hat{u} \le 0.3\}$ and $\delta = 0.01$ are chosen, and the expected range of ϵ is set as $[0.05,1]$. Then, the finite abstraction is (ϵ,δ)-stochastically simulated by original model with the relation $\mathscr{R} = \{(x,\hat{x}) | (x-P\hat{x})^\top M(x-P\hat{x}) \le \epsilon^2\}$ where

$$M = \begin{bmatrix} 2.4021 & -0.2239 & 0.2239 \\ -0.2239 & 0.03576 & -0.03576 \\ 0.2239 & -0.03576 & 0.03576 \end{bmatrix},$$

$\epsilon = 0.7984$, and the interface function is $\nu(x,\hat{x},\hat{u}) := K(x-P\hat{x}) + D\hat{x} + \tilde{R}\hat{u}$ with $K = [7.5764\,;-1.2399\,;1.2399]^\top$, $D = 0.1852$, and $\tilde{R} = 0.2530$. Moreover, $\hat{R}_{\mathsf{r}} = 0.0159$ is selected as in (3.2.36), and the lifting stochastic kernel for this relation is constructed such that the noise term in the original and finite systems are the same.

Having a finite abstraction $\widehat{\mathfrak{D}}$, which is (ϵ,δ)-stochastically simulated by $\mathfrak{D}$, a robust controller, denoted by $\tilde{\mathbf{C}}_\rho$, is designed by leveraging $\widehat{\mathfrak{D}}$ and used as the safety advisor in the Safe-visor architecture for the original gMDP. The design procedure and the running mechanism of the safety advisor is given in Figure 4.3. As shown in Figure 4.3 (left), a Markov policy ρ is first synthesized for the product gMDP $\widehat{\mathfrak{D}} \otimes \mathcal{A}$. Then, a

[1]Note that Section 3.2 focuses on building abstraction and establishing (ϵ,δ)-approximate probabilistic relation for stochastic systems with adversary input selected by rational agent. However, gMDP can be treated as a game without any adversary inputs (*i.e.,* one and a half player games) so that the method in Section 3.2 is still applicable to the problem of interest in this section.

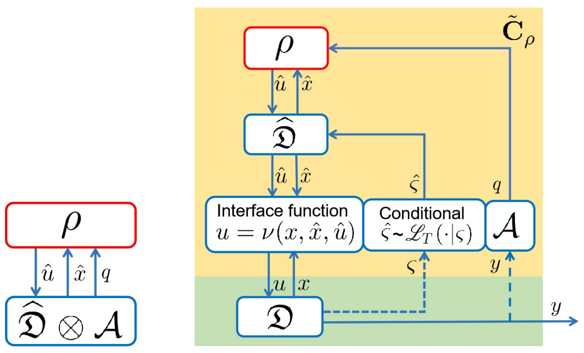

ρ
$\hat{u}$
$\hat{x}$
q
$\widehat{\mathfrak{D}} \otimes \mathcal{A}$
$\tilde{\mathbf{C}}_\rho$
$\widehat{\mathfrak{D}}$
$\hat{\varsigma}$
Interface function
$u = \nu(x, \hat{x}, \hat{u})$
Conditional
$\hat{\varsigma} \sim \mathscr{L}_T(\cdot|\varsigma)$
$\mathcal{A}$
ς
y
u
x
$\mathfrak{D}$

Figure 4.3: Left: Synthesizing Markov policy for $\widehat{\mathfrak{D}} \otimes \mathcal{A}$. **Right:** Construction of $\tilde{\mathbf{C}}_\rho$ (the yelow region).

control strategy $\tilde{\mathbf{C}}_\rho$ is constructed based on ρ as depicted in Figure 4.3 (right). In runtime, when a state x of $\mathfrak{D}$ is fed to $\tilde{\mathbf{C}}_\rho$:

1. State $\hat{x}$ of $\widehat{\mathfrak{D}}$ is first updated according to x and the conditional kernel $\mathscr{L}_T$. Then, the state q of $\mathcal{A}$ is updated according to the output function $h(x)$ of $\mathfrak{D}$ and transition function τ of $\mathcal{A}$;
2. $\hat{u}$ is provided by ρ based on $\hat{x}$ and q, and is refined to $\mathfrak{D}$ via the interface function ν as in (2.4.5).

Here, the construction of $\tilde{\mathbf{C}}_\rho$ is formally defined as follows.

Definition 4.2.8. (Construction of $\tilde{\mathbf{C}}_\rho$) *Given a Markov policy $\rho = (\rho_0, \rho_1, \ldots, \rho_{H-1})$ for $\widehat{\mathfrak{D}} \otimes \mathcal{A}$, a control strategy $\tilde{\mathbf{C}}_\rho = (\tilde{\mathsf{M}}, \tilde{\mathsf{U}}, \tilde{\mathsf{Y}}, \tilde{\mathsf{H}}, \tilde{\mathsf{m}}_0, \tilde{\pi}_{\mathsf{M}}, \tilde{\pi}_{\mathsf{Y}})$ is constructed for $\mathfrak{D}$, with*

- $\tilde{\mathsf{M}} = X \times \hat{X} \times Q$;
- $\tilde{\mathsf{U}} = X$;
- $\tilde{\mathsf{Y}} = U$;
- $\tilde{\mathsf{H}} = [0, H-1]$;
- $\tilde{\mathsf{m}}_0 = \big(\tilde{\mathsf{m}}_X(0), \tilde{\mathsf{m}}_{\hat{X}}(0), \tilde{\mathsf{m}}_Q(0)\big) \in \tilde{\mathsf{M}}$ *with* $\tilde{\mathsf{m}}_X(0) = x_0$, $\tilde{\mathsf{m}}_{\hat{X}}(0) = \hat{x}_0$ *and* $\tilde{\mathsf{m}}_Q(0) = \tau\big(q_0, L \circ h(\tilde{\mathsf{m}}_X(0))\big)$;
- $\tilde{\pi}_{\mathsf{M}}$ *updates* $\big(\tilde{\mathsf{m}}_X(k), \tilde{\mathsf{m}}_{\hat{X}}(k), \tilde{\mathsf{m}}_Q(k)\big) \in \tilde{\mathsf{M}}$ *at all time instant* $k \in \mathsf{H}\backslash\{0\}$ *with the following steps:*
  1. *update* $\tilde{\mathsf{m}}_{\hat{X}}(k)$ *according to the conditional stochastic kernel*
  $$\mathscr{L}_T\big(d\hat{x} \,\big|\, \tilde{\mathsf{m}}_{\hat{X}}(k-1), \tilde{\mathsf{m}}_X(k-1), x(k), \hat{u}(k-1)\big), \tag{4.2.4}$$
  with $x(k)$ which is the state of $\mathfrak{D}$ at time instant k and $\hat{u}(k-1) = \tilde{\rho}_k(\tilde{\mathsf{m}}_X(k-1), \tilde{\mathsf{m}}_{\hat{X}}(k-1), \tilde{\mathsf{m}}_Q(k-1))$;
  2. *update* $\tilde{\mathsf{m}}_X(k)$ *as* $\tilde{\mathsf{m}}_X(k) = x(k)$;
  3. *update* $\tilde{\mathsf{m}}_Q(k)$ *as* $\tilde{\mathsf{m}}_Q(k) = \tau\big(\tilde{\mathsf{m}}_Q(k-1), L \circ h(\tilde{\mathsf{m}}_X(k))\big)$;

- $\tilde{\pi}_{\mathsf{Y}}$ *updates* $\mathsf{y}(k) \in \mathsf{Y}$ *at time instant* $k \in \mathsf{H}$ *as* $\mathsf{y}(k) = \nu\big(\tilde{\mathsf{m}}_X(k), \tilde{\mathsf{m}}_{\hat{X}}(k), \rho_k(\tilde{\mathsf{m}}_{\hat{X}}(k), \tilde{\mathsf{m}}_Q(k))\big)$*, where* ν *is the interface function associated with the (*ϵ*-*δ*)-approximate probabilistic relation.*

Remark 4.2.9. *The conditional stochastic kernel* (4.2.4) *can be obtained by decomposing the lifting kernel* $\mathscr{L}_T(dx' \times d\hat{x}' \mid x, \hat{x}, \hat{u})$ *in Definition 2.4.8 as*

$$\mathscr{L}_T\big(dx'|x, \hat{x}, \hat{x}', \nu(x, \hat{x}, \hat{u})\big)\hat{T}(d\hat{x}'|\hat{x}, \hat{u}). \tag{4.2.5}$$

This decomposition is feasible according to [33, Corollary 3.1.2].

Before synthesizing ρ for $\widehat{\mathfrak{D}} \otimes \mathcal{A}$, the following assumption is required for the (ϵ,δ)-approximate probabilistic relation.

Assumption 4.2.10. *Consider gMDPs* $\mathfrak{D} = (X, U, x_0, T, Y, h)$ *and* $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{x}_0, \hat{T}, Y, \hat{h})$ *with* $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$ *as in Definition 2.4.8. It is assumed that*

$$\int_{\bar{\mathscr{R}}_{\hat{x}'}} \mathscr{L}_T\big(dx'|x, \hat{x}, \hat{x}', \nu(x, \hat{x}, \hat{u})\big) \geq 1 - \delta$$

holds $\forall \hat{x}, \hat{x}' \in \hat{X}$*, with* $\bar{\mathscr{R}}_{\hat{x}'} = \{x' \in X | (x', \hat{x}') \in \mathscr{R}\}$ *and* $\mathscr{L}_T(dx'|x, \hat{x}, \hat{x}', \nu(x, \hat{x}, \hat{u}))$ *being the conditional probability of* $x' \in X$ *given* x*,* $\hat{x}$*,* $\hat{x}'$ *and* $\nu(x, \hat{x}, \hat{u})$*.*

Similar to Assumption 3.3.2, Assumption 4.2.10 does not introduce extra subtlety in practice for establishing an (ϵ,δ)-approximate probabilistic relation between the original gMDP $\mathfrak{D}$ and its finite abstraction $\widehat{\mathfrak{D}}$. Now, I am ready to discuss how to synthesize a Markov policy ρ for the gMDP $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ concerning Problem 4.2.5 and 4.2.6, given an (ϵ, δ)-approximate probabilistic relation.

Problem of Robust Satisfaction. For the problem of robust satisfaction as in Problem 4.2.5, consider a gMDP $\mathfrak{D} = (X, U, x_0, T, Y, h)$ and its finite abstraction $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{x}_0, \hat{T}, Y, \hat{h})$, a DFA $\mathcal{A}$ that characterizes the desired safety specification and the product gMDP $\widehat{\mathfrak{D}} \otimes \mathcal{A} = \{\bar{X}, \bar{U}, \bar{x}_0, \bar{T}, \bar{Y}, \bar{h}\}$ as in Definition 4.2.4. Given a Markov policy $\rho = (\rho_0, \rho_1, \ldots, \rho_{H-1})$ defined over the time horizon $[0, H]$, a cost-to-go function $\bar{V}_n^\rho : \hat{X} \times Q \to [0, 1]$ is defined, which assigns a real number to states of $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ at time instant $H - n$. This function is initialized with $\bar{V}_0^\rho(\hat{x}, q) = 1$ when $q \in F$, and $\bar{V}_0^\rho(\hat{x}, q) = 0$ when $q \notin F$, and recursively compute $\bar{V}_{n+1}^\rho(\hat{x}, q)$ as

$$\bar{V}_{n+1}^\rho(\hat{x}, q) := \begin{cases} (1 - \delta) \displaystyle\sum_{\hat{x}' \in \hat{X}} \bar{V}_n^\rho\big(\hat{x}', \underline{q}(\hat{x}', q)\big)\hat{T}\big(\hat{x}' \mid \hat{x}, \rho_{H-n-1}(\hat{x}, q)\big), & \text{if } q \notin F; \\ 1, & \text{if } q \in F, \end{cases} \tag{4.2.6}$$

with

$$\underline{q}(\hat{x}', q) := \arg\min_{q' \in Q'_\epsilon(\hat{x}')} \bar{V}_n^\rho(\hat{x}', q'), \tag{4.2.7}$$

where

$$Q'_\epsilon(\hat{x}') := \big\{q' \in Q \,\big|\, \exists x \in X \, s.t. \, q' = \tau(q, L \circ h(x)) \text{ with } h(x) \in N_\epsilon(\hat{h}(\hat{x}'))\big\}, \tag{4.2.8}$$

and

$$N_\epsilon(\hat{y}) := \{y \in Y \mid \|y - \hat{y}\| \leq \epsilon\}. \tag{4.2.9}$$

With this notion, the following theorem provides the safety guarantee associated with ρ for the problem of robust satisfaction.

Theorem 4.2.11. *Consider gMDPs $\mathfrak{D} = (X, U, x_0, T, Y, h)$ and $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{x}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$, and a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ characterizing the desired safety property. For any Markov policy ρ designed for $\widehat{\mathfrak{D}} \times \mathcal{A}$ and a control strategy $\tilde{\mathbf{C}}_\rho$ of $\mathfrak{D}$ constructed based on ρ as in Definition 4.2.8, one obtains*

$$\mathbb{P}_{\tilde{\mathbf{C}}_\rho \times \mathfrak{D}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\} \geq \bar{V}_H^\rho(\hat{x}_0, \bar{q}_0), \tag{4.2.10}$$

with $y_{\omega H}$ being output sequences of $\mathfrak{D}$ up to the time instant H, $\bar{V}_H^\rho(\hat{x}_0, \bar{q}_0)$ computed as in (4.2.6), and $\bar{q}_0 = \tau(q_0, L \circ h(x_0))$.

Theorem 4.2.11 is adapted from Theorem 3.3.4 with some modification and can therefore be proved similarly. Since the safety advisor is responsible for maximizing the safety probability, a $\rho \in \mathcal{P}^H$ that maximizes $\bar{V}_H^\rho(\hat{x}_0, \bar{q}_0)$ as in (4.2.10) is of particular interest. The following proposition shows how such a Markov policy can be synthesized.

Proposition 4.2.12. *Consider gMDPs $\mathfrak{D} = (X, U, x_0, T, Y, h)$ and $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{x}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$, and a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$. The Markov policy $\rho^* = (\rho_0^*, \rho_1^*, \ldots, \rho_{H-1}^*)$ maximizes $\bar{V}_H^\rho(\hat{x}_0, \bar{q}_0)$ as in (4.2.10), with*

$$\rho_{H-n-1}^* \in \arg\max_{\rho_{H-n-1} \in \mathcal{P}} (1-\delta) \sum_{\hat{x}' \in \hat{X}} \bar{V}_n^*\big(\hat{x}', \underline{q}^*(\hat{x}', q)\big) \hat{T}\big(\hat{x}' \,\big|\, \hat{x}, \rho_{H-n-1}(\hat{x}, q)\big), \tag{4.2.11}$$

for all $n \in [0, H-1]$.

In Proposition 4.2.12, $\bar{V}_n^*(\hat{x}, q)$ is the cost-to-go function associated with ρ^*, and this function can be recursively computed as

$$\bar{V}_{n+1}^*(\hat{x}, q) := \begin{cases} \max\limits_{\rho_{H-n-1} \in \mathcal{P}} (1-\delta) \sum\limits_{\hat{x}' \in \hat{X}} \bar{V}_n^*\big(\hat{x}', \underline{q}^*(\hat{x}', q)\big) \hat{T}\big(\hat{x}' \big| \hat{x}, \rho_{H-n-1}(\hat{x}, q)\big), & \text{if } q \notin F; \\ 1, & \text{if } q \in F, \end{cases} \tag{4.2.12}$$

initialized by $\bar{V}_0^*(\hat{x}, q) = 1$, when $q \in F$, and $\bar{V}_0^*(\hat{x}_0, q) = 0$ otherwise, where

$$\underline{q}^*(\hat{x}', q) := \arg\min_{q' \in Q'_\epsilon(\hat{x}')} \bar{V}_n^*(\hat{x}', q'), \tag{4.2.13}$$

and $Q'_\epsilon(\hat{x}')$ is the set as in (4.2.8).

Problem of Worst-case Violation. For the problem of worst-case violation as in Problem 4.2.6, consider a Markov policy $\rho = (\rho_0, \rho_1, \dots, \rho_{H-1})$ defined over the time horizon $[0, H]$. A cost-to go function $\underline{V}^{\rho}_n : \hat{X} \times Q \to [0, 1]$ is defined, which maps each state of $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ at the time instant $H - n$ to a real number. Here, $\underline{V}^{\rho}_{n+1}(x, q)$ is recursively computed as

$$\underline{V}^{\rho}_{n+1}(\hat{x}, q) := \begin{cases} (1-\delta)\sum_{\hat{x}' \in \hat{X}} \underline{V}^{\rho}_n\big(\hat{x}', \bar{q}(\hat{x}', q)\big)\hat{T}\big(\hat{x}' \big| \hat{x}, \rho_{H-n-1}(\hat{x}, q)\big) + \delta, & \text{if } q \notin F; \\ 1, & \text{if } q \in F, \end{cases} \tag{4.2.14}$$

initialized by $\underline{V}^{\rho}_0(\hat{x}, q) = 1$, when $q \in F$, and $\underline{V}^{\rho}_0(\hat{x}, q) = 0$ otherwise, where

$$\bar{q}(\hat{x}', q) := \arg\max_{q' \in Q'_{\epsilon}(\hat{x}')} \underline{V}^{\rho}_n(\hat{x}', q'), \tag{4.2.15}$$

and $Q'_{\epsilon}(\hat{x}')$ is the set as in (4.2.8). Similar to Theorem 4.2.11, next theorem provides the safety guarantee associated with ρ for the problem of worst-case violation.

Theorem 4.2.13. *Consider gMDPs $\mathfrak{D} = (X, U, x_0, T, Y, h)$ and $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{x}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$, and a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ characterizing the desired safety property. For any Markov policy ρ designed for $\widehat{\mathfrak{D}} \times \mathcal{A}$ and a control strategy $\tilde{\mathbf{C}}_{\rho}$ of $\mathfrak{D}$ constructed based on ρ as in Definition 4.2.8, one obtains*

$$\mathbb{P}_{\tilde{\mathbf{C}}_{\rho} \times \mathfrak{D}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\} \leq \underline{V}^{\rho}_H(\hat{x}_0, \bar{q}_0), \tag{4.2.16}$$

where $y_{\omega H}$ is the output sequences of $\mathfrak{D}$ up to the time instant H, $\underline{V}^{\rho}_H(\hat{x}_0, \bar{q}_0)$ is computed as in (4.2.14)*, and $\bar{q}_0 = \tau(q_0, L \circ h(x_0))$.*

Theorem 4.2.13 is adapted from Theorem 3.3.10 with some modification and so that it can be proved in a similar way. As for the problem of worst-case violation, the safety advisor is responsible for minimizing the probability of violating the desired safety specifications. Thus, a $\rho \in \mathcal{P}^H$ that minimizes $\underline{V}^{\rho}_H(\hat{x}_0, \bar{q}_0)$ in (4.2.16) is of particular interest. The following proposition shows how to synthesize such a Markov policy.

Proposition 4.2.14. *Consider gMDPs $\mathfrak{D} = (X, U, x_0, T, Y, h)$ and $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{x}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$, and a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$. The Markoc policy $\rho_* = (\rho_{*0}, \rho_{*1}, \dots, \rho_{*H-1})$ minimizes $\underline{V}^{\rho}_H(\hat{x}_0, \bar{q}_0)$ in* (4.2.16)*, with*

$$\rho_{*H-n-1} \in \arg\min_{\rho_{H-n-1} \in \mathcal{P}} \Big((1-\delta)\sum_{\hat{x}' \in \hat{X}} \underline{V}_{*,n}\big(\hat{x}', \bar{q}_*(\hat{x}', q)\big)\hat{T}\big(\hat{x}' \,\big|\, \hat{x}, \rho_{H-n-1}(\hat{x}, q)\big) + \delta\Big), \tag{4.2.17}$$

for all $n \in [0, H-1]$.

In Proposition 4.2.14, $\underline{V}_{*,n}(\hat{x}, q)$ denotes the cost-to-go function associated with ρ_*. This function is initialized with $\underline{V}_{*,0}(\hat{x}, q) = 1$, when $q \in F$, $\underline{V}_{*,0}(\hat{x}, q) = 0$, when $q \notin F$, and recursively compute it as

$$\underline{V}_{*,n+1}(\hat{x}, q) := \begin{cases} \min\limits_{\rho_{H-n-1} \in \mathcal{P}} \Big((1-\delta)\sum\limits_{\hat{x}' \in \hat{X}} \underline{V}_{*,n}\big(\hat{x}', \bar{q}_*(\hat{x}', q)\big)\hat{T}\big(\hat{x}' \,\big|\, \hat{x}, \rho_{H-n-1}(\hat{x}, q)\big) + \delta\Big), & \text{if } q \notin F; \\ 1, & \text{if } q \in F, \end{cases} \tag{4.2.18}$$

where

$$\bar{q}_*(\hat{x}', q) := \arg\max_{q' \in Q'_\epsilon(\hat{x}')} \underline{V}_{*,n}(\hat{x}', q'), \tag{4.2.19}$$

and $Q'_\epsilon(\hat{x}')$ is a set as in (4.2.8).

Finally, the construction of safety advisor is summarized as follows:

- For the problem of robust satisfaction, a Markov policy ρ^* is synthesized as in (4.2.11). Then, control strategy $\tilde{\mathbf{C}}_{\rho^*}$ is constructed as in Definition 4.2.8 based on ρ^* and apply $\tilde{\mathbf{C}}_{\rho^*}$ as the safety advisor.
- As for the problem of worst-case violation, a control strategy $\tilde{\mathbf{C}}_{\rho_*}$ is constructed as in Definition 4.2.8 based on a Markov policy ρ_* synthesized as in (4.2.17), and employ $\tilde{\mathbf{C}}_{\rho_*}$ as the safety advisor.

Since both ρ^* and ρ_* are look-up tables that are computed offline, the safety advisor can be applied efficiently at runtime.

4.2.3 Design of Supervisor

In general, the proposed supervisor takes two steps to decide whether to accept the input from the unverified controller, denoted by $u_{uc}(k)$, at every time instant k:

- Step 1: Check whether the (ϵ,δ)-approximate probabilistic relation will still hold between the finite abstraction and the original system presuming that $u_{uc}(k)$ is accepted. If the relation will not hold, reject $u_{uc}(k)$ without going through Step 2 and feed input provided by the safety advisor, denoted by u_{safe}, to the system $\mathfrak{D}$.
- Step 2: Estimate the probability of violating the desired safety specification, denoted by $\mathcal{E}_{pv}(k)$, presuming that $u_{uc}(k)$ is accepted. Accept $u_{uc}(k)$ only if $\mathcal{E}_{pv}(k) \leq \eta$; otherwise, feed u_{safe} to the system $\mathfrak{D}$.

Here, the Safe-visor architecture with the safety-advisor proposed in Section 4.2.2 and the Supervisor in this subsection is illustrated in Figure 4.4 (left). Different from Figure 4.3 (right), $\hat{u}'$ that is fed to the abstraction $\widehat{\mathfrak{D}}$ is decided by the Supervisor, instead of ρ. Concretely, if $u_{uc}(k)$ is rejected, one has $\hat{u}' = \hat{u}$, with $\hat{u}$ being the input

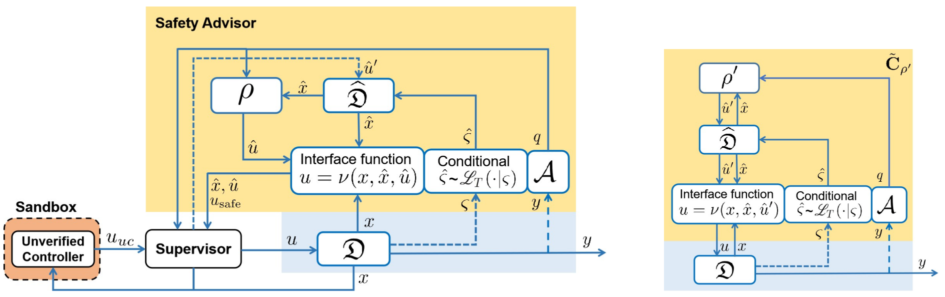


Figure 4.4: Left: Safe-visor architecture designed based on an (ϵ,δ)-approximate probabilistic relation. **Right:** Controlled gMDP $\tilde{C}_{\rho'} \times \mathfrak{D}$ that is equivalent to $\mathfrak{D}$ operating in the Safe-visor architecture.

provided by ρ; otherwise, one need to select $\hat{u}' \in \hat{U}'$, with $\hat{U}'$ being the input set of $\widehat{\mathfrak{D}}$ as discussed in Section 3.2.3 (the selection of such $\hat{u}'$ would be elaborated later). Step 1 ensures that $u_{uc}(k)$ would only be accepted if there exists $\hat{u}'$ such that the safety guarantee based on the (ϵ, δ)-approximate probabilistic relation still holds. Checking Step 1 can be performed by virtue of the following proposition.

Proposition 4.2.15. *Consider a gMDP $\mathfrak{D} = (X, U, x_0, T, Y, h)$ and its finite abstraction $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{x}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$. If $u_{uc}(k)$ is applied to $\mathfrak{D}$ at the time instant k, the (ϵ,δ)-approximate probabilistic relation still holds between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$ at the time instant $k+1$ if the set*

$$U_f = \Big\{ \hat{u} \in \hat{U}' \,\Big|\, \mathbb{P}\big\{(x', \hat{x}') \in \mathscr{R}\big\} \geq 1 - \delta, \text{ with}$$
$$x' = f(x(k), u_{uc}(k), \varsigma(k)), \hat{x}' = \hat{f}(\hat{x}(k), \hat{u}, \hat{\varsigma}(k)) \Big\}, \tag{4.2.20}$$

is not empty, with $\hat{U}'$ being the input set for $\widehat{\mathfrak{D}}$. Here, f and $\hat{f}$ are transition maps of $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$ as in (2.4.2), respectively. Moreover, $x(k) \in X$, $\hat{x}(k) \in \hat{X}$, $\varsigma(k)$ and $\hat{\varsigma}(k)$ are current states of $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$, noise affecting $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$, respectively.

Intuitively, the non-emptiness of U_f ensures that there exists at least one $\hat{u} \in \hat{U}'$ corresponding to $u_{uc}(k)$ such that $(x(k+1), \hat{x}(k+1)) \in \mathscr{R}$ holds with the probability of at least $1 - \delta$. Accordingly, if U_f is not empty, and $u_{uc}(k)$ is accepted after going through Step 2, one needs to use $\hat{u} \in U_f$ for $\widehat{\mathfrak{D}}$. The selection of $\hat{u}$ is related to the estimation $\mathcal{E}_{pv}(k)$ in Step 2 which is discussed later. Prior to discussing how to obtain $\mathcal{E}_{pv}(k)$, I show how to check whether U_f is empty with the help of Proposition 4.2.15 through the running example.

Running example (continued). Consider the current state $x(k)$ of the original model, $\hat{x}(k)$ of the finite abstraction, and input $u_{uc}(k)$ provided by the unverified controller. To show whether U_f for the running example is empty, one needs to show whether there exists $\hat{u} \in \hat{U}'$ such that

$$\|Ax(k) + Bu_{uc}(k) + R\varsigma(k) - P\big(\hat{A}_{\mathsf{r}}\hat{x}(k) + \hat{B}_{\mathsf{r}}\hat{u} + \hat{R}_{\mathsf{r}}\hat{\varsigma}(k)\big)\|_M \leq \epsilon \tag{4.2.21}$$

holds for all $\|\varsigma\| \leq \chi_1^{-1}(1-\delta)$ with $\delta = 0.01$, $\hat{\varsigma}(k) = \varsigma(k)$, $\|\bar{x}\|_M = \sqrt{\bar{x}^\top M \bar{x}}$, and $\chi_1^{-1}(\cdot)$ being the chi-square inverse cumulative distribution function with one degree of freedom. Note that (4.2.21) holds when

$$\|\varphi - P\hat{B}_{\mathsf{r}}\hat{u}\|_M \leq \epsilon - \gamma, \tag{4.2.22}$$

with $\varphi = Ax(k) + Bu_{uc}(k) - P\hat{A}_{\mathsf{r}}\hat{x}_{\mathsf{r}}(k)$ and

$$\gamma = \max_{\|\varsigma\| \leq \chi_1^{-1}(0.99), \beta \in \Delta} \|(R - P\hat{R}_{\mathsf{r}})\varsigma + P\beta\|_M.$$

Here, Δ denotes the set of all possible quantization errors introduced by discretization of the original state set as in (3.2.11). Note that $\hat{U}'$ contains a finite number of $\hat{u}$ (usually not too large in practice), φ can be readily computed at runtime, and γ can be computed offline after constructing the finite abstraction. Therefore, one can find out whether there exists $\hat{u} \in \hat{U}'$ such that (4.2.22) holds at runtime efficiently.

Next, I proceed with discussing how to obtain $\mathcal{E}_{pv}(k)$ in Step 2. First, the Safe-visor architecture as in Figure 4.4 (left) can be equivalently described by a controlled gMDP $\tilde{\mathbf{C}}_{\rho'} \times \mathfrak{D}$ as in Figure 4.4 (right) according to its running mechanism. Here, $\tilde{\mathbf{C}}_{\rho'}$ (the yellow region) is a control strategy constructed as in Definition 4.2.8 based on a Markov policy ρ', which differs from ρ^* as in (4.2.11) (for problem of robust satisfaction) or ρ_* as in (4.2.17) (for problem of worst-case violation) due to the acceptance of unverified controllers. Consider the maximal tolerable probability of violating the desired safety specification (*i.e.,* η) within a time horizon $[0, H]$. For the problem of robust satisfaction, if the supervisor is designed such that one has

$$1 - \bar{V}_H^{\rho'}(\hat{x}_0, \bar{q}_0) \leq \eta, \tag{4.2.23}$$

Then, (4.2.2) can be ensured by combining (4.2.23) and Theorem 4.2.11. Similarly, if the supervisor is designed for the problem of worst-case violation such that

$$\underline{V}_H^{\rho'}(\hat{x}_0, \bar{q}_0) \leq \eta, \tag{4.2.24}$$

one can guarantee (4.2.3) by considering (4.2.24) and Theorem 4.2.13.[2] Thus, a direct idea is to set $\mathcal{E}_{pv}(k) = 1 - \bar{V}_H^{\rho'}(\hat{x}_0, \bar{q}_0)$ for the problem of robust satisfaction and $\mathcal{E}_{pv}(k) = \underline{V}_H^{\rho'}(\hat{x}_0, \bar{q}_0)$ for the problem of worst-case violation. Note that when the initial state $(\hat{x}_0, \bar{q}_0)$ and the time horizon $[0, H]$ are fixed, $\bar{V}_H^{\rho}(\hat{x}_0 . \bar{q}_0)$ and $\underline{V}_H^{\rho}(\hat{x}_0 . \bar{q}_0)$ can be computed given ρ'. The remaining problem is how to determine ρ' at runtime.

In general, determining ρ' at runtime is very challenging. At each time instant $k \in [0, H-2]$ in each individual execution, ρ'_z are unknown for all time instant $z \in (k, H-1]$

[2]In both (4.2.23) and (4.2.24), one only needs to focus on the case in which $\bar{q}_0 \notin F$, since $\bar{q}_0 \in F$ indicates that the accepting states are reached at the initial state, which is not of interest.

since the supervisor does not know which inputs the unverified controller will provide in the future. Moreover, the supervisor does not have complete information about ρ'_z for all $z \in [0,k]$, either. In each individual execution, the system only reaches one state at each time step. Therefore, the supervisor does not know what inputs the unverified controller would offer if the system reached $\hat{X} \times Q \backslash \{(\hat{x}(z), q(z))\}$ at time instants $z \in [0,k]$, let alone whether they would be accepted and what $\hat{u}'$ would accordingly be fed to $\widehat{\mathfrak{D}}$. As a result, $\rho'_z(\tilde{x},\tilde{q})$ is unknown for all $(\tilde{x},\tilde{q}) \in \hat{X} \times Q \backslash \{(\hat{x}(z), q(z))\}$, with $\hat{x}(z)$ and $q(z)$ being the state of $\widehat{\mathfrak{D}}$ and $\mathcal{A}$ at time instant $z \in [0,k]$, respectively.

To cope with this issue, instead of determining ρ' and computing $\bar{V}^{\rho'}_H(\hat{x}_0, \bar{q}_0)$ or $\underline{V}^{\rho'}_H(\hat{x}_0, \bar{q}_0)$ at runtime, a *history-based supervisor* is proposed to provide an estimation of $\mathcal{E}_{pv}(k)$ without requiring to know ρ' at runtime. This estimation is given with the help of history paths $\bar{\omega}_k$ of the Safe-visor architecture as in Figure 4.4 (left), which is defined below.

Definition 4.2.16. (History Path of Safe-visor Architecture) *Consider a finite abstraction $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{x}_0, \hat{T}, Y, \hat{h})$, a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ characterizing the safety property of interest and the corresponding Safe-visor architecture as in Figure 4.4 (left). The history path of the Safe-visor architecture up to the time instant k is denoted by*

$$\bar{\omega}_k = \Big(\hat{x}(0), q(0), \hat{u}(0),\, \hat{x}(1), q(1), \hat{u}(1), \ldots, \hat{x}(k-1), q(k-1), \hat{u}(k-1), \hat{x}(k), q(k)\Big),$$

where $\hat{x}(k) \in \hat{X}$, $q(k) \in Q$, and $\hat{u}(k) \in \hat{U}$ are states of $\widehat{\mathfrak{D}}$, $\mathcal{A}$, and the input fed to $\widehat{\mathfrak{D}}$ at the time instant k, respectively. Moreover, $\bar{\omega}_{\hat{x}k}$, $\bar{\omega}_{qk}$, and $\bar{\omega}_{uk}$ denote the subpath of $\hat{x}$, q, and $\hat{u}$ corresponding to $\bar{\omega}_k$, respectively.

In general, *history-based supervisor* provides $\mathcal{E}_{pv}(k)$ such that (4.2.23) or (4.2.24) can be respected. Consider a gMDP $\mathfrak{D} = (X, U, x_0, T, Y, h)$ and its finite abstraction $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{x}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$, a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$, a labeling function $L : Y \to \Pi$ associated with $\mathcal{A}$ as in Definition 4.2.3, and η to be the maximal tolerable probability of violating the desired safety specification. I proceed with showing how to design such a supervisor for both problems of interest.

Supervisor for the Problem of Robust Satisfaction. The design of supervisor for the problem of robust satisfaction is formally proposed as follows.

Definition 4.2.17. (History-based Supervisor for the Problem of Robust Satisfaction) *At each time instant $k \in [0, H-1]$, given the history path $\bar{\omega}_k$ as in Definition 4.2.16, the feasibility of an input $u_{uc}(k)$ from the unverified controller is checked by the following two steps:*

i) Check set U_f as in (4.2.20) *and reject $u_{uc}(k)$ if U_f is empty;*

ii) If U_f is not empty, estimate $\mathcal{E}_{pv}(k)$ as

$$\mathcal{E}_{pv}(k) = \bar{\mathcal{E}}_{pv}(k) = \prod_{z=1}^{k} \Big((1-\delta) \sum_{\hat{x} \in \hat{X}'_{\epsilon}(q(z-1))} \hat{T}(\hat{x} \,|\, \hat{x}(z-1), \hat{u}(z-1)) \Big)$$

$$\times (1-\delta)\Big(1-\sum_{\hat{x}\in\hat{X}} \bar{V}^*_{H-k-1}\big(\hat{x},\underline{q}^*(\hat{x}(k),q(k))\big)\hat{T}(\hat{x}\,|\,\hat{x}(k),\hat{u}^*)\Big)$$

$$+\,\delta+\sum_{j=1}^{k}\Big(\delta\prod_{z=1}^{j}\big((1-\delta)\sum_{\hat{x}\in\hat{X}'_\epsilon(q(z-1))}\hat{T}(\hat{x}\,|\,\hat{x}(z-1),\hat{u}(z-1))\big)\Big), \tag{4.2.25}$$

with $\bar{V}^*_{H-k-1}$ *as in* (4.2.12) *associated with the safety advisor,* $\underline{q}^*$ *as in* (4.2.13),

$$\hat{u}^* = arg\max_{\hat{u}\in U_f}\bar{\mathcal{E}}_{pv}(k), \tag{4.2.26}$$

and

$$\hat{X}'_\epsilon(q(z-1)) := \Big\{\hat{x}\in\hat{X}\,\big|\,\exists x\in X, \tau(q(z-1), L\circ h(x))\notin F, h(x)\in N_\epsilon(\hat{h}(\hat{x}))\Big\}, \tag{4.2.27}$$

where $N_\epsilon(\hat{h}(\hat{x}))$ *is as in* (4.2.9). *If* $\mathcal{E}_{pv}(k)\le\eta$*, the supervisor accepts* $u_{uc}(k)$ *and feeds* $\hat{u}'=\hat{u}^*$ *as in* (4.2.26) *to* $\widehat{\mathfrak{D}}$*; otherwise, it rejects* $u_{uc}(k)$ *and feeds* $\hat{u}'=\hat{u}$ *to* $\widehat{\mathfrak{D}}$*, with* $\hat{u}$ *provided by* ρ^*_k *as in* (4.2.11) *associated with the safety advisor.*

By applying the history-based supervisor as in Definition 4.2.17, (4.2.23) can be guaranteed so that the problem of robust satisfaction can be solved. This is formally formulated in the next theorem.

Theorem 4.2.18. *Consider a gMDP* $\mathfrak{D}=(X,U,x_0,T,Y,h)$ *and a DFA* $\mathcal{A}=(Q,q_0,\Pi,\tau,F)$ *that characterizes the desired safety specification. Employing the supervisor as in Definition 4.2.17 at all time instants* $k\in[0,H-1]$ *in the Safe-visor architecture, one has*

$$\mathbb{P}_{\mathfrak{D}}\Big\{y_{\omega H}\models\mathcal{A}\Big\}\ge 1-\eta, \tag{4.2.28}$$

for the problem of robust satisfaction as in Problem 4.2.5, with $y_{\omega H}$ *being output sequences of* $\mathfrak{D}$ *up to the time instant* H.

The proof of Theorem 4.2.18 is provided in Section 4.6. Here are some intuition for different parts in $\mathcal{E}_{pv}(k)$ as in (4.2.25) and the complexity for computing them at runtime. Generally, the estimation of $\mathcal{E}_{pv}(k)$ in (4.2.25) can be divided into three parts:

- Part 1: $\prod_{z=1}^{k}\big((1-\delta)\sum_{\hat{x}\in\hat{X}'_\epsilon(q(z-1))}\hat{T}(\hat{x}\,|\,\hat{x}(z-1),\hat{u}(z-1))\big)$:
  Given the history path $\bar{\omega}_k$, Part 1 denotes the maximal probability of $\mathfrak{D}$ not being accepted by $\mathcal{A}$ within the time horizon $[0,k]$, while $x(z)\mathscr{R}\hat{x}(z)$ holds for all $z\in[0,k]$. As defined in (4.2.27), for all $\hat{x}\in\hat{X}'_\epsilon(q(z-1))$ with $q(z-1)\in Q$, there exits at least one x with $x\mathscr{R}\hat{x}$, such that F is not reachable with $x(z)=x$. In another word, if $\hat{x}(z)\in\hat{X}'_\epsilon(q(z-1))$, $\mathfrak{D}$ may not be accepted by $\mathcal{A}$ even when one has $x(z)\mathscr{R}\hat{x}(z)$. Therefore, given $\hat{x}(z-1)$ and $\hat{u}(z-1)$, $(1-\delta)\sum_{\hat{x}\in\hat{X}'_\epsilon(q(z-1))}\hat{T}(\hat{x}\,|\,\hat{x}(z-1),\hat{u}(z-1))$ denotes the maximal probability of $\mathfrak{D}$ not being accepted by $\mathcal{A}$ at the

time instant z while $x(z)\mathscr{R}\hat{x}(z)$ still holds. As for the complexity of computing Part 1 at runtime, one has

$$\prod_{z=1}^{k}\Big((1-\delta)\sum_{\hat{x}\in\hat{X}'_{\epsilon}(q(z-1))}\hat{T}(\hat{x}\,|\,\hat{x}(z-1),\hat{u}(z-1))\Big)$$
$$=\underbrace{\prod_{z=1}^{k-1}\Big((1-\delta)\sum_{\hat{x}\in\hat{X}'_{\epsilon}(q(z-1))}\hat{T}(\hat{x}\big|\hat{x}(z-1),\hat{u}(z-1))\Big)}_{\text{Term 1}}\times\underbrace{(1-\delta)\sum_{\hat{x}\in\hat{X}'_{\epsilon}(q(k-1))}\hat{T}(\hat{x}\big|\hat{x}(k-1),\hat{u}(k-1))}_{\text{Term 2}}.$$

On one hand, $\forall q\in Q$, set $\hat{X}'_{\epsilon}(q)$ can be computed offline, and $\hat{T}$ is readily computed when synthesizing the safety advisor. On the other hand, at the time instant $k-1$, Term 1 has already been computed at time instant $k-2$. Therefore, the number of operations required for computing Part 1 at time instant $k-1$ is proportional to the number of states in the set $\hat{X}'_{\epsilon}(q(k-1))$.

- Part 2: $(1-\delta)\big(1-\sum_{\hat{x}\in\hat{X}}\bar{V}^{*}_{H-k-1}\big(\hat{x},\underline{q}^{*}(\hat{x}(k),q(k))\big)\hat{T}(\hat{x}\,|\,\hat{x}(k),\hat{u}^{*})\big)$:
Part 2 quantifies the probability of $\mathfrak{D}$ not being accepted by $\mathcal{A}$ within time horizon $[k+1,H]$, while (i) $x(k+1)\mathscr{R}\hat{x}(k+1)$ holds, given $\hat{x}(k)$, $q(k)$, and $\hat{u}(k)=\hat{u}^{*}$; (ii) $\mathfrak{D}$ is controlled by the safety advisor within $[k+1,H]$. Since $\bar{V}^{*}_{H-k-1}$ and $\hat{T}$ are readily computed when synthesizing the safety advisor, the number of operations required for computing Part 2 is proportional to the number of elements in sets $\hat{X}$ and U_f.
- Part 3: $\delta+\sum_{j=1}^{k}\big(\delta\prod_{z=1}^{j}\big((1-\delta)\sum_{\hat{x}\in\hat{X}'_{\epsilon}(q(z-1))}\hat{T}(\hat{x}\,|\,\hat{x}(z-1),\hat{u}(z-1))\big)\big)$:
Given the history path $\bar{\omega}_k$, Part 3 quantifies the maximal probability of $\mathfrak{D}$ not being accepted by $\mathcal{A}$ within the time horizon $[0,k]$ while $\exists z\in[1,k+1]$ such that $x(z)\mathscr{R}\hat{x}(z)$ does not hold. Concretely, the first δ in Part 3 quantifies $\mathbb{P}\{(x(1),\hat{x}(1))\notin\mathscr{R}\mid(x(0),\hat{x}(0))\in\mathscr{R}\}$. Moreover, the term
$$\delta\prod_{z=1}^{j}\big((1-\delta)\sum_{\hat{x}\in\hat{X}'_{\epsilon}(q(z-1))}\hat{T}(\hat{x}\,|\,\hat{x}(z-1),\hat{u}(z-1))\big)$$
represents the maximal probability of $\mathfrak{D}$ not being accepted by $\mathcal{A}$ within the time horizon $[0,j+1]$, while (i) $x(z)\mathscr{R}\hat{x}(z)$ holds for all $z\in[0,j]$; (ii) $x(j+1)\mathscr{R}\hat{x}(j+1)$ does not hold. One may notice that $(1-\delta)\sum_{\hat{x}\in\hat{X}'_{\epsilon}(q(z-1))}\hat{T}(\hat{x}\,|\,\hat{x}(z-1),\hat{u}(z-1))$ is the same as Term 2 in Part 1. Therefore, the number of operations needed for computing Part 3 at the time instant $k-1$ is also proportional to the number of states in the set $\hat{X}'_{\epsilon}(q(k-1))$.

In conclusion, the number of operations required for computing $\mathcal{E}_{pv}(k)$ as in (4.2.25) is proportional to the number of elements in $\hat{X}$ and U_f. Consequently, $\mathcal{E}_{pv}(k)$ can be computed efficiently at runtime. The real-time applicability of the supervisor in the experiments in Section 4.4.

Supervisor for the Problem of Worst-case Violation. Next, the design of the supervisor for the problem of worst-case violation is discussed.

Definition 4.2.19. (History-based Supervisor for the Problem of Worst-case Violation) *At each time instant $k \in [0, H-1]$, given the history path $\bar{\omega}_k$ as in Definition 4.2.16, the validity of an input $u_{uc}(k)$ from the unverified controller is checked by the following two steps:*

i) Check the set U_f as in (4.2.20) and reject $u_{uc}(k)$ if U_f is empty;

i) If U_f is not empty, estimate $\mathcal{E}_{pv}(k)$ as

$$\begin{aligned}\mathcal{E}_{pv}(k) = \underline{\mathcal{E}}_{pv}(k) = 1 - \prod_{z=1}^{k}\Big((1-\delta)\sum_{\hat{x}\in \hat{X}'_{-\epsilon}(q(z-1))} \hat{T}(\hat{x} \,|\, \hat{x}(z-1), \hat{u}(z-1))\Big)\\ \times (1-\delta)\Big(1 - \sum_{\hat{x}\in\hat{X}} \underline{V}_{*,H-k-1}(\hat{x}, \bar{q}_*(\hat{x}(k), q(k)))\hat{T}(\hat{x} \,|\, \hat{x}(k), \hat{u}^*)\Big),\end{aligned} \tag{4.2.29}$$

where $\underline{V}_{*,H-k-1}$ is as in (4.2.18) associated with the safety advisor, $\bar{q}_*$ is as in (4.2.19),

$$\hat{u}^* = arg \max_{\hat{u}\in U_f} \underline{\mathcal{E}}_{pv}(k), \tag{4.2.30}$$

and

$$\hat{X}'_{-\epsilon}(q(z-1)) := \Big\{\hat{x}\in\hat{X} \,\Big|\, \forall x \in X, \tau(q(z-1), L\circ h(x)) \notin F, h(x) \in N_\epsilon(\hat{h}(\hat{x}))\Big\}, \tag{4.2.31}$$

with $N_\epsilon(\hat{h}(\hat{x}))$ as in (4.2.9). If $\mathcal{E}_{pv}(k) \leq \eta$, the supervisor accepts $u_{uc}(k)$ and feeds $\hat{u}' = \hat{u}^*$ as in (4.2.30) to $\widehat{\mathfrak{D}}$; otherwise, it rejects $u_{uc}(k)$ and feeds $\hat{u}' = \hat{u}$ to $\widehat{\mathfrak{D}}$, with $\hat{u}$ provided by ρ_{*_k} as in (4.2.17) associated with the safety advisor.

By employing the history-based supervisor as in Definition 4.2.19, (4.2.24) is ensured, which solves the problem of worst-case violation. This is formalized in the next theorem.

Theorem 4.2.20. *Consider a gMDP $\mathfrak{D} = (X, U, x_0, T, Y, h)$ and a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ characterizing the desired safety specification. Utilizing the supervisor as in Definition 4.2.19 at all time instants $k \in [0, H-1]$ in the Safe-visor architecture, one has*

$$\mathbb{P}_{\mathfrak{D}}\Big\{y_{\omega H} \models \mathcal{A}\Big\} \leq \eta, \tag{4.2.32}$$

for the problem of worst-case violation as in Problem 4.2.6, where $y_{\omega H}$ are output sequences of $\mathfrak{D}$ up to the time instant H.

The proof of Theorem 4.2.20 is provided in Section 4.6. Note that $\mathcal{E}_{pv}(k)$ in (4.2.29) denotes the maximal probability of $\mathfrak{D}$ not being accepted by $\mathcal{A}$ within the time horizon $[0, H]$, given the history path $\bar{\omega}_k$. The term after the first minus sign in (4.2.29) can be divided into two components as follows:

Algorithm 2: Running mechanism of Safe-visor architecture for gMDP.

```
Input: gMDP $\mathfrak{D} = (X, U, x_0, T, Y, h)$, DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$, safety advisor
       $\tilde{\mathbf{C}}_{\rho^*}$ for the problem of robust satisfaction (resp. $\tilde{\mathbf{C}}_{\rho_*}$ for the problem of
       worst-case violation) as in Section 4.2.2, and the supervisor as in
       Definition 4.2.17 (resp. as in Definition 4.2.19).
Output: $u(k)$ for controlling the system $\mathfrak{D}$ at each time instant $k$.
1  $k = 0$, $x(0) = x_0$
2  while $k < H$ do
3      if $k = 0$ then
4          Initialize $\hat{x}(0) = \hat{x}_0$ such that $x_0 \mathscr{R} \hat{x}_0$.
5          Update $q(0)$ as $\bar{q}_0 = \tau(q_0, L \circ h(x_0))$.
6      else
7          Update the state $x(k)$ of $\mathfrak{D}$ from the measurement.
8          Update $q(k)$ with $q(k) = \tau\big(q(k-1), L \circ h(x(k))\big)$.
9          Update $\hat{x}(k)$ with $\mathscr{L}_T(d\hat{x} | x(k), \hat{x}(k-1), \hat{u}(k-1))$.
10     end
11     Update $u_{uc}(k)$ from the unverified controller.
12     Update $u_{\text{safe}}(k)$ from $\tilde{\mathbf{C}}_{\rho^*}$ (resp. $\tilde{\mathbf{C}}_{\rho_*}$) according to $\hat{x}(k)$, $x(k)$, and $q(k)$.
13     Feed $\hat{x}(k)$, $x(k)$, $q(k)$, $u_{uc}(k)$, and $u_{\text{safe}}(k)$ to the supervisor.
14     Update $u(k)$ and $\hat{u}(k)$ according to the decision of the supervisor.
15     $k = k + 1$.
16 end
```

- Component 1: $\prod_{z=1}^{k} \big((1-\delta) \sum_{\hat{x} \in \hat{X}'_{-\epsilon}(q(z-1))} \hat{T}(\hat{x} \,|\, \hat{x}(z-1), \hat{u}(z-1))\big)$:
  Given the history path $\bar{\omega}_k$, Component 1 denotes the minimal probability of $\mathfrak{D}$ not being accepted by $\mathcal{A}$ within the time horizon $[0, k]$, while $x(z) \mathscr{R} \hat{x}(z)$ holds for all time instant $z \in [0, k]$. According to (4.2.31), for all $\hat{x} \in \hat{X}'_{-\epsilon}(q(z-1))$ with $q(z-1) \in Q$, there is no x with $x \mathscr{R} \hat{x}$ such that F is reached at time step z. In other words, if $\hat{x}(z) \in \hat{X}'_{-\epsilon}(q(z-1))$, one can ensure that $\mathfrak{D}$ will not be accepted by $\mathcal{A}$ at the time instant z if one can ensure $x(z) \mathscr{R} \hat{x}(z)$. Therefore, given $\hat{x}(z-1)$ and $\hat{u}(z-1)$, $(1-\delta) \sum_{\hat{x} \in \hat{X}'_{-\epsilon}(q(z-1))} \hat{T}(\hat{x} \,|\, \hat{x}(z-1), \hat{u}(z-1))$ denotes the minimal probability of $\mathfrak{D}$ not being accepted by $\mathcal{A}$ at z while $x(z) \mathscr{R} \hat{x}(z)$ still holds.
- Component 2: $(1-\delta)\big(1 - \sum_{\hat{x} \in \hat{X}} \underline{V}_{*, H-k-1}(\hat{x}, \bar{q}_*(\hat{x}(k), q(k))) \hat{T}(\hat{x} \,|\, \hat{x}(k), \hat{u}^*)\big)$:
  Component 2 denotes the probability of $\mathfrak{D}$ not being accepted by $\mathcal{A}$ within the time horizon $[k+1, H]$, while (i) $x(k+1) \mathscr{R} \hat{x}(k+1)$ holds, given $\hat{x}(k)$, $q(k)$, and $\hat{u}(k) = \hat{u}^*$; (ii) $\mathfrak{D}$ is controlled by the safety advisor within $[k+1, H]$.

Similar to the computation of (4.2.25), $\mathcal{E}_{pv}(k)$ as in (4.2.29) can be efficiently computed at runtime. In brief, the required number of operations for computing $\mathcal{E}_{pv}(k)$ in (4.2.29) is also proportional to the number of elements in $\hat{X}$ and U_f. The real-time applicability of the supervisor is shown in Section 4.4 through an example. Finally, the running mechanism of the Safe-visor architecture equipped with the safety advisor proposed in Section 4.2.2 and supervisor proposed in Section 4.2.3 is summarized in Algorithm 2.

4.3 Design of Safe-visor Architecture for General Discrete-time Stochastic Games

4.3.1 Problem Formulation

In Section 4.3, the stochastic systems of interest are general discrete-time stochastic games (gDTSGs), as introduced in Definition 2.4.3. Here, it is desired to design a Safe-visor architecture over these systems while enforcing those safety specifications modeled by the accepting languages of deterministic finite automata (DFA), as introduced in Definition 2.5.1. To provide formal safety guarantees regardless of how Player II chooses adversarial inputs, a Safe-visor architecture is designed over Player I considering that Player II will select its action after Player I at each time step and in a rational fashion against the choice of Player I. Note that such setting is common for robust control problems. Moreover, the full state information of the gDTSGs is considered to be available, and safety properties are defined over the output of the gDTSGs.

Throughout Section 4.3, η denotes the *maximal tolerable probability of violating the safety specification.* Accordingly, similar to Section 4.2, two problems are of interest.

Problem 4.3.1. (Worst-case Violation) *Consider a gDTSG $\mathfrak{D}$ as in Definition 2.4.3. The* problem of worst-case violation *with respect to the parameter η is to design a Safe-visor architecture as in Figure 1.1 (if existing) for Player I of $\mathfrak{D}$ such that inequality*

$$\mathbb{P}_{\mathfrak{D}}\Big\{y_{\omega H} \models \mathcal{A}\Big\} \leq \eta, \tag{4.3.1}$$

holds regardless of how Player II provides adversarial inputs, where $\mathcal{A}$ is a DFA accepting all bad prefixes of the desired safety specification, and $y_{\omega H}$ are the output sequences of $\mathfrak{D}$ up to time step H as in Definition 2.4.4.

Problem 4.3.2. (Robust Satisfaction) *Consider a gDTSG $\mathfrak{D}$ as in Definition 2.4.3. The* problem of robust satisfaction *with respect to the parameter η is to design a Safe-visor architecture as in Figure 1.1 (if existing) for Player I of $\mathfrak{D}$ such that inequality*

$$\mathbb{P}_{\mathfrak{D}}\Big\{y_{\omega H} \models \mathcal{A}\Big\} \geq 1-\eta, \tag{4.3.2}$$

holds regardless of how Player II provides adversarial inputs, where $\mathcal{A}$ is a DFA that accepts all good prefixes of the desired safety specification, and $y_{\omega H}$ are the output sequences of $\mathfrak{D}$ up to time step H as in Definition 2.4.4.

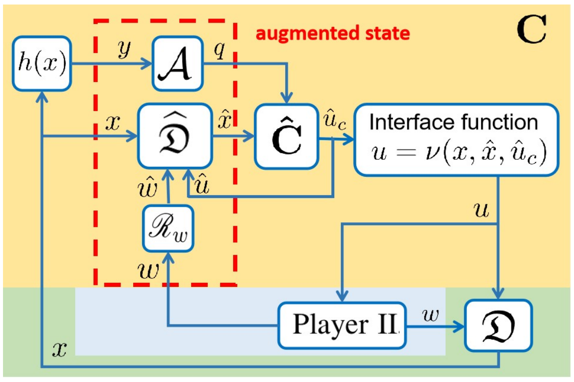


Figure 4.5: Safety advisor (yellow region), which is a controller over $\mathfrak{D}$ (green region), with augmented state $(x, \hat{x}, q, \hat{u}, w)$ (red dashed rectangle).

4.3.2 Design of Safety Advisor

Given a gDTSG $\mathfrak{D}$ modeling the stochastic systems of interest, a finite abstraction of $\mathfrak{D}$, denoted by $\widehat{\mathfrak{D}}$, is constructed for building the safety advisor using the results in Section 3.2.1. Then, one needs to deploy the results in Section 3.2.2 to established an (ϵ, δ)-approximate probabilistic relation between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$, which characterizes the similarity between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$.

Consider the finite abstraction $\widehat{\mathfrak{D}}$. To construct the safety advisor, one needs to synthesize a discrete controller $\hat{\mathbf{C}}$ over the finite abstraction $\widehat{\mathfrak{D}}$ by leveraging Proposition 3.3.11 when focusing on the problem of worst-case violation as in Problem 4.3.1, As for the problem of robust satisfaction as stated in Problem 4.3.2, such controller $\hat{\mathbf{C}}$ should be constructed based on Proposition 3.3.5. Using controller $\hat{\mathbf{C}}$, the safety advisor $\mathbf{C}$ is built as in Figure 4.5. Here, the safety advisor utilizes an augmented state $(x, \hat{x}, q, \hat{u}, w)$, which contains states x, $\hat{x}$, and q of $\mathfrak{D}$, $\widehat{\mathfrak{D}}$, and $\mathcal{A}$, respectively, the control input $\hat{u}$ fed to $\widehat{\mathfrak{D}}$, and the adversary input w from Player II of $\mathfrak{D}$. The running mechanism of $\mathbf{C}$ at each time step is summarized in Algorithm 3.

4.3.3 Design of Supervisor

Next, the design of the supervisor is elaborated. Given a gDTSG $\mathfrak{D}$ and a safety specification modeled by DFA $\mathcal{A}$, the design of the supervisor is depicted in Figure 4.6. Here, the supervisor consists of a *augmented state* and a *decision maker*. The augmented state of the supervisor, denoted by $(x, \hat{x}, q, \hat{u}, w)$, is the same as that of the safety advisor, and I simply say the *augmented state of the Safe-visor architecture* in the rest of the discussion for the sake of brevity. At runtime, x, $\hat{x}$, q, and w in the augmented states are updated as described in Algorithm 3. Meanwhile, different from step 8 of Algorithm 3, $\hat{u}$ here is updated as $\hat{u} := \hat{u}'$, in which $\hat{u}'$ is determined based on the decision of the supervisor (cf. Definition 4.3.4 and 4.3.6, either accepting or rejecting the unverified controller). With the augmented state, the decision maker of

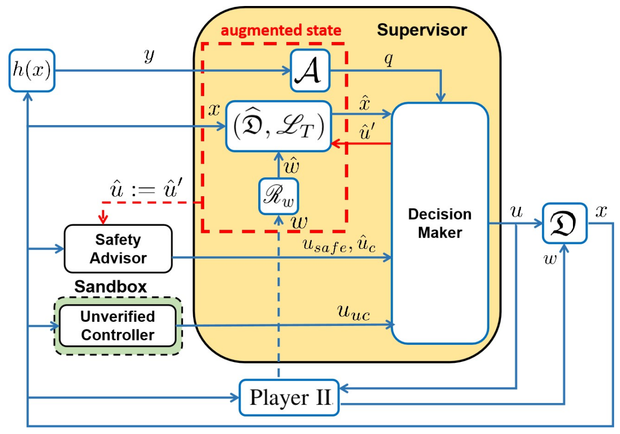


Figure 4.6: Supervisor in the architecture (yellow region).

the supervisor decides whether or not to accept the input from the unverified controller at time instant k, denoted by $u_{uc}(k)$, in the following way:

- Step 1: Presume that $u_{uc}(k)$ is accepted. If the (ϵ,δ)-approximate probabilistic relation between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$ does not hold any more, reject $u_{uc}(k)$ without going through Step 2 and feed input from the safety advisor, denoted by u_{safe}, to $\mathfrak{D}$; proceed to Step 2, otherwise;
- Step 2: Estimate the probability of violating the desired safety properties, denoted by $\mathcal{E}_{pv}(k)$, presuming that $u_{uc}(k)$ is accepted. Accept $u_{uc}(k)$ if $\mathcal{E}_{pv}(k) \leq \eta$; otherwise, feed u_{safe} to $\mathfrak{D}$.

Concretely, step 1 aims at maintaining the (ϵ,δ)-approximate probabilistic relation between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$, which is crucial for providing safety guarantee. One can check Step 1 with the following proposition.

Proposition 4.3.3. *Consider a gDTSG $\mathfrak{D}=(X,U,W,X_0,\ Y,h)$ and its finite abstraction $\widehat{\mathfrak{D}} = (\hat{X},\hat{U},\hat{W},\hat{X}_0,\hat{T},\ Y,\hat{h})$ with $\widehat{\mathfrak{D}} \preceq_{\epsilon}^{\delta} \mathfrak{D}$ with respect to the relation $\mathscr{R}$ and $\mathscr{R}_w$ as in* (3.2.15) *and* (3.2.16)*, respectively. If the set*

$$U_f := \Big\{ \hat{u} \in \hat{U} \Big| \forall (w,\hat{w}) \in \mathscr{R}_w, \mathbb{P}\big\{(x',\hat{x}') \in \mathscr{R}\big\} \geq 1-\delta \text{ holds}, \\ \text{with } x' = f(x(k), u_{uc}(k), w, \varsigma(k)), \hat{x}' = \hat{f}(\hat{x}(k), \hat{u}, \hat{w}, \hat{\varsigma}(k)) \Big\}, \quad (4.3.3)$$

is not empty, then the (ϵ,δ)-approximate probabilistic relation can be maintained between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$ at the time instant $k+1$ when $u_{uc}(k)$ is applied to $\mathfrak{D}$ at the time instant k.

For the set U_f defined in (4.3.3), f and $\hat{f}$ are defined as in (2.4.4), and $x(k) \in X$, $\hat{x}(k) \in \hat{X}$, $\varsigma(k)$ and $\hat{\varsigma}(k)$ denote current states of $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$, noises affecting $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$,

Algorithm 3: Running mechanism of the safety advisor for gDTSGs.

```
Input: A gDTSG $\mathfrak{D}$, a safety specification modeled by DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$,
       safety advisor $\mathbf{C}$, with its associate augmented state $(x, \hat{x}, q, \hat{u}, w)$, and the
       current state $x(k)$ of $\mathfrak{D}$.
Output: $u(k)$ for controlling $\mathfrak{D}$ at each time step $k \in \mathbb{N}$.
1 if $k = 0$ then
2     Update $\hat{x}(k)$ such that $(x(k), \hat{x}(k)) \in \mathscr{R}$ (cf. Condition 3 of
        Definition 2.4.6).
3 else
4     Update $\hat{x}(k)$ according to $x(k)$, the stochastic kernel $\mathscr{L}_T$, and $w(k-1)$ (cf.
        Condition 2 of Definition 2.4.6 and Definition 3.3.1).
5 end
6 Update $q$ of $\mathcal{A}$ as $q(k) = \tau(q(k-1), L \circ h(x(k)))$.
7 Compute $u(k)$ by refining $\hat{u}_c$, which is offered by $\hat{\mathbf{C}}$ based on $\hat{x}(k)$ and $q(k)$, to
    $\mathfrak{D}$ with the interface function $\nu$ (cf. Figure 4.5).
8 Updates $\hat{u}(k)$ as $\hat{u}(k) := \hat{u}_c$.
9 Update $w(k)$ after Player II has made decision.
```

respectively. As a key insight, if $U_f \neq \emptyset$, then, by definition of U_f, there exists at least one $\hat{u} \in \hat{U}$ corresponding to $u_{uc}(k)$ such that $(x(k+1), \hat{x}(k+1)) \in \mathscr{R}$ holds with the probability of at least $1-\delta$, indicating that the approximate probabilistic relation between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$ is maintained. In general, checking the non-emptiness of U_f depends on the concrete form of f. In Section 4.4.3, I show how to check whether U_f is empty using Proposition 4.3.3 via a case study of a quadrotor.

Having Proposition 4.3.3, I am ready to discuss the design of the supervisor for Problem 4.3.1 and 4.3.2.

Problem of worst-case violation To propose the design of the supervisor for Problem 4.3.1, the following definition is required.

Definition 4.3.4. (History-based Supervisor for the Problem of Worst-case Violation over gDTSG) *Consider a gDTSG* $\mathfrak{D} = (X, U, W, X_0, Y, h)$ *and its finite abstraction* $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ *with* $\widehat{\mathfrak{D}} \preceq_\epsilon^\delta \mathfrak{D}$*, DFA* $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ *modeling the desired safety specification, a labeling function* $L : Y \to \Pi$ *associated with* $\mathcal{A}$ *as in Definition 3.1.5, and* η *to be the maximal tolerable probability of violating the desired safety property. For all* $k \in [0, H-1]$*, the validity of an input* $u_{uc}(k)$ *from the unverified controller is checked as follows:*

1. *Reject* $u_{uc}(k)$ *if* U_f *as in* (4.3.3) *is empty;*

2. *If* U_f *is not empty, compute* $\mathcal{E}_{pv}(k)$ *as*

$$\mathcal{E}_{pv}(k) := 1 - \mathcal{C}_1(k)\mathcal{C}_2(k), \tag{4.3.4}$$

with

$$\mathcal{C}_1(k) := \prod_{z=1}^{k} \Big((1-\delta) \min_{\hat{w}\in\hat{W}} \sum_{\hat{x}\in\hat{X}'_{-\epsilon}(q(z-1))} \hat{T}\big(\hat{x} \,\big|\, \hat{x}(z-1), \hat{u}(z-1), \hat{w}\big)\Big), \tag{4.3.5}$$

$$\mathcal{C}_2(k) := (1-\delta)\Big(1 - \max_{\hat{w}\in\hat{W}} \sum_{\hat{x}\in\hat{X}} \underline{V}_{*,H-k-1}\big(\hat{x}, \bar{q}_*(\hat{x}(k), q(k))\big)\hat{T}\big(\hat{x} \,\big|\, \hat{x}(k), \hat{u}^*, \hat{w}\big)\Big), \tag{4.3.6}$$

$\underline{V}_{*,H-k-1}$ *and* $\bar{q}_*$ *as* (3.3.22) *and* (3.3.24)*, respectively,*

$$\hat{u}^* := arg \max_{\hat{u}\in U_f} \mathcal{E}_{pv}(k), \tag{4.3.7}$$

and

$$\hat{X}'_{-\epsilon}(q(z-1)) := \big\{\hat{x}\in\hat{X} \,\big|\, \forall x \in X, \tau(q(z-1), L\circ h(x)) \notin F, h(x)\in N_\epsilon(\hat{h}(\hat{x}))\big\},$$

in which $N_\epsilon(\hat{h}(\hat{x})) := \{y \in Y \mid \|y - \hat{y}\| \leq \epsilon\}$*. If* $\mathcal{E}_{pv}(k) \leq \eta$*, the supervisor accepts* $u_{uc}(k)$ *and update the augmented state with* $\hat{u}' := \hat{u}^*$ *(cf. Figure 4.6), with* $\hat{u}^*$ *being computed as in* (4.3.7)*; otherwise, it rejects* $u_{uc}(k)$ *and set* $\hat{u}'$ *as* $\hat{u}' := \hat{u}_c$*, with* $\hat{u}_c$ *provided by the safety advisor (cf. Algorithm 3).*

By leveraging the supervisor in Definition 4.3.4, the formal guarantee can be provided with the following result.

Theorem 4.3.5. *Consider a gDTSG* $\mathfrak{D} = (X, U, W, X_0, Y, h)$ *and a DFA* $\mathcal{A}$ *modeling the desired safety properties. For the problem of worst-case violation as in Problem 4.3.1, by leveraging the supervisor in Definition 4.3.4 at all time* $k \in [0, H-1]$ *in the Safe-visor architecture for Player I of* $\mathfrak{D}$*, one has*

$$\mathbb{P}\{y_{\omega H} \models \mathcal{A}\} \leq \eta, \tag{4.3.8}$$

regardless of how Player II provides adversarial inputs, with $y_{\omega H}$ *being output sequences of* $\mathfrak{D}$ *as in Definition 2.4.4.*

Detailed proof of Theorem 4.3.5 are provided in Section 4.7. As a key insight, consider a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$.

- $\mathcal{C}_1(k)$ denotes the minimal probability of F not being reached over the time horizon $[0, k]$, while one has $(x(z), \hat{x}(z)) \in \mathscr{R}$, $\forall z \in [0, k]$. Considering the definition of $\hat{X}'_{-\epsilon}(q(z-1))$ as in Definition 4.3.4, one can verify $\forall \hat{x} \in \hat{X}'_{-\epsilon}(q(z-1))$ with $q(z-1) \in Q$, $\nexists x$ with $(x, \hat{x}) \in \mathscr{R}$ such that F is reached at time step z. In other words, if $\hat{x}(z) \in \hat{X}'_{-\epsilon}(q(z-1))$, one can ensure that F is not reached at the time z by ensuring $(x(z), \hat{x}(z)) \in \mathscr{R}$. Hence, given $\hat{x}(z-1)$, $\hat{u}(z-1)$, and $q(z-1)$,

$$\mathcal{C}'(z) := \min_{\hat{w}\in\hat{W}} (1-\delta) \sum_{\hat{x}\in\hat{X}'_{-\epsilon}(q(z-1))} \hat{T}(\hat{x}|\hat{x}(z-1), \hat{u}(z-1), \hat{w}), \tag{4.3.9}$$

denotes the minimal probability of F not being reached at time z while $(x(z), \hat{x}(z)) \in \mathscr{R}$ still holds.

- $\mathcal{C}_2(k)$ is the probability of F not being reached over the time horizon $[k+1, H]$, while (i) $(x(k+1), \hat{x}(k+1)) \in \mathscr{R}$ holds, given $\hat{x}(k)$, $q(k)$, and $\hat{u}(k) = \hat{u}^*$; (ii) $\mathfrak{D}$ is controlled by the safety advisor within $[k+1, H]$.

Hence, $\mathcal{C}_1(k)\mathcal{C}_2(k)$ denotes the lower bound on the probability of F not being reached over $[0, H]$. Since (4.3.7) ensures that all $\hat{u}$ which maintains the (ϵ,δ)-approximate probabilistic relation between $\mathfrak{D}$ and $\widehat{\mathfrak{D}}$ would not result in $\mathcal{E}_{pv} > \eta$. By accepting $u_{uc}(k)$ at each time instant k where $1-\mathcal{C}_1(k)\mathcal{C}_2(k) \leq \eta$ holds, one can ensure that (4.3.8) holds. Moreover, the number of operations required for computing $\mathcal{E}_{pv}(k)$ in (4.3.4) is proportional to cardinality of sets $\hat{X}$, $\hat{W}$, and U_f. Concretely,

- $\mathcal{C}_1(k)$: One can verify that $\mathcal{C}_1(k) = \mathcal{C}_1(k-1)\mathcal{C}'(k)$, with $\mathcal{C}'(k)$ as in (4.3.9). Since $\mathcal{C}_1(k-1)$ has already been computed at time step $k-1$, one only needs $\mathcal{C}'(k)$ to obtain $\mathcal{C}_1(k)$ at time k. On the other hand, $\forall q \in Q$, set $\hat{X}'_{-\epsilon}(q)$ can be computed offline, and $\hat{T}$ is readily computed when constructing the finite abstraction of the original gDTSG. Hence, the number of operations required for computing $\mathcal{C}_1(k)$ at time instant k is proportional to the cardinality of the set $\hat{X}'_{-\epsilon}(q(k-1))$ and $\hat{W}$.

- $\mathcal{C}_2(k)$: Since $\underline{V}_{*,H-k-1}$ and $\hat{T}$ have already been computed when synthesizing the safety advisor, the number of operations required for computing $\mathcal{C}_2(k)$ is proportional to the number of elements in sets $\hat{X}$, $\hat{W}$, and U_f.

Problem of robust satisfaction To propose the design of the supervisor for the problem of robust satisfaction as in Problem 4.3.2, the following definition is needed.

Definition 4.3.6. (History-based Supervisor for the Problem of Robust Satisfaction over gDTSG) *Consider a gDTSG $\mathfrak{D} = (X, U, W, X_0, Y, h)$ and its finite abstraction $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$, DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ modeling the desired safety specification, a labeling function $L : Y \rightarrow \Pi$ associated with $\mathcal{A}$ as in Definition 3.1.5, and η to be the maximal tolerable probability of violating the desired safety property. At each time instant $k \in [0, H-1]$, the feasibility of an input $u_{uc}(k)$ from the unverified controller is checked by the following two steps:*

i) Check set U_f as in (4.3.3) and reject $u_{uc}(k)$ if U_f is empty;

ii) If U_f is not empty, estimate $\mathcal{E}_{pv}(k)$ as

$$\mathcal{E}_{pv}(k) := \prod_{z=1}^{k} \Big((1-\delta) \max_{\hat{w} \in \hat{W}} \sum_{\hat{x} \in \hat{X}'_{\epsilon}(q(z-1))} \hat{T}(\hat{x} \,|\, \hat{x}(z-1), \hat{u}(z-1), \hat{w})\Big)$$
$$\times (1-\delta)\Big(1 - \min_{\hat{w} \in \hat{W}} \sum_{\hat{x} \in \hat{X}} \bar{V}^*_{H-k-1}\big(\hat{x}, \underline{q}^*(\hat{x}(k), q(k))\big)\hat{T}(\hat{x} \,|\, \hat{x}(k), \hat{u}^*, \hat{w})\Big)$$

$$+\delta+\sum_{j=1}^{k}\Big(\delta\prod_{z=1}^{j}\big((1-\delta)\max_{\hat{w}\in\hat{W}}\sum_{\hat{x}\in\hat{X}'_\epsilon(q(z-1))}\hat{T}(\hat{x}\,|\,\hat{x}(z-1),\hat{u}(z-1),\hat{w})\big)\Big),\quad(4.3.10)$$

with $\bar{V}^*_{H-k-1}$ *as in* (3.3.9) *associated with the safety advisor,* $\underline{q}^*$ *as in* (3.3.11),

$$\hat{u}^* = arg\max_{\hat{u}\in U_f}\bar{\mathcal{E}}_{pv}(k),\quad(4.3.11)$$

and

$$\hat{X}'_\epsilon(q(z-1)) := \Big\{\hat{x}\in\hat{X}\,\Big|\,\exists x\in X, \tau(q(z-1), L\circ h(x))\notin F, h(x)\in N_\epsilon(\hat{h}(\hat{x}))\Big\},\quad(4.3.12)$$

where $N_\epsilon(\hat{h}(\hat{x})) := \{y\in Y \mid \|y-\hat{y}\|\leq\epsilon\}$. *If* $\mathcal{E}_{pv}(k)\leq\eta$, *the supervisor accepts* $u_{uc}(k)$ *and feeds* $\hat{u}'=\hat{u}^*$ *as in* (4.3.11) *to* $\widehat{\mathfrak{D}}$; *otherwise, it rejects* $u_{uc}(k)$ *and feeds* $\hat{u}'=\hat{u}$ *to* $\widehat{\mathfrak{D}}$, *with* $\hat{u}$ *provided by* ρ^*_k *as in* (3.3.7) *associated with the safety advisor.*

By utilizing the history-based supervisor as in Definition 4.3.6, the problem of robust satisfaction over gDTSGs can be solved. This is formally formulated in the next theorem.

Theorem 4.3.7. *Consider a gDTSG* $\mathfrak{D}=(X,U,W,X_0,Y,h)$ *and a DFA* $\mathcal{A}=(Q,q_0,\Pi,\tau,F)$ *that characterizes the desired safety specification. For the problem of robust satisfaction as in Problem 4.3.2, by employing the supervisor as in Definition 4.3.6 at all time instants* $k\in[0,H-1]$ *in the Safe-visor architecture for Player I of* $\mathfrak{D}$*, one has*

$$\mathbb{P}_{\mathfrak{D}}\Big\{y_{\omega H}\models\mathcal{A}\Big\}\geq 1-\eta,\quad(4.3.13)$$

regardless of how Player II provides adversarial inputs, with $y_{\omega H}$ *being output sequences of* $\mathfrak{D}$ *up to the time instant* H.

The proof of Theorem 4.3.7 is provided in Section 4.7. Note that similar to the computation of (4.2.29), $\mathcal{E}_{pv}(k)$ as in (4.3.10) can be computed efficiently at runtime, as the number of operations required for computing $\mathcal{E}_{pv}(k)$ in (4.3.10) is proportional to the number of elements in sets $\hat{X}$, $\hat{W}$, and U_f. Finally, the running mechanism of the proposed Safe-visor architecture at each time step k over gDTSG is summarized in Algorithm 4. The real-time applicability of the Safe-visor architecture will be shown in Section 4.4.3 via a case study (cf. Table 4.3).

4.4 Case Studies

In this section, I first apply the results in Section 4.2 to the running example in Section 4.2 and a control problem regarding a DC motor. Then, the results in Section 4.3

Algorithm 4: Running mechanism of the Safe-visor architecture over gDTSGs.

```
Input: A gDTSG $\mathfrak{D}$, a DFA $\mathcal{A}$ modeling the desire specification, safety advisor $\mathbf{C}$
       as in Figure 4.5, supervisor as in Definition 4.3.4 (resp. Definition 4.3.6),
       and $u_{uc}(k)$ from the unverified controller.
Output: $u(k)$ for controlling $\mathfrak{D}$ at each time step $k \in \mathbb{N}$.
1 Compute $\mathcal{E}_{pv}(k)$ as in (4.3.4) (resp. (4.3.10)).
2 Update $x(k)$, $\hat{x}(k)$, and $q(k)$ in the augmented state as in Algorithm 3.
3 Decide whether to accept $u_{uc}(k)$.
4 if $u_{uc}(k)$ is accepted then
5     Set $u(k) = u_{uc}(k)$ and update $\hat{u}(k)$ in the augmented state as $\hat{u}(k) = \hat{u}^*$
       with (4.3.7) (resp. (4.3.11)).
6 else
7     Obtain $u_{\text{safe}}(k)$ and $\hat{u}_c(k)$ from the safety advisor, set $u(k) = u_{\text{safe}}(k)$, and
       update $\hat{u}(k)$ in the augmented state as $\hat{u}(k) = \hat{u}_c(k)$.
8 end
9 Update $w(k)$ in the augmented state based on the decision of Player II.
```

will be deployed to a case studyof a quadrotor tracking a ground vehicle using a DNNs-based agent. In the first two case studies, the desired safety guarantees are validated through empirical Monte Carlo simulation. These simulation are performed via `MATLAB 2019b`, on a machine with Windows 10 operating system (Intel(R) Xeon(R) E-2186G CPU (3.8 GHz) and 32 GB of RAM). The last case study will be used for 1) empirical Monte Carlo simulation; 2) experiment on a physical test-bed of quadrotor helicopter. The physical test-bed includes: 1) a quadrotor equipped with Pixhawk Mini as flight control unit and Raspberry Pi Zero as flight companion computer; 2) Vicon motion capture system for capturing the position and velocity of the quadrotor at runtime; and 3) a ground control station (GCS) with Ubuntu 20.04 (Intel Core i9-10900K CPU (3.7 GHz) and 32 GB of RAM). The simulations are performed via `MATLAB 2019b` on the GCS.

4.4.1 Running Example

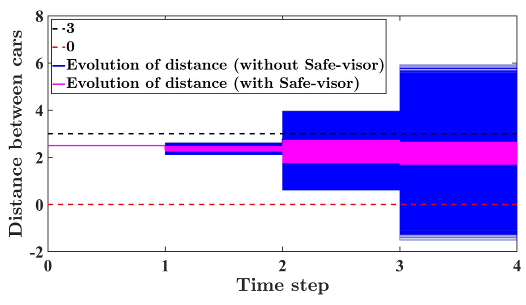
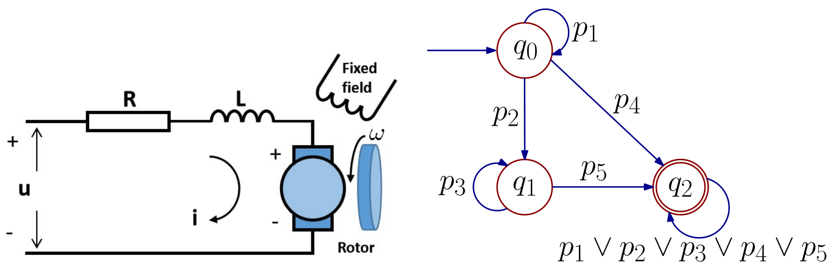


Figure 4.7: Simulation results for the running example.



(a) DC Motor. (b) DFA modelling ψ_{dc}.

Figure 4.8: DC Motor and DFA describing ψ_{dc} for the DC motor, with q_2 being the accepting state, alphabet $\Pi = \{p_1, p_2, p_3, p_4, p_5\}$, and labeling function $L : Y \rightarrow \Pi$ with $L(y) = p_1$, when $y \in \big([1.5\pi, 1.875\pi) \cup (2.125\pi, 2.5\pi]\big) \times [0, 2.4]$; $L(y) = p_2$, when $y \in [1.875\pi, 2.125\pi] \times [0, 2.4]$; $L(y) = p_3$, when $y \in [1.75\pi, 2.25\pi] \times [0, 2.4]$; $L(y) = p_4$, when $y \in \mathbb{R}^2 \backslash ([1.5\pi, 2.5\pi] \times [0, 2.4])$; $L(y) = p_5$, when $y \in \mathbb{R}^2 \backslash ([1.75\pi, 2.25\pi] \times [0, 2.4])$.

For simulating Safe-visor architecture regarding the safety specification ψ for the running example, the system is initialized at $x_0 = [2.5\,; 2.4\,; 1.5]$, with $\eta = 0.1$. To ensure the last condition in Definition 2.4.8, the finite abstraction is initialized with $\hat{x}_0 = 2.55$. Moreover, to model the unverified controller that would endanger the system, a controller that randomly selects input at each time instant following a uniform distribution within the input range is deployed. The running example is simulated with 1.0×10^5 empirical Monte Carlo runs, and the simulation results are shown in Figure 4.7, and summarized in Table 4.1. One can readily verify that the desired safety probability specified by η is respected.

4.4.2 DC Motor

In the second case study, a DC motor as in Figure 4.8 (a) is of interest, which can be described by the following difference equations:

$$\mathfrak{D}\colon \begin{cases} x(k+1) = Ax(k) + Bu(k) + Ee^{Fx(k)} + R\varsigma(k), \\ y(k) = x(k), \end{cases} \qquad k \in \mathbb{N}, \tag{4.4.1}$$

with

$$A = \begin{bmatrix} 0.6387 & 0.0080 \\ -0.1606 & -0.0020 \end{bmatrix}, \; B = [0.3996\,; 0.4011], \; E = [-0.2\,; 0],$$
$$F = [-0.0796\,; 0]^\top, \text{ and } R = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}.$$

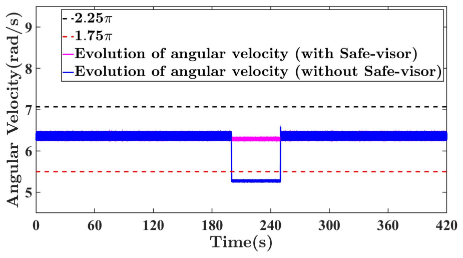
Here, $x(k) := [x_1(k)\,; x_2(k)]$ is the state of the DC motor, in which $x_1(k)$ and $x_2(k)$ are the angular velocity and the armature current of the motor, respectively. Input $u(k) \in [0, 9]$ is the voltage source applied to the motor's armature. Additionally, $\varsigma(k) := [\varsigma_1(k)\,; \varsigma_2(k)]$, where $\varsigma_1(k)$ and $\varsigma_2(k)$ are standard Gaussian random variables that affect $x_1(k)$ and $x_2(k)$, respectively. This model is adapted from a continuous-time model of a DC-motor as in [207] by discretizing it with a sampling time $\tau = 0.02$s and including stochasticity in the model as an additive noise. In this case study, the DC motor is required to satisfy a safety specification ψ_{dc} within 7 minutes: (i) the armature current should be within $[0, 2.4]$; (ii) the angular velocity should stay within $[1.5\pi, 2.5\pi]$; (iii) additionally, if angular velocity reaches $[1.875\pi, 2.215\pi]$, it should stay within $[1.75\pi, 2.25\pi]$ afterwards, instead of $[1.5\pi, 2.5\pi]$. Accordingly, a DFA is used to model such specification, which accepts all bad prefixes of output sequences that violate ψ_{dc}, as shown in Figure 4.8 (b). Accordingly, the problem of worst-case violation as in Problem 4.2.6 should be solved.

First, the safety advisor for the DC motor is designed following the proposed approach in Section 4.2.2. To construct a finite abstraction for the model of DC motor, the input set $[0, 9]$ are partition into 40 partitions and $X = [1.5\pi, 2.5\pi] \times [0, 2.4]$ is chosen as the region of interest. Then, this region is uniformly partitioned into 40 cells on both dimensions, which results in a finite gMDP with 1601 discrete states (1600 states correspond to the representative points of partitions, and 1 sink state) and 40 inputs.

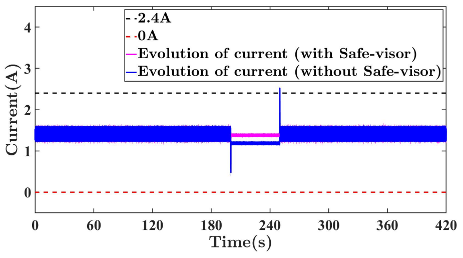
As for establishing an (ϵ,δ)-approximate probabilistic relation using the results in Section 3.2.2 and 3.2.3, $\hat{U}' = \hat{U}$ and $\delta = 0$ are chosen, and the expected range of ϵ is set as $[0.05, 1]$. Then, this finite abstraction is (ϵ, δ)-stochastically simulated by the original model as in (4.4.1) w.r.t. the relation $\mathscr{R} = \{(x, \hat{x}) \,|\, (x - \hat{x})^\top (x - \hat{x}) \leq \epsilon^2\}$ with $\delta = 0$ and $\epsilon = 0.1138$, when the interface function

$$\nu(x, \hat{x}, \hat{u}) := (K + bL)(x - \hat{x}) + \hat{u} + Ge^{F\hat{x}},$$

is employed for the controller refinement. $\mathscr{R} = \{(x, \hat{x}) \,|\, (x-\hat{x})^\top M(x-\hat{x}) \leq \epsilon^2\}$ Here, $\hat{x}$ is the state of the finite abstraction, $K = [-0.5948\,; -0.0110]^\top$, $L = [-0.0452\,; 0.0039]^\top$, $G = 3.0954$, and $b = \frac{e^{Fx} - e^{F\hat{x}}}{F(x-\hat{x})}$. The lifting stochastic kernel for this relation is constructed such that the noise terms in the original and the finite systems are the same. Then, the safety advisor is synthesized using the results in Section 4.2.2.



(a) Evolution of the angular velocity.



(b) Evolution of the armature current.

Figure 4.9: Simulation results for DC motor for safety specification ψ_{dc}.

As for the simulation, the system is initialized at $x_0 = [2\pi\,; 1.256]$ and $\eta = 0.001$ is chosen. To ensure the last condition in Definition 2.4.8, the finite abstraction is initialized with $\hat{x}_0 = [6.3225\,; 1.23]$. As for the unverified controller, a controller is trained

	ψ	ψ_{dc}
Percentage of satisfaction (with Safe-visor architecture)	100%	100%
Average acceptance rate of the unverified controller	27.27%	84.73%
Percentage of satisfaction (without Safe-visor architecture)	55.87%	0%
Percentage of satisfaction (when system is fully controlled by the safety advisor)	100%	100%
Average execution time of the supervisor (ms)	0.1005	0.3782

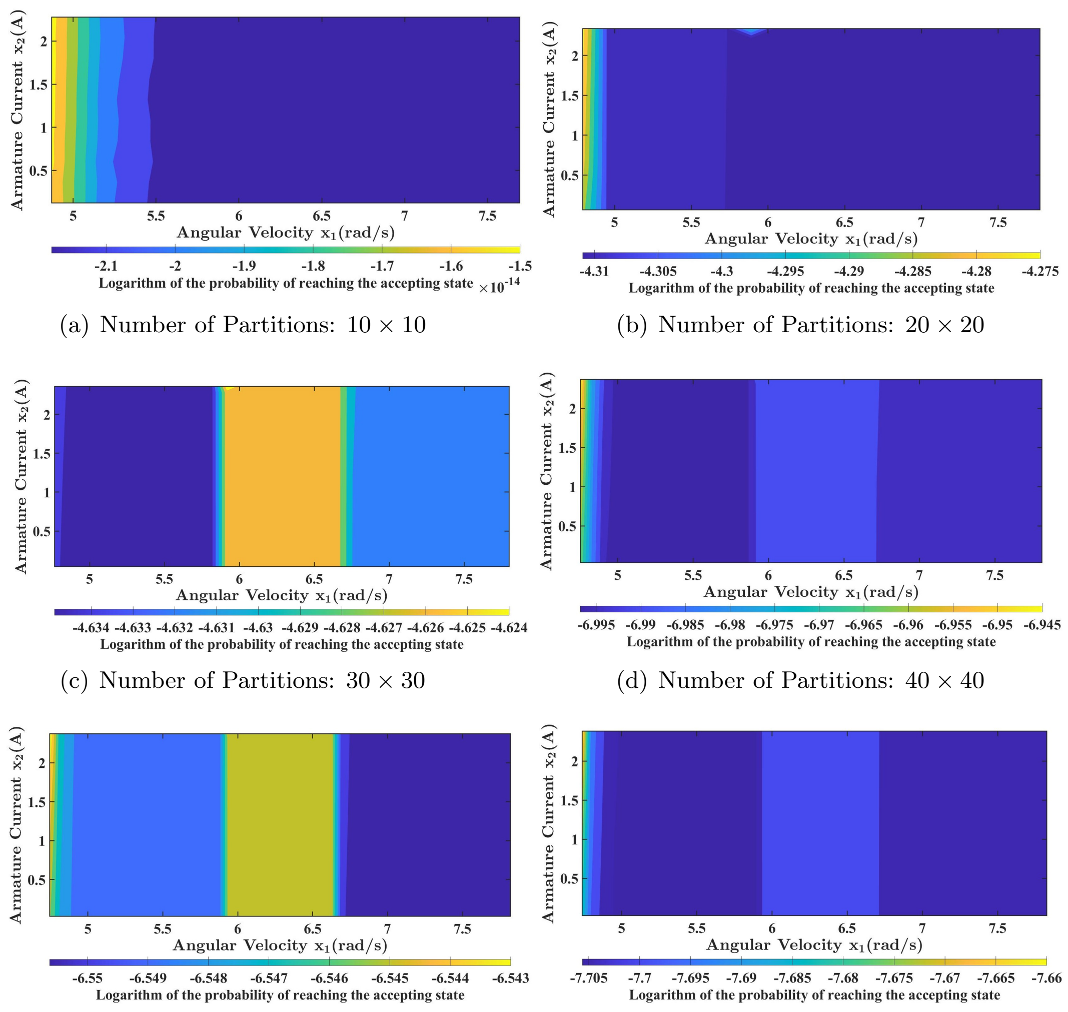
Table 4.1: Simulation results for the running example and the DC motor, in which the average acceptance rate of the unverified controller refers to the average percentage of inputs from the unverified controller being accepted among these runs.

using deep reinforcement learning with deep deterministic policy gradient (DDPG) algorithms [128] for tracking the desired angular velocity, denoted by x_{1_d}. In the simulation, it is configured that $x_{1_d} = 2.1\pi\,\text{rad/s}$ when $k \in [0, 10000] \cup [12500, 21000]$ and $x_{1_d} = 1.78\pi\,\text{rad/s}$ when $k \in [10000, 12500]$. Then, the case study is simulated with 1.0×10^5 empirical Monte Carlo runs and the results are shown in Figure 4.9 and summarized in Table 4.1. Without sandboxing the unverified controller, all output sequences violate ψ_{dc}. Meanwhile, by sandboxing the unverified controller, 100% of output sequences satisfy ψ_{dc} while 84.73% of inputs from the unverified controller can be accepted. One can readily see that the safety probability specified by η is respected. Simultaneously, the unverified controller can still be applied most of the time when it does not endanger the system's safety.

With the case study of the DC motor, the effects of the abstraction resolution on the safety guarantee provided by the safety advisor and the execution speed of the supervisor are also demonstrated. To this end, 6 different abstraction resolutions are selected for the continuous state sets (see Table 4.2, 10×10 means uniformly partitioning $X = [1.5\pi, 2.5\pi] \times [0, 2.4]$ into 10 cells on both dimensions, and so on). The system is initialized at $x_0 = [2\pi\,; 1.256]$ and each case is simulated with 1.0×10^4 empirical Monte Carlo runs. The average execution time of the supervisor and the probabilistic guarantee provided by the safety advisor in each case are shown in Table 4.2 and Figure 4.10, respectively. According to Table 4.2, the average execution time of the supervisor is increasing by reducing the quantization parameters in constructing finite abstractions. However, as shown in Figure 4.10, the upper bound of the probability of reaching the accepting state, which indicates a violation of the desired specification as in Figure 4.8 (b), reduces by reducing the quantization parameters.

4.4.3 Quadrotor

The proposed construction scheme in Section 4.3 is applied here to a case study of controlling a quadrotor using a DNNs-based agent to track a ground vehicle in 1) simulation with 1.0×10^4 different realization of noise; 2) experiment on the physical test-bed. In the experiment on the physical test-bed, the safety advisor, the supervisor,



(a) Number of Partitions: 10 × 10

(b) Number of Partitions: 20 × 20

(c) Number of Partitions: 30 × 30

(d) Number of Partitions: 40 × 40

(e) Number of Partitions: 50 × 50

(f) Number of Partitions: 60 × 60

Figure 4.10: Contours of the safety guarantee provided by the safety advisor for the casse study of the DC motor.

Number of partitions	10×10	20×20	30×30	40×40	50×50	60×60
Number of states	101	401	901	1601	2501	3601
ϵ of the relation	0.5021	0.2297	0.1518	0.1138	0.0911	0.0759
Average execution time (ms)	0.2388	0.2579	0.3174	0.3763	0.4431	0.5041

Table 4.2: Average execution time of the supervisor with different sizes of grids for partitioning the continuous state set.

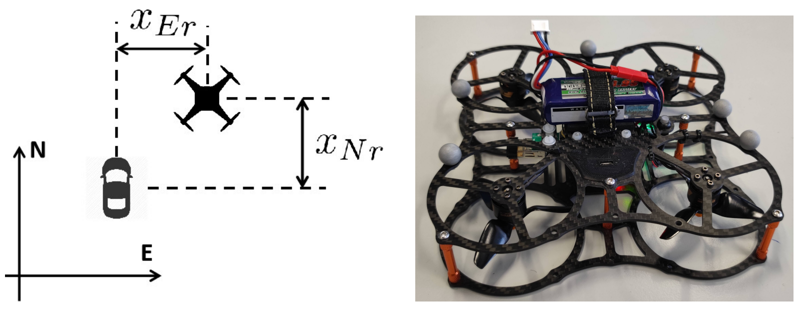


Figure 4.11: Case study for controlling a quadrotor tracking a ground vehicle in Chapter 4

and the DNNs-based controller are running on the GCS. Based on the decision of the supervisor, the GCS sends desired accelerations (i.e. the control input, cf. (4.4.2)) to the quadrotor at runtime.

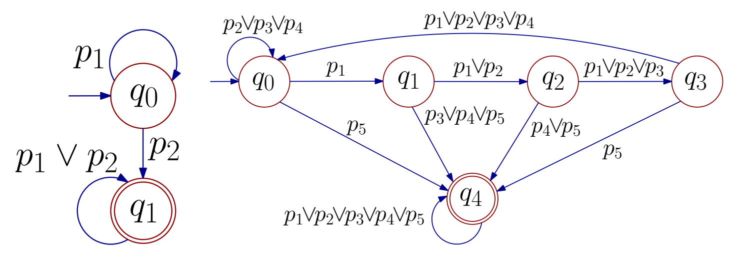


Figure 4.12: Left: DFA $\mathcal{A}_E$, with accepting state q_1, alphabet $\Pi = \{p_1, p_2\}$, and labeling function $L : Y \to \Pi$ with $L(y) = p_1$ when $y \in [-0.5, 0.5]$, and $L(y) = p_2$ when $y \in (-\infty, -0.5) \cup (0.5, +\infty)$. Right: DFA $\mathcal{A}_N$, with accepting state q_4, alphabet $\Pi = \{p_1, p_2, p_3, p_4, p_5\}$, and labeling function $L : Y \to \Pi$ with $L(y) = p_1$ when $y \in [-0.3, 0.3]$, $L(y) = p_2$ when $y \in [-0.4, -0.3) \cup (0.3, 0.4]$, $L(y) = p_3$ when $y \in [-0.45, -0.4) \cup (0.4, 0.45]$, $L(y) = p_4$ when $y \in [-0.5, -0.45) \cup (0.45, 0.5]$, and $L(y) = p_5$ when $y \in (-\infty, -0.5) \cup (0.5, +\infty)$.

Modeling and safety specifications: By employing the feedback linearization technique in [68], the relative motion between the quadrotor and the ground vehicle on N and E axes (see Figure 4.11 (left)) can be modeled as:

$$\begin{cases} x_{\mathsf{i}}(k+1) = Ax_{\mathsf{i}}(k) + Bu_{\mathsf{i}}(k) + Dw_{\mathsf{i}}(k) + R\varsigma_{\mathsf{i}}(k), \\ y_{\mathsf{i}}(k) = Cx_{\mathsf{i}}(k), \qquad\qquad k \in \mathbb{N}, \mathsf{i} \in \{N, E\}, \end{cases} \tag{4.4.2}$$

where $A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$, $B = [\frac{\Delta t^2}{2}; \Delta t]$, $D = -B$, and $C = [1; 0]^\top$, with $\Delta t = 0.1s$ being the sampling time, and $R = \begin{bmatrix} 0.004 & 0 \\ 0 & 0.045 \end{bmatrix}$ being obtained through experimental trials on the physical test-bed. Here, for $\mathsf{i} \in \{N, E\}$, $x_{\mathsf{i}}(k) := [x_{\mathsf{i}r}(k); v_{\mathsf{i}r}(k)]$ with $x_{\mathsf{i}r}(k)$ and $v_{\mathsf{i}r}(k)$ being the relative position and relative velocity between the quadrotor and the vehicle on i axis, respectively; $u_{\mathsf{i}}(k) \in [-2.5, 2.5]$ (m/s^2) denotes the acceleration of the quadrotor on i axis as the control input; $w_{\mathsf{i}}(k) \in [-0.6, 0.6]$ (m/s^2) denotes the acceleration of the vehicle on i axis as the adversary input; $\varsigma_{\mathsf{i}}(k)$ is a standard Gaussian

Safety specifications	P'_s	Acceptance rate	P''_s	T_{avg} (ms)	T_σ (ms)
(ϕ_E, H) (Simulation)	100%	92.75%	100%	3.0790	1.3873
(ϕ_N, H) (Simulation)	100%	92.40%	100%	3.3238	1.3415
(ϕ_E, H) (Test-bed)	100%	2.50%	0%	4.2301	2.1056
(ϕ_N, H) (Test-bed)	100%	8.00%	0%	3.3957	2.7630

Table 4.3: Results of simulation and experiment on the physical test-bed, where P'_s and P''_s denotes the percentage of the outputs satisfying the desired safety properties with using and without using the Safe-visor architecture, respectively; acceptance rate is the percentage of inputs from the DNNs-based controller being accepted among different runs; T_{avg} and T_σ are the average and the standard deviation of the execution time, respectively.

random variable; and $y_i(k)$ is the output. Within 1 min (time horizon $H = 600$), the following safety specifications are desired: (1) ϕ_E: y_E should be within $[-0.5, 0.5]$ (m); (2) ϕ_N: y_N should be within $[-0.5, 0.5]$ (m); additionally, if y_N reaches $[-0.3, 0.3]$ (m) at any time instant k, then y_N should be within $[-0.4, 0.4]$ (m) at time instant $k+1$ and within $[-0.45, 0.45]$ (m) at time instant $k+2$, instead of $[-0.5, 0.5]$ (m). Their associated DFAs $\mathcal{A}_E$ and $\mathcal{A}_N$ are shown in Figure 4.12. Accordingly, the Safe-visor architecture will be designed with respect to ϕ_E and ϕ_N, denoted by sva_E and sva_N, respectively.

Construction of Safe-visor architecture: To construct the safety advisor as introduced in Section 4.3.2, the finite abstraction of the model as in (4.4.2) are built by selecting $X = [-0.5, 0.5] \times [-0.4, 0.4]$ and partitioning X uniformly with grid cells whose sizes are $(0.02, 0.02)$. Then, the input set $[-2.5, 2.5]$ for the quadrotor and the input set $[-0.6, 0.6]$ for the ground vehicle are uniformly divided with 25 and 12 cells, respectively. As a result, a finite gDTSG with 2001 states (denoted by $\hat{X}$), 25 control inputs for Player I (denoted by $\hat{U}'$), and 12 adversarial inputs for Player II (denoted by $\hat{W}$) is obtained. By employing the results in Section 3.2.2 and 3.2.3, the finite abstraction is (ϵ, δ)-stochastically simulated by the original model with respect to the relation $\mathscr{R} := \{(x, \hat{x}) \mid (x-\hat{x})^\top M (x-\hat{x}) \leq \epsilon^2\}$, and $\mathscr{R}_w := \{(w, \hat{w}) | (w-\hat{w})^\top (w-\hat{w}) \leq \tilde{\epsilon}^2\}$, with $\delta = 0$, $\epsilon = 0.0674$, $\tilde{\epsilon} = 0.05$, $M = \left[\begin{smallmatrix} 1.4632 & 0.1757 \\ 0.1757 & 0.0666 \end{smallmatrix}\right]$, and a interface function $u := K(x-\hat{x}) + \hat{u}$ where $K = [-16.66; -4.83]^\top$, and $\hat{U} := \{\hat{u} \in \hat{U}' | ||\hat{u}|| \leq 0.12\}$ is used to build the safety advisor. Having the finite abstraction and the approximate probabilistic relation, the safety advisors are synthesized for sva_E and sva_N as discussed in Section 4.3.2. The total offline computation time[3] for sva_E and sva_N are approximately 1.2 hours and 5.2 hours, respectively.

After the safety advisors are constructed offline, the supervisors is implemented leveraging the results in Theorem 4.3.5, for which checking the non-emptiness of the set U_f in (4.3.3) at runtime is necessary (cf. Proposition 4.3.3). Consider the current state $x(k)$ of the original system, $\hat{x}(k)$ in the current state of the Safe-visor architecture, and

[3]The computation of the cost-to-go function $\underline{V}_{*,n+1}(\hat{x}, q)$ in (3.3.23) for different $(\hat{x}, q) \in \hat{X} \times Q$ are independent from each other. Therefore, the computation can be done in a parallel fashion, which is not implemented.

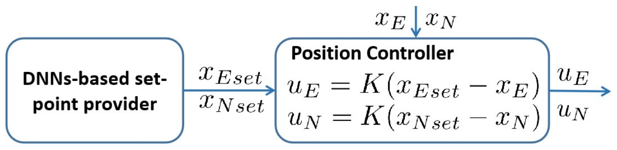

DNNs-based set-point provider
x_{Eset}
x_{Nset}
x_E x_N
Position Controller
$u_E = K(x_{Eset} - x_E)$
$u_N = K(x_{Nset} - x_N)$
u_E
u_N

Figure 4.13: DNNs-based controller deployed in the experiments on the physical test-bed.

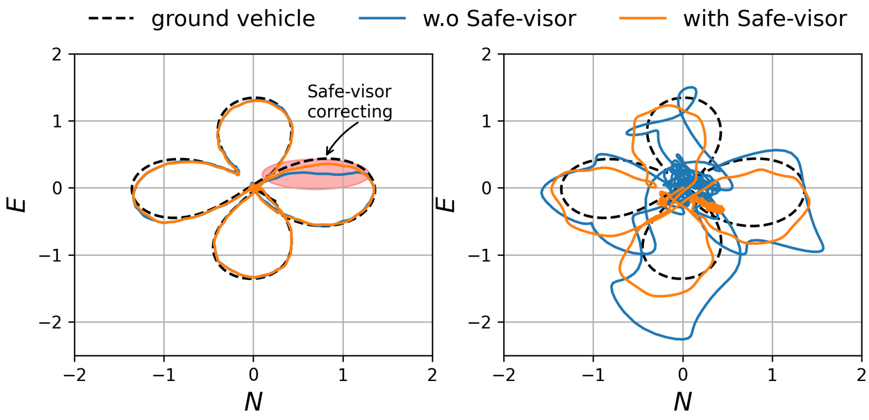


Figure 4.14: Trajectories of the quadrotor and the ground vehicle in simulation (Left), and real-world experiment (Right).

input $u_{dnn}(k)$ provided by the DNNs-based controller. The set U_f is not empty if there exists $\hat{u} \in \hat{U}$ such that

$$\|Ax(k)+Bu_{dnn}(k)+Dw(k)+R\varsigma(k)-(A\hat{x}(k)+B\hat{u}+D\hat{w}(k)+R\hat{\varsigma}(k))\|_M \le \epsilon, \quad (4.4.3)$$

holds for all $\varsigma \in \mathbb{R}^2$, with $\|\bar{x}\|_M := \sqrt{\bar{x}^\top M \bar{x}}$. By setting $\hat{\varsigma}(k) = \varsigma(k)$, (4.4.3) holds if one has $\|\varphi - B\hat{u}\|_M \le \epsilon - \gamma$, with $\varphi := A(x(k) - \hat{x}(k)) + Bu_{dnn}(k)$ and $\gamma := \max_{\beta\in\Delta}\|\beta\|_M + \max_{(w,\hat{w})\in\mathscr{R}_w}\|D(w-\hat{w})\|_M = 0.0152$, where Δ is the set of all possible quantization errors introduced by discretization of the original state set as defined in (3.2.11). Since φ can readily be computed at runtime, one can find out if there exists $\hat{u} \in \hat{U}$ such that (4.4.3) holds efficiently.

Experiments and results In the experiments,a DNNs-based agent is used to control the quadrotor to track the vehicle. The agent is trained as a setpoints provider for a low-level position controller, as depicted in Figure 4.13, with $K = [1.4781; 1.7309]^\top$. The agent takes the current positions and velocities of the quadrotor and the ground vehicle as inputs, and provides the position and velocity setpoints for the quadrotor. Here, DDPG algorithm [128] is used to train the agent in simulation, in which the vehicle follows random trajectories.

In both simulation and the experiment on the test-bed, the ground vehicle follows a clover trajectory. The system is initialized with $x_E = x_N = [0; 0]$ and the maximal tolerable probability of violation for sva_E and sva_N as $\eta_E = \eta_N = 0.01$. The results of the simulation and the experiment on the physical test-bed are summarized in Table 4.3.

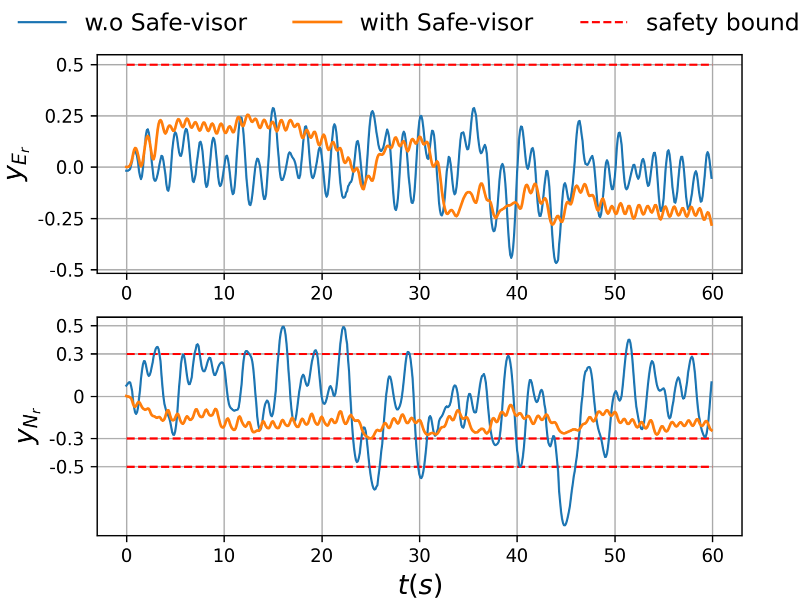


Figure 4.15: Evolution of y_E and y_N with and without using the Safe-visor architecture.

The simulation results show that the desired lower bound of safety probability specified by η_E and η_N are respected, while more than 90% of the inputs from the DNNs-based controllers are accepted. Although the quadrotor tracks the vehicle very well in the simulation without actually violating the safety specification, as marked red in the Figure 4.14 (Left), the architecture rejects some potential risky actions due to the robustness settings as discussed at the beginning of Section 4.3.1. Notably, as shown in Figure 4.14 (Right), the DNNs-based controller behaves worse on the physical test-bed and the safety specifications are violated when the Safe-visor architecture is not deployed (see the bottom part of Figure 4.15, in which y_{Nr} left the region $[-0.5, 0.5]$), which might be attributed to the mismatch between the model that is used for training and the physical system. For instance, the environmental noises and disturbances are not considered in simulation training, as it requires robust training strategies, causing difficulties in convergence. Meanwhile, by leveraging the Safe-visor architecture, the desired safey specifications are enforced, while the DNNs-based controller can still be employed.

4.5 Summary

In this chapter, abstraction-based approaches for constructing Safe-visor architecture has been proposed for stochastic CPSs modeled by gDTSGs and gMDP. In particular, the safety properties of interest here can be characterized by accepting languages of DFA. Robust controllers regarding the desired safety specification are deployed as the safety advisor, which can be constructed leveraging the results in Chapter 3. Concretely, given a finite abstraction that is (ϵ, δ)-stochastically simulated by the original model, a controller based on a finite abstraction is first constructed, . The synthesized controller is then refined over the original systems based on an (ϵ,δ)-approximate probabilistic relation, which is the main key for providing the safety guarantees. To reach a compromise between safety and functionality, history-based supervisors are designed which check the input provided by the unverified controller based on the history paths at runtime. The main idea for this checking is to estimate the probability of violating the desired safety specification presuming the input from the unverified controller is accepted, and compare this estimation with the maximal tolerable probability of violation. Finally, the proposed methodologies are applied to construct the Safe-visor architecture for three case studies.

4.6 Proof fo Statements in Section 4.2

To show Theorem 4.2.18 and Theorem 4.2.20, the definition of the *n-steps reachable state set* of DFA with respect to a gMDP is required.

Definition 4.6.1. (n-steps Reachable State Set) *Given a gMDP $\mathfrak{D} = (X, U, x_0, T, Y, h)$ and a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ with a labeling function $L : Y \to \Pi$, an n-step reachable state set $\tilde{Q}_n(x_0)$ of $\mathcal{A}$ is recursively defined as*

$$\tilde{Q}_0(x_0) = \Big\{q \in Q \,\Big|\, q = \tau(q_0, L \circ h(x_0))\Big\},$$
$$\tilde{Q}_n(x_0) = \Big\{q \in Q \,\Big|\, \exists q' \in \tilde{Q}_{n-1}(x_0), \sigma \in \Pi \text{ s.t. } q = \tau(q', \sigma)\Big\},$$

with $n \in \mathbb{N}_{>0}$.

Additionally, the following lemma is also needed for the proof.

Lemma 4.6.2. *Consider a gMDP $\mathfrak{D} = (X, U, x_0, T, Y, h)$ and its finite abstraction $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{x}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$, and a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ that characterizes the desired safety specification. Given a Markov policy $\rho = (\rho_0, \rho_1, \ldots, \rho_{H-1})$ over the product gMDP $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ within the time horizon $[0, H]$, one has*

$$1 - \bar{V}^{\rho}_{n+1}(\hat{x}, q) = (1 - \delta) \sum_{\tilde{x} \in \hat{X}'_{\epsilon}(q)} \Big(1 - \bar{V}^{\rho}_{n}(\tilde{x}, \underline{q}(\tilde{x}, q))\Big)\hat{T}(\tilde{x} \,|\, \hat{x}, \hat{u}) + \delta, \tag{4.6.1}$$

$$1 - \underline{V}^{\rho}_{n+1}(\hat{x}, q) = (1 - \delta) \sum_{\tilde{x} \in \hat{X}'_{-\epsilon}(q)} \Big(1 - \underline{V}^{\rho}_{n}(\tilde{x}, \bar{q}(\tilde{x}, q))\Big)\hat{T}(\tilde{x} \,|\, \hat{x}, \hat{u}), \tag{4.6.2}$$

with $q \notin F$, $\hat{u} = \rho_{H-n-1}(\hat{x}, q)$, $\bar{V}^{\rho}_{n+1}(\hat{x}, q)$ as in (4.2.6), $\underline{q}$ as in (4.2.7), $\hat{X}'_{\epsilon}(q)$ as in (4.2.27), $\underline{V}^{\rho}_{n+1}(\hat{x}, q)$ as in (4.2.14), $\bar{q}$ as in (4.2.15), and $\hat{X}'_{-\epsilon}(q)$ as in (4.2.31).

The proof of Lemmas 4.6.2 can readily be derived based on (4.2.6), (4.2.7), (4.2.14) and (4.2.15). Now, I am ready to show the results of Theorem 4.2.18 and 4.2.20.

Proof of Theorem 4.2.18 For the sake of clarity of the proof, let's define

$$f(\hat{x}(k), q(k)) = 1 - \sum_{\hat{x}(k+1)\in\hat{X}} \bar{V}^*_{H-k-1}\Big(\hat{x}(k+1), \underline{q}^*\big(\hat{x}(k+1), q(k)\big)\Big)\hat{T}\big(\hat{x}(k+1) \,\big|\, \hat{x}(k), \hat{u}(k)\big),$$

for $k \in [0, H-1]$ with $\bar{V}^*_{H-k-1}$ as in (4.2.12), $\underline{q}^*$ as in (4.2.13), and $\hat{u}(k) = \rho'_k(\hat{x}(k), q(k))$, and

$$g(\hat{x}(z-1), q(z-1)) = \hat{T}\big(\hat{x}(z) \,\big|\, \hat{x}(z-1), \rho'_{z-1}(\hat{x}(z-1), q(z-1))\big),$$

for $z \in [0, k]$. First, consider an initial state $(\hat{x}_0, \bar{q}_0)$. By expanding out $\bar{V}^{\rho'}_H(\hat{x}_0, \bar{q}_0)$ up to the time instant $k \in [0, H-1]$ with the help of (4.6.1), one has $1 - \bar{V}^{\rho'}_H(\hat{x}_0, \bar{q}_0) = \theta_1(k) + \theta_2(k)$, with

$$\begin{aligned}\theta_1(k) = {}& (1-\delta)^{k+1}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))}\Big(\sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q(1))}\Big(\ldots\Big(\sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))}\Big(\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))} f\big(\hat{x}(k), q(k)\big)\\ &\times g\big(\hat{x}(k-1), q(k-1)\big)\Big)g\big(\hat{x}(k-2), q(k-2)\big)\Big)\ldots\Big)g\big(\hat{x}(1), q(1)\big)\Big)g\big(\hat{x}(0), q(0)\big), \text{ and} \end{aligned} \tag{4.6.3}$$

$$\begin{aligned}\theta_2(k) = {}& \delta\Big(1 + (1-\delta)\Big(\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g\big(\hat{x}(0), q(0)\big)\Big) + (1-\delta)^2\Big(\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))}\Big(\sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q(1))} g\big(\hat{x}(1), q(1)\big)\Big)\\ &\times g\big(\hat{x}(0), q(0)\big)\Big) + \ldots + (1-\delta)^k\Big(\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))}\Big(\sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q(1))}\Big(\ldots\Big(\sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))}\Big(\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))}\\ &g\big(\hat{x}(k-1), q(k-1)\big)\Big)g\big(\hat{x}(k-2), q(k-2)\big)\Big)\ldots\Big)g\big(\hat{x}(1), q(1)\big)\Big)g\big(\hat{x}(0), q(0)\big)\Big)\Big). \end{aligned} \tag{4.6.4}$$

Let us choose

$$\hat{x}^*(k) := \arg\max_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))} f(\hat{x}(k), q(k)),$$

with $q(k-1) \in \tilde{Q}_{k-1}(x_0)$, $\hat{X}'_\epsilon(q(k-1))$ as in (4.2.27), and $q^*(k) = \underline{q}(q(k-1), \hat{x}^*(k))$ with $\underline{q}$ as in (4.2.7). Then, one has

$$\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))} f\big(\hat{x}(k), q(k)\big)g\big(\hat{x}(k-1), q(k-1)\big) \leq f\big(\hat{x}^*(k), q^*(k)\big)\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))} g\big(\hat{x}(k-1), q(k-1)\big).$$

Thus, proceed from (4.6.3), one has

$$\begin{aligned}\theta_1(k) \leq {}& (1-\delta)^{k+1}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))}\Big(\sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q(1))}\Big(\ldots\Big(\sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))}\Big(\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))} g\big(\hat{x}(k-1),\\ &q(k-1)\big)\Big)g\big(\hat{x}(k-2), q(k-2)\big)\Big)\ldots\Big)g\big(\hat{x}(1), q(1)\big)\Big)g\big(\hat{x}(0), q(0)\big)f\big(\hat{x}^*(k), q^*(k)\big). \end{aligned} \tag{4.6.5}$$

Next, let's select

$$\hat{x}^*(k-1) = \underset{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))}{\arg\max} \sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))} g\big(\hat{x}(k-1), q(k-1)\big), \tag{4.6.6}$$

with $q(k-2) \in \tilde{Q}_{k-2}(x_0)$, and

$$q^*(k-1) = \underline{q}(q(k-2), \hat{x}^*(k-1)), \tag{4.6.7}$$

with $\underline{q}$ as in (4.2.7). With (4.6.6) and (4.6.7), one has

$$\begin{aligned}&\sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))}\Big(\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))} g\big(\hat{x}(k-1), q(k-1)\big)\Big) g\big(\hat{x}(k-2), q(k-2)\big)\\ &\qquad\le \Big(\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q^*(k-1))} g\big(\hat{x}^*(k-1), q^*(k-1)\big)\Big)\sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))} g\big(\hat{x}(k-2), q(k-2)\big).\end{aligned}$$

Therefore, from (4.6.5), one has

$$\begin{aligned}\theta_1(k) \le (1-\delta)^{k+1}&\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))}\Big(\sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q(1))}\Big(\ldots\Big(\sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))} g\big(\hat{x}(k-2), q(k-2)\big)\Big)\ldots\Big)\\ &\times g\big(\hat{x}(1), q(1)\big)\Big) g\big(\hat{x}(0), q(0)\big) f\big(\hat{x}^*(k), q^*(k)\big)\Big(\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q^*(k-1))} g\big(\hat{x}^*(k-1), q^*(k-1)\big)\Big).\end{aligned} \tag{4.6.8}$$

For all $z \in [2, k-1]$, one can choose $x^*(z-1)$ similar to (4.6.6) and $q^*(z-1)$ analogously to (4.6.7). Then, one has

$$\begin{aligned}&\sum_{\hat{x}(z-1)\in\hat{X}'_\epsilon(q(z-2))}\Big(\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q(z-1))} g\big(\hat{x}(z-1), q(z-1)\big)\Big) g\big(\hat{x}(z-2), q(z-2)\big)\\ &\le \Big(\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q^*(z-1))} g\big(\hat{x}^*(z-1), q^*(z-1)\big)\Big)\sum_{\hat{x}(z-1)\in\hat{X}'_\epsilon(q(z-2))} g\big(\hat{x}(z-2), q(z-2)\big).\end{aligned} \tag{4.6.9}$$

Therefore, continuing from (4.6.8) with (4.6.9) for all $z \in [2, k-1]$, one has

$$\theta_1(k) \le (1-\delta)^{k+1}\Big(\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g\big(\hat{x}(0), q(0)\big)\Big)\prod_{z=2}^{k}\Big(\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q^*(z-1))} g\big(\hat{x}^*(z-1), q^*(z-1)\big)\Big) f\big(\hat{x}^*(k), q^*(k)\big). \tag{4.6.10}$$

Similar to the idea of going from (4.6.5) to (4.6.10) with $x^*(z-1)$ and $q^*(z-1)$ for all $z \in [2, k-1]$, starting from (4.6.4), one has

$$\begin{aligned}\theta_2(k) \le \delta\Big(1 &+ (1-\delta)\Big(\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g(\hat{x}(0), q(0))\Big) + (1-\delta)^2\Big(\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g(\hat{x}(0), q(0))\Big)\Big(\sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q^*(1))} g(\hat{x}^*(1), q^*(1))\Big)\\ &+ \ldots + (1-\delta)^k\Big(\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g(\hat{x}(0), q(0))\Big)\prod_{z=2}^{k}\Big(\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q^*(z-1))} g(\hat{x}^*(z-1), q^*(z-1))\Big)\Big).\end{aligned} \tag{4.6.11}$$

Finally. combining (4.6.10) and (4.6.11), one has

$$1 - \bar{V}_H^{\rho'}(\hat{x}_0, \bar{q}_0) \le (1-\delta)^{k+1}\Big(\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g(\hat{x}(0), q(0))\Big)\prod_{z=2}^{k}\Big(\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q^*(z-1))} g(\hat{x}^*(z-1), q^*(z-1))\Big)$$

$$\times f(\hat{x}^*(k), q^*(k)) + \delta\Big(1 + \sum_{j=1}^{k}(1-\delta)^j \sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g(\hat{x}(0), q(0)) \prod_{z=2}^{j}\Big(\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q^*(z-1))} g(\hat{x}^*(z-1), q^*(z-1))\Big)\Big).$$

Note that $\bar{\omega}'_k = \big(\hat{x}(0), q(0), \rho_0(\hat{x}(0), q(0)), \hat{x}^*(1), q^*(1), \rho_1(\hat{x}^*(1), q^*(1)), \ldots, \hat{x}^*(k), q^*(k)\big)$ is one of the history paths as in Definition 4.2.16 up to the time instant k. By applying the history-based supervisor as in Definition 4.2.17, one can ensure that for an arbitrary path $\bar{\omega}_k$, one has

$$\begin{aligned}&(1-\delta)^{k+1}\prod_{z=1}^{k}\Big(\sum_{\bar{\omega}_{\hat{x}k}(z)\in\hat{X}'_\epsilon(\bar{\omega}_{qk}(z-1))} g(\bar{\omega}_{\hat{x}k}(z-1), \bar{\omega}_{qk}(z-1))\Big) f(\bar{\omega}_{\hat{x}k}(k), \bar{\omega}_{qk}(k))\\ &+ \delta\Big(1 + \sum_{j=1}^{k}(1-\delta)^j \prod_{z=1}^{j}\Big(\sum_{\bar{\omega}_{\hat{x}k}(z)\in\hat{X}'_\epsilon(\bar{\omega}_{qk}(z-1))} g(\bar{\omega}_{\hat{x}k}(z-1), \bar{\omega}_{qk}(z-1))\Big)\Big) \leq \eta.\end{aligned}$$

Therefore, one gets $\bar{V}_H^{\rho'}(\hat{x}_0, \bar{q}_0) \geq 1-\eta$ with the supervisor as in Definition 4.2.17. According to Theorem 4.2.11, one has $\mathbb{P}_{\tilde{\mathbf{C}}_{\rho'}\times\mathfrak{D}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\} \geq \bar{V}_H^{\rho'}(\hat{x}_0, \bar{q}_0)$ so that $\mathbb{P}_{\mathfrak{D}}\Big\{y_{\omega H} \models \mathcal{A}\Big\} \geq 1-\eta$, which completes the proof. ■

Proof of Theorem 4.2.20 For the sake of clarity of the proof, let's define

$$\begin{aligned}&r(\hat{x}(k), q(k))\\ &=1 - \sum_{\hat{x}(k+1)\in\hat{X}} \underline{V}_{*,H-k-1}\Big(\hat{x}(k+1), \bar{q}_*\big(\hat{x}(k+1), q(k)\big)\Big)\hat{T}\big(\hat{x}(k+1)\,\big|\,\hat{x}(k), \hat{u}(k)\big),\end{aligned}$$

for $k \in [0, H-1]$ with $\underline{V}^*_{H-k-1}$ as in (4.2.18), $\bar{q}_*$ as in (4.2.19), and $\hat{u}(k) = \rho'_k(\hat{x}(k), q(k))$, and

$$g(\hat{x}(z-1), q(z-1)) = \hat{T}\big(\hat{x}(z)\,\big|\,\hat{x}(z-1), \rho'_{z-1}(\hat{x}(z-1), q(z-1))\big),$$

for $z \in [0, k]$. Consider an initial state $(\hat{x}_0, \bar{q}_0)$. Then, with the help of (4.6.2), $\underline{V}_H^{\rho'}(\hat{x}_0, \bar{q}_0)$ is expanded up to the time instant $k \in [0, H-1]$ as

$$\begin{aligned}1-\underline{V}_H^{\rho'}(\hat{x}_0, \bar{q}_0) = (1-\delta)^{k+1}&\sum_{\hat{x}(1)\in\hat{X}'_{-\epsilon}(q(0))}\Big(\sum_{\hat{x}(2)\in\hat{X}'_{-\epsilon}(q(1))}\Big(\cdots\Big(\sum_{\hat{x}(k-1)\in\hat{X}'_{-\epsilon}(q(k-2))}\Big(\sum_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q(k-1))} r\big(\hat{x}(k), q(k)\big)\\ &\times g\big(\hat{x}(k-1), q(k-1)\big)\Big)g\big(\hat{x}(k-2), q(k-2)\big)\Big)\cdots\Big)g\big(\hat{x}(1), q(1)\big)\Big)g\big(\hat{x}(0), q(0)\big).\end{aligned} \tag{4.6.12}$$

Let us select

$$\hat{x}_*(k) := \operatorname*{arg\,min}_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q(k-1))} r(\hat{x}(k), q(k)),$$

with $q(k-1) \in \tilde{Q}_{k-1}(x_0)$, $\hat{X}'_{-\epsilon}(q(k-1))$ as in (4.2.31), and $q_*(k) = \bar{q}(q(k-1), \hat{x}_*(k))$ with $\bar{q}$ as in (4.2.15). Then, one has

$$\sum_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q(k-1))} r(\hat{x}(k), q(k))g(\hat{x}(k-1), q(k-1)) \geq r(\hat{x}_*(k), q_*(k))\sum_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q(k-1))} g(\hat{x}(k-1), q(k-1)).$$

Therefore, from (4.6.12), one has

$$1-\underline{V}_H^{\rho'}(\hat{x}_0,\bar{q}_0)\geq(1-\delta)^{k+1}\sum_{\hat{x}(1)\in\hat{X'}_{-\epsilon}(q(0))}\Big(\sum_{\hat{x}(2)\in\hat{X'}_{-\epsilon}(q(1))}\Big(\dots\Big(\sum_{\hat{x}(k-1)\in\hat{X'}_{-\epsilon}(q(k-2))}\Big(\sum_{\hat{x}(k)\in\hat{X'}_{-\epsilon}(q(k-1))}g\big(\hat{x}(k-1)$$
$$,q(k-1)\big)\Big)g\big(\hat{x}(k-2),q(k-2)\big)\Big)\dots\Big)g\big(\hat{x}(1),q(1)\big)\Big)g\big(\hat{x}(0),q(0)\big)r\big(\hat{x}_*(k),q_*(k)\big). \quad (4.6.13)$$

Next, let us select

$$\hat{x}_*(k-1)=\underset{\hat{x}(k-1)\in\hat{X'}_{-\epsilon}(q(k-2))}{\arg\min}\sum_{\hat{x}(k)\in\hat{X'}_{-\epsilon}(q(k-1))}g\big(\hat{x}(k-1),q(k-1)\big), \quad (4.6.14)$$

with $q(k-2)\in\tilde{Q}_{k-2}(x_0)$, and

$$q_*(k-1)=\bar{q}(q(k-2),\hat{x}_*(k-1)), \quad (4.6.15)$$

with $\bar{q}$ as in (4.2.15). Then, one has

$$\begin{aligned}&\sum_{\hat{x}(k-1)\in\hat{X'}_{-\epsilon}(q(k-2))}\Big(\sum_{\hat{x}(k)\in\hat{X'}_{-\epsilon}(q(k-1))}g\big(\hat{x}(k-1),q(k-1)\big)\Big)g\big(\hat{x}(k-2),q(k-2)\big)\\ \geq&\Big(\sum_{\hat{x}(k)\in\hat{X'}_{-\epsilon}(q_*(k-1))}g\big(\hat{x}_*(k-1),q_*(k-1)\big)\Big)\sum_{\hat{x}(k-1)\in\hat{X'}_{-\epsilon}(q(k-2))}g\big(\hat{x}(k-2),q(k-2)\big).\end{aligned}$$

Thus, proceed from (4.6.13), one has

$$1-\underline{V}_H^{\rho'}(\hat{x}_0,\bar{q}_0)\geq(1-\delta)^{k+1}\sum_{\hat{x}(1)\in\hat{X'}_{-\epsilon}(q(0))}\Big(\sum_{\hat{x}(2)\in\hat{X'}_{-\epsilon}(q(1))}\Big(\dots\Big(\sum_{\hat{x}(k-1)\in\hat{X'}_{-\epsilon}(q(k-2))}g\big(\hat{x}(k-2),q(k-2)\big)\Big)\dots\Big)$$
$$\times\ g\big(\hat{x}(1),q(1)\big)\Big)g\big(\hat{x}(0),q(0)\big)\Big(\sum_{\hat{x}(k)\in\hat{X'}_{-\epsilon}(q_*(k-1))}g\big(\hat{x}_*(k-1),q_*(k-1)\big)\Big)r\big(\hat{x}_*(k),q_*(k)\big). \quad (4.6.16)$$

For all $z\in[2,k-1]$, one can select $x_*(z-1)$ similar to (4.6.14) and $q_*(z-1)$ similar to (4.6.15). Accordingly, one has

$$\begin{aligned}&\sum_{\hat{x}(z-1)\in\hat{X'}_{-\epsilon}(q(z-2))}\Big(\sum_{\hat{x}(z)\in\hat{X'}_{-\epsilon}(q(z-1))}g\big(\hat{x}(z-1),q(z-1)\big)\Big)g\big(\hat{x}(z-2),q(z-2)\big)\\ \geq&\Big(\sum_{\hat{x}(z)\in\hat{X'}_{-\epsilon}(q_*(z-1))}g\big(\hat{x}_*(z-1),q_*(z-1)\big)\Big)\sum_{\hat{x}(z-1)\in\hat{X'}_{-\epsilon}(q(z-2))}g\big(\hat{x}(z-2),q(z-2)\big). \quad (4.6.17)\end{aligned}$$

Then, with (4.6.17) for all $z\in[2,k-1]$ and (4.6.16), one has

$$\begin{aligned}&1-\underline{V}_H^{\rho'}(\hat{x}_0,\bar{q}_0)\\ \geq&(1-\delta)^{k+1}\Big(\sum_{\hat{x}(1)\in\hat{X'}_{-\epsilon}(q(0))}g(\hat{x}(0),q(0))\Big)\prod_{z=2}^{k}\Big(\sum_{\hat{x}(z)\in\hat{X'}_{-\epsilon}(q_*(z-1))}g(\hat{x}_*(z-1),q_*(z-1))\Big)r(\hat{x}_*(k),q_*(k)).\end{aligned}$$

Note that $\bar{\omega}'_k = \big(\hat{x}(0), q(0), \rho_0(\hat{x}(0), q(0)), \hat{x}_*(1), q_*(1), \rho_1(\hat{x}_*(1), q_*(1)) \ldots \hat{x}_*(k), q_*(k)\big)$ is one of history paths as in Definition 4.2.16 up to the time instant k, and the history-based supervisor as in Definition 4.2.19 ensures that for all history paths $\bar{\omega}_k$, one has

$$(1-\delta)^{k+1}\prod_{z=1}^{k}\Big(\sum_{\bar{\omega}_{\hat{x}k}(z)\in \hat{X}'_{-\epsilon}(\bar{\omega}_{qk}(z-1))} g(\bar{\omega}_{\hat{x}k}(z-1), \bar{\omega}_{qk}(z-1))\Big) r(\bar{\omega}_{\hat{x}k}(k), \bar{\omega}_{qk}(k)) \geq 1-\eta.$$

Therefore, one has $\underline{V}_H^{\rho'}(\hat{x}_0, \bar{q}_0) \leq \eta$ when applying the supervisor as in Definition 4.2.19. According to Theorem 4.2.13, one has $\mathbb{P}_{\tilde{\mathbf{C}}_{\rho'}\times\mathfrak{D}}\{\exists k \leq H, y_{\omega k} \models \mathcal{A}\} \leq \underline{V}_H^{\rho'}(\hat{x}_0, \bar{q}_0)$ so that $\mathbb{P}_{\mathfrak{D}}\Big\{y_{\omega H} \models \mathcal{A}\Big\} \leq \eta$, which completes the proof. ■

4.7 Proof fo Statements in Section 4.3

Before showing the proof for Theorem 4.3.5 and 4.3.7, some additional definitions and lemmas are required. First, we define the *n-steps reachable state set* of a DFA with respect to a gDTSG as follows.

Definition 4.7.1. (n-steps Reachable State Set) *Consider a gDTSG* $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ *and a DFA* $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ *with a labeling function* $L : Y \to \Pi$. *An* n-steps reachable state set $\tilde{Q}_n(x_0)$ *of* $\mathcal{A}$ *with respect to an initial state* $x_0 \in X_0$ *is recursively defined as*

$$\tilde{Q}_0(x_0) = \Big\{q \in Q \,\Big|\, q = \tau(q_0, L\circ h(x_0))\Big\},$$
$$\tilde{Q}_n(x_0) = \Big\{q \in Q \,\Big|\, \exists q' \in \tilde{Q}_{n-1}(x_0), \sigma \in \Pi \ s.t.\ q = \tau(q', \sigma)\Big\},$$

with $n \in \mathbb{N}_{>0}$.

Additionally, the following lemmas are also required for showing the results of Theorem 4.3.5 and 4.3.7.

Lemma 4.7.2. *Given a gDTSG* $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ *and its finite abstraction* $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ *with* $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$, *and a DFA* $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ *modeling the desired safety property. Given a Markov policy* $\rho = (\rho_0, \rho_1, \ldots, \rho_{H-1})$ *over the product gDTSG* $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ *within the time horizon* $[0, H]$, *one has*

$$1-\underline{V}_{n+1}^{\rho,\lambda^*(\rho)}(\hat{x}, q) = (1-\delta) \min_{\lambda_{H-n-1}\in\Lambda} \sum_{\tilde{x}\in\hat{X}'_{-\epsilon}(q)} \Big(1-\underline{V}_n^{\rho,\lambda^*(\rho)}(\tilde{x}, \bar{q}(\tilde{x}, q))\Big)\hat{T}(\tilde{x} \,|\, \hat{x}, \hat{u}, \hat{w}), \tag{4.7.1}$$

with $q \notin F$, $\hat{u} = \rho_{H-n-1}(\hat{x}, q)$, $\hat{w} = \lambda_{H-n-1}(\hat{x}, q, \hat{u})$, $\underline{V}_{n+1}^{\rho,\lambda^*(\rho)}(\hat{x}, q)$ *as in* (3.3.15), $\lambda^*(\rho)$ *as in* (3.3.18), $\bar{q}$ *as in* (3.3.17), *and* $\hat{X}'_{-\epsilon}(q)$ *as in Definition 4.3.4.*

Lemma 4.7.3. *Given a gDTSG $\mathfrak{D} = (X, U, W, X_0, T, Y, h)$ and its finite abstraction $\widehat{\mathfrak{D}} = (\hat{X}, \hat{U}, \hat{W}, \hat{X}_0, \hat{T}, Y, \hat{h})$ with $\widehat{\mathfrak{D}} \preceq^{\delta}_{\epsilon} \mathfrak{D}$, and a DFA $\mathcal{A} = (Q, q_0, \Pi, \tau, F)$ modeling the desired safety property.. Given a Markov policy $\rho = (\rho_0, \rho_1, \ldots, \rho_{H-1})$ over the product gDTSG $\widehat{\mathfrak{D}} \otimes \mathcal{A}$ within the time horizon $[0, H]$, one has*

$$1-\bar{V}^{\rho,\lambda_*(\rho)}_{n+1}(\hat{x}, q) = (1-\delta) \max_{\lambda_{H-n-1}\in\Lambda} \sum_{\tilde{x}\in\hat{X}'_{\epsilon}(q)} \Big(1-\bar{V}^{\rho,\lambda_*(\rho)}_{n}(\tilde{x}, \underline{q}(\tilde{x}, q))\Big)\hat{T}(\tilde{x} \,|\, \hat{x}, \hat{u}, \hat{w}), \tag{4.7.2}$$

with $q \notin F$, $\hat{u} = \rho_{H-n-1}(\hat{x}, q)$, $\hat{w} = \lambda_{H-n-1}(\hat{x}, q, \hat{u})$, $\bar{V}^{\rho,\lambda_*(\rho)}_{n+1}(\hat{x}, q)$ as in (3.3.1), $\lambda_*(\rho)$ as in (3.3.5), $\underline{q}$ as in (3.3.3), and $\hat{X}'_{\epsilon}(q)$ as in Definition 4.3.6.

The proof of Lemmas 4.7.2 and 4.7.3 can readily be derived with the help of (3.3.15), (3.3.17), (3.3.1), and (3.3.3). Now, I am ready to show the results for Theorem 4.3.5 and 4.3.7.

Proof of Theorem 4.3.5 For the sake of clarity of the proof, let's define

$$\begin{aligned} &r(\hat{x}(k), q(k), \hat{w}(k)) \\ &= 1-\max_{\lambda_k\in\Lambda} \sum_{\hat{x}(k+1)\in\hat{X}} \underline{V}_{*,H-k-1}\Big(\hat{x}(k+1), \bar{q}_*\big(\hat{x}(k+1), q(k)\big)\Big)\hat{T}\big(\hat{x}(k+1) \,|\, \hat{x}(k), \hat{u}(k), \hat{w}(k)\big), \end{aligned}$$

for $k \in [0, H-1]$ with $\underline{V}_{*,H-k-1}$ and $\bar{q}_*$ as in (3.3.22) and (3.3.24), respectively, $\hat{u}(k) = \rho'_k(\hat{x}(k), q(k))$, and $\hat{w}(k) = \lambda_k(\hat{x}(k), q(k), \hat{u}(k))$ and

$$g(\hat{x}(z-1), q(z-1), \hat{w}(z-1)) = \hat{T}\big(\hat{x}(z) \,|\, \hat{x}(z-1), \rho'_{z-1}(\hat{x}(z-1), q(z-1)), \hat{w}(z-1)\big),$$

for $z \in [0, k]$. Consider an initial state $(\hat{x}_0, \bar{q}_0)$, with $\hat{x}_0 \in \hat{X}_0$. By leveraging (4.7.1), we expand out $\underline{V}^{\rho',\lambda^*(\rho')}_{H}(\hat{x}_0, \bar{q}_0)$ up to the time instant $k \in [0, H-1]$ as

$$\begin{aligned} 1-\underline{V}^{\rho',\lambda^*(\rho')}_{H}(\hat{x}_0, \bar{q}_0) &= (1-\delta)^{k+1} \min_{\lambda_0\in\Lambda} \sum_{\hat{x}(1)\in\hat{X}'_{-\epsilon}(q(0))} \Big(\min_{\lambda_1\in\Lambda} \sum_{\hat{x}(2)\in\hat{X}'_{-\epsilon}(q(1))} \Big(\ldots\Big(\min_{\lambda_{k-2}\in\Lambda} \\ &\sum_{\hat{x}(k-1)\in\hat{X}'_{-\epsilon}(q(k-2))} \Big(\min_{\lambda_{k-1}\in\Lambda} \sum_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q(k-1))} r\big(\hat{x}(k), q(k), \hat{w}(k)\big)g\big(\hat{x}(k-1), q(k-1), \hat{w}(k-1)\big)\Big) \\ &\times g\big(\hat{x}(k-2), q(k-2), \hat{w}(k-2)\big)\Big)\ldots\Big)g\big(\hat{x}(1), q(1), \hat{w}(1)\big)\Big)g\big(\hat{x}(0), q(0), \hat{w}(0)\big), \end{aligned} \tag{4.7.3}$$

with $\hat{w}(m) := \lambda_m(\hat{x}(m), q(m), \rho'_m(\hat{x}(m), q(m)))$, $m \in [0, k-1]$. Firstly, by selecting

$$\hat{x}_*(k) := \operatorname*{arg\,min}_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q(k-1))} r(\hat{x}(k), q(k), \hat{w}(k)),$$

with $q(k-1) \in \tilde{Q}_{k-1}(x_0)$, $\hat{X}'_{-\epsilon}(q(k-1))$ as in Definition 4.3.4, and $q_*(k) := \bar{q}(q(k-1), \hat{x}_*(k))$ with $\bar{q}$ as in (3.3.17), one has

$$\min_{\lambda_{k-1}\in\Lambda} \sum_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q(k-1))} r(\hat{x}(k), q(k), \hat{w}(k))g(\hat{x}(k-1), q(k-1), \hat{w}(k-1))$$

$$\geq r(\hat{x}_*(k), q_*(k), \hat{w}(k)) \min_{\lambda_{k-1}\in\Lambda} \sum_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q(k-1))} g(\hat{x}(k-1), q(k-1), \hat{w}(k-1)),$$

with $\hat{w}(k-1) = \lambda_{k-1}\Big(\hat{x}(k-1), q(k-1), \rho'_{k-1}(\hat{x}(k-1), q(k-1))\Big)$. Hence, proceed with (4.7.3), one gets

$$1 - \underline{V}_H^{\rho',\lambda^*(\rho')}(\hat{x}_0, \bar{q}_0) \geq (1-\delta)^{k+1} \min_{\lambda_0\in\Lambda} \sum_{\hat{x}(1)\in\hat{X}'_{-\epsilon}(q(0))} \Big(\min_{\lambda_1\in\Lambda} \sum_{\hat{x}(2)\in\hat{X}'_{-\epsilon}(q(1))} \Big(\dots\Big(\min_{\lambda_{k-2}\in\Lambda}$$

$$\sum_{\hat{x}(k-1)\in\hat{X}'_{-\epsilon}(q(k-2))} \Big(\min_{\lambda_{k-1}\in\Lambda} \sum_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q(k-1))} g\big(\hat{x}(k-1), q(k-1), \hat{w}(k-1)\big)\Big)\big(\hat{x}(k-2)$$

$$, q(k-2), \hat{w}(k-2)\big)\Big)\dots\Big) g\big(\hat{x}(1), q(1), \hat{w}(1)\big)\Big) g\big(\hat{x}(0), q(0), \hat{w}(0)\big) r\big(\hat{x}_*(k), q_*(k), \hat{w}(k)\big). \quad (4.7.4)$$

with $\hat{w}(m) := \lambda_m(\hat{x}(m), q(m), \rho'_m(\hat{x}(m), q(m)))$, $m \in [0, k-1]$. Secondly, we select

$$\hat{x}_*(k-1) = \arg\min_{\hat{x}(k-1)\in\hat{X}'_{-\epsilon}(q(k-2))} \min_{\lambda_{k-1}\in\Lambda} \sum_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q(k-1))} g\big(\hat{x}(k-1), q(k-1), \hat{w}(k-1)\big), \quad (4.7.5)$$

with $\hat{w}(k-1) = \lambda_{k-1}\Big(\hat{x}(k-1), \rho'_{k-1}(\hat{x}(k-1), q(k-1))\Big)$, $q(k-2) \in \tilde{Q}_{k-2}(x_0)$, and

$$q_*(k-1) = \bar{q}(q(k-2), \hat{x}_*(k-1)), \quad (4.7.6)$$

with $\bar{q}$ as in (3.3.17). Then, one has

$$\min_{\lambda_{z''}\in\Lambda} \sum_{\hat{x}(z')\in\hat{X}'_{-\epsilon}(q(z''))} \Big(\min_{\lambda_{z'}\in\Lambda} \sum_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q(z'))} g\big(\hat{x}(z'), q(z'), \hat{w}(z')\big)\Big) g\big(\hat{x}(z''), q(z''), \hat{w}(z'')\big)$$

$$\geq \Big(\min_{\lambda_{z'}\in\Lambda} \sum_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q_*(z'))} g\big(\hat{x}_*(z'), q_*(z'), \hat{w}(z')\big)\Big) \min_{\lambda_{z''}\in\Lambda} \sum_{\hat{x}(z')\in\hat{X}'_{-\epsilon}(q(z''))} g\big(\hat{x}(z''), q(z''), \hat{w}(z'')\big).$$

with $z' := k-1$ and $z'' := k-2$. Thus, proceed from (4.7.4), one has

$$1 - \underline{V}_H^{\rho',\lambda^*(\rho')}(\hat{x}_0, \bar{q}_0) \geq (1-\delta)^{k+1} \min_{\lambda_0\in\Lambda} \sum_{\hat{x}(1)\in\hat{X}'_{-\epsilon}(q(0))} \Big(\min_{\lambda_1\in\Lambda} \sum_{\hat{x}(2)\in\hat{X}'_{-\epsilon}(q(1))} \Big(\dots\Big(\min_{\lambda_{k-2}\in\Lambda}$$

$$\sum_{\hat{x}(k-1)\in\hat{X}'_{-\epsilon}(q(k-2))} g\big(\hat{x}(k-2), q(k-2), \hat{w}(k-2)\big)\Big)\dots\Big) g\big(\hat{x}(1), q(1), \hat{w}(1)\big)\Big) g\big(\hat{x}(0), q(0), \hat{w}(0)\big)$$

$$\times \Big(\min_{\lambda_{k-1}\in\Lambda} \sum_{\hat{x}(k)\in\hat{X}'_{-\epsilon}(q_*(k-1))} g\big(\hat{x}_*(k-1), q_*(k-1), \hat{w}(k-1)\big)\Big) r\big(\hat{x}_*(k), q_*(k), \hat{w}(k)\big). \quad (4.7.7)$$

with $\hat{w}(m) := \lambda_m(\hat{x}(m), q(m), \rho'_m(\hat{x}(m), q(m)))$, $m \in [0, k-1]$. For all $z \in [2, k-1]$, by choosing $x_*(z-1)$ similar to (4.7.5) and $q_*(z-1)$ similar to (4.7.6), one obtains

$$\min_{\lambda_{k''}\in\Lambda} \sum_{\hat{x}(k')\in\hat{X}'_{-\epsilon}(q(k''))} \Big(\min_{\lambda_{k'}\in\Lambda} \sum_{\hat{x}(z)\in\hat{X}'_{-\epsilon}(q(k'))} g\big(\hat{x}(k'), q(k'), \hat{w}(k')\big)\Big) g\big(\hat{x}(k''), q(k''), \hat{w}(k'')\big)$$

$$\geq \Big(\min_{\lambda_{k'}\in\Lambda} \sum_{\hat{x}(z)\in\hat{X}'_{-\epsilon}(q_*(k'))} g\big(\hat{x}_*(k'), q_*(k'), \hat{w}(k')\big)\Big) \min_{\lambda_{k''}\in\Lambda} \sum_{\hat{x}(k')\in\hat{X}'_{-\epsilon}(q(k''))} g\big(\hat{x}(k''), q(k''), \hat{w}(k'')\big). \quad (4.7.8)$$

with with $k' := z-1$ and $k'' := z-2$, $\hat{w}(m) := \lambda_m\Big(\hat{x}(m), q(m), \rho'_m(\hat{x}(m), q(m))\Big)$, $m \in \{z-1, z-2\}$. Then, with (4.7.8) for all $z \in [2, k-1]$ and (4.7.7), one gets

$$1 - \underline{V}_H^{\rho',\lambda^*(\rho')}(\hat{x}_0, \bar{q}_0) \geq (1-\delta)^{k+1}\Big(\min_{\lambda_0 \in \Lambda} \sum_{\hat{x}(1) \in \hat{X}'_{-\epsilon}(q(0))} g(\hat{x}(0), q(0), \hat{w}(0))\Big)$$
$$\times \prod_{z=2}^{k}\Big(\min_{\lambda_{z-1}\in\Lambda} \sum_{\hat{x}(z)\in\hat{X}'_{-\epsilon}(q_*(z-1))} g(\hat{x}_*(z-1), q_*(z-1), \hat{w}(z-1))\Big) r(\hat{x}_*(k), q_*(k), \hat{w}(k)).$$

with $\hat{w}(m) := \lambda_m(\hat{x}(m), q(m), \rho'_m(\hat{x}(m), q(m)))$, $m \in [0, k-1]$. Note that $\bar{\omega}'_k := \big(\hat{x}(0), q(0), \rho_0(\hat{x}(0), q(0)), \hat{x}_*(1), q_*(1), \rho_1(\hat{x}_*(1), q_*(1)) \ldots \hat{x}_*(k), q_*(k)\big)$ is one of history paths of the memory state of the Safe-visor architecture up to the time instant k, and the supervisor as in Definition 4.3.4 ensures that for all such history paths $\bar{\omega}_k$, one has

$$(1-\delta)^{k+1}\prod_{z=1}^{k}\Big(\min_{\lambda_{z-1}\in\Lambda} \sum_{\bar{\omega}_{\hat{x}k}(z)\in\hat{X}'_{-\epsilon}(\bar{\omega}_{qk}(z-1))} g(\bar{\omega}_{\hat{x}k}(z-1),$$
$$\bar{\omega}_{qk}(z-1)), \hat{w}(z-1)\Big) r(\bar{\omega}_{\hat{x}k}(k), \bar{\omega}_{qk}(k), \hat{w}(k)) \geq 1-\eta.$$

with $\hat{w}(m) := \lambda_m(\hat{x}(m), q(m), \rho'_m(\hat{x}(m), q(m)))$, $m \in [0, k-1]$. Therefore, one has

$$\underline{V}_H^{\rho',\lambda^*(\rho')}(\hat{x}_0, \bar{q}_0) \leq \eta,$$

when applying the supervisor as in Definition 4.3.4. According to Theorem 3.3.10, one has $\mathbb{P}\Big\{y_{\omega H} \models \mathcal{A}\Big\} \leq \eta$, which completes the proof. ■

Proof of Theorem 4.3.7 For the sake of clarity of the proof, let's define

$$f(\hat{x}(k), q(k), \hat{w}(k)) :=$$
$$1 - \min_{\lambda_k\in\Lambda} \sum_{\hat{x}(k+1)\in\hat{X}} \bar{V}^*_{H-k-1}\Big(\hat{x}(k+1), \underline{q}^*\big(\hat{x}(k+1), q(k)\big)\Big)\hat{T}\big(\hat{x}(k+1) \,\big|\, \hat{x}(k), \hat{u}(k), \hat{w}(k)\big),$$

for $k \in [0, H-1]$ with $\bar{V}^*_{H-k-1}$ as in (3.3.9), $\underline{q}^*$ as in (3.3.11), $\hat{u}(k) = \rho'_k(\hat{x}(k), q(k))$, and $\hat{w}(k) = \lambda_k(\hat{x}(k), q(k), \hat{u}(k))$,

$$g(\hat{x}(z-1), q(z-1), \hat{w}(z-1)) := \hat{T}\big(\hat{x}(z) \,\big|\, \hat{x}(z-1), \rho'_{z-1}(\hat{x}(z-1), q(z-1)), \hat{w}(z-1)\big),$$

for $z \in [0, k]$. First, consider an initial state $(\hat{x}_0, \bar{q}_0)$. By expanding out $\bar{V}_H^{\rho',\lambda_*(\rho')}(\hat{x}_0, \bar{q}_0)$ up to the time instant $k \in [0, H-1]$ with the help of (4.6.1), one has $1 - \bar{V}_H^{\rho',\lambda_*(\rho')}(\hat{x}_0, \bar{q}_0) = \theta_1(k) + \theta_2(k)$, with

$$\theta_1(k) = (1-\delta)^{k+1}\max_{\lambda_0\in\Lambda} \sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} \Big(\max_{\lambda_1\in\Lambda} \sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q(1))} \Big(\ldots\Big(\max_{\lambda_{k-2}\in\Lambda} \sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))} \Big($$
$$\max_{\lambda_{k-1}\in\Lambda} \sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))} f\big(\hat{x}(k), q(k), \hat{w}(k)\big) g\big(\hat{x}(k-1), q(k-1), \hat{w}(k-1)\big)\Big) g\big(\hat{x}(k-2), q(k-2),$$

$$\hat{w}(k-2))\Big)\ldots\Big)g\big(\hat{x}(1),q(1),\hat{w}(1)\big)\Big)g\big(\hat{x}(0),q(0),\hat{w}(0)\big), \tag{4.7.9}$$

and

$$\theta_2(k)=\delta\Big(1+(1-\delta)\Big(\max_{\lambda_0\in\Lambda}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))}g\big(\hat{x}(0),q(0),\hat{w}(0)\big)\Big)+(1-\delta)^2\Big(\max_{\lambda_0\in\Lambda}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))}\Big($$
$$\max_{\lambda_1\in\Lambda}\sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q(1))}g\big(\hat{x}(1),q(1),\hat{w}(1)\big)\Big)g\big(\hat{x}(0),q(0),\hat{w}(0)\big)\Big)+\ldots+(1-\delta)^k\Big(\max_{\lambda_0\in\Lambda}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))}\Big($$
$$\max_{\lambda_1\in\Lambda}\sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q(1))}\Big(\ldots\Big(\max_{\lambda_{k-2}\in\Lambda}\sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))}\Big(\max_{\lambda_{k-1}\in\Lambda}\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))}g\big(\hat{x}(k-1),q(k-1),\hat{w}(k-1)\big)\Big)$$
$$\times g\big(\hat{x}(k-2),q(k-2),\hat{w}(k-2)\big)\Big)\ldots\Big)g\big(\hat{x}(1),q(1),\hat{w}(1)\big)\Big)g\big(\hat{x}(0),q(0),\hat{w}(0)\big)\Big)\Big), \tag{4.7.10}$$

with $\hat{w}(m)=\lambda_m(\hat{x}(m),q(m),\rho'_m(\hat{x}(m),q(m)))$, $m\in[0,k-1]$. Let us choose

$$\hat{x}^*(k):=\arg\max_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))}f(\hat{x}(k),q(k),\hat{w}(k)),$$

with $q(k-1)\in\tilde{Q}_{k-1}(x_0)$, $\hat{X}'_\epsilon(q(k-1))$ as Definition 4.3.6, and $q^*(k)=\underline{q}(q(k-1),\hat{x}^*(k))$ with $\underline{q}$ as in (3.3.3). Then, one has

$$\max_{\lambda_{k-1}\in\Lambda}\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))}f\big(\hat{x}(k),q(k),\hat{w}(k)\big)g\big(\hat{x}(k-1),q(k-1),\hat{w}(k-1)\big)$$
$$\leq f\big(\hat{x}^*(k),q^*(k),\hat{w}(k)\big)\max_{\lambda_{k-1}\in\Lambda}\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))}g\big(\hat{x}(k-1),q(k-1),\hat{w}(k-1)\big).$$

with $\hat{w}(k-1)=\lambda_{k-1}(\hat{x}(k-1),q(k-1),\rho'_{k-1}(\hat{x}(k-1),q(k-1)))$. Thus, proceed from (4.7.9), one has

$$\theta_1(k)\leq(1-\delta)^{k+1}\max_{\lambda_0\in\Lambda}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))}\Big(\max_{\lambda_1\in\Lambda}\sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q(1))}\Big(\ldots\Big(\max_{\lambda_{k-2}\in\Lambda}\sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))}\Big($$
$$\max_{\lambda_{k-1}\in\Lambda}\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))}g\big(\hat{x}(k-1),q(k-1),\hat{w}(k-1)\big)\Big)g\big(\hat{x}(k-2),q(k-2),\hat{w}(k-2)\big)\Big)\ldots\Big)$$
$$\times g\big(\hat{x}(1),q(1),\hat{w}(1)\big)\Big)g\big(\hat{x}(0),q(0),\hat{w}(0)\big)f\big(\hat{x}^*(k),q^*(k),\hat{w}(k)\big), \tag{4.7.11}$$

with $\hat{w}(m)=\lambda_m(\hat{x}(m),q(m),\rho'_m(\hat{x}(m),q(m)))$, $m\in[0,k-1]$. Next, let's select

$$\hat{x}^*(k-1)=$$
$$\arg\max_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))}\max_{\lambda_{k-1}\in\Lambda}\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))}g\big(\hat{x}(k-1),q(k-1),\hat{w}(k-1)\big), \tag{4.7.12}$$

with $\hat{w}(k-1)=\lambda_{k-1}(\hat{x}(k-1),q(k-1),\rho'_{k-1}(\hat{x}(k-1),q(k-1)))$, $q(k-2)\in\tilde{Q}_{k-2}(x_0)$, and

$$q^*(k-1)=\underline{q}(q(k-2),\hat{x}^*(k-1)), \tag{4.7.13}$$

with $\underline{q}$ as in (3.3.3). With (4.7.12) and (4.7.13), one has

$$\begin{aligned}&\max_{\lambda_{k-2}\in\Lambda}\sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))}\Big(\max_{\lambda_{k-1}\in\Lambda}\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q(k-1))} g\big(\hat{x}(k-1),q(k-1),\hat{w}(k-1)\big)\Big)g\big(\hat{x}(k-2),q(k-2),\hat{w}(k-2)\big)\\ \leq\ &\Big(\max_{\lambda_{k-1}\in\Lambda}\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q^*(k-1))} g\big(\hat{x}^*(k-1),q^*(k-1),\hat{w}(k-1)\big)\Big)\max_{\lambda_{k-2}\in\Lambda}\sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))} g\big(\hat{x}(k-2),q(k-2),\hat{w}(k-2)\big),\end{aligned}$$

with with $\hat{w}(k-1)=\lambda_{k-1}(\hat{x}(k-1),q(k-1),\rho'_{k-1}(\hat{x}(k-1),q(k-1)))$, and $\hat{w}(k-2)=\lambda_{k-2}(\hat{x}(k-2),q(k-2),\rho'_{k-2}(\hat{x}(k-2),q(k-2)))$ Therefore, from (4.7.11), one has

$$\begin{aligned}\theta_1(k)\leq&(1-\delta)^{k+1}\max_{\lambda_0\in\Lambda}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))}\Big(\max_{\lambda_1\in\Lambda}\sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q(1))}\Big(\dots\Big(\max_{\lambda_{k-2}\in\Lambda}\sum_{\hat{x}(k-1)\in\hat{X}'_\epsilon(q(k-2))} g\big(\hat{x}(k-2)\\ &,q(k-2),\hat{w}(k-2)\big)\Big)\dots\Big)g\big(\hat{x}(1),q(1),\hat{w}(1)\big)\Big)g\big(\hat{x}(0),q(0),\hat{w}(0)\big)f\big(\hat{x}^*(k),q^*(k),\hat{w}(k)\big)\\ &\Big(\max_{\lambda_{k-1}\in\Lambda}\sum_{\hat{x}(k)\in\hat{X}'_\epsilon(q^*(k-1))} g\big(\hat{x}^*(k-1),q^*(k-1),\hat{w}(k-1)\big)\Big).\end{aligned}\tag{4.7.14}$$

with $\hat{w}(m)=\lambda_m(\hat{x}(m),q(m),\rho'_m(\hat{x}(m),q(m)))$, $m\in[0,k-1]$. For all $z\in[2,k-1]$, one can choose $x^*(z-1)$ similar to (4.7.12) and $q^*(z-1)$ analogously to (4.7.13). Then, one has

$$\begin{aligned}&\max_{\lambda_{z-2}\in\Lambda}\sum_{\hat{x}(z-1)\in\hat{X}'_\epsilon(q(z-2))}\Big(\max_{\lambda_{z-1}\in\Lambda}\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q(z-1))} g\big(\hat{x}(z-1),q(z-1),\hat{w}(z-1)\big)\Big)g\big(\hat{x}(z-2),q(z-2),\hat{w}(z-2)\big)\\ \leq\ &\Big(\max_{\lambda_{z-1}\in\Lambda}\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q^*(z-1))} g\big(\hat{x}^*(z-1),q^*(z-1),\hat{w}(z-1)\big)\Big)\max_{\lambda_{z-2}\in\Lambda}\sum_{\hat{x}(z-1)\in\hat{X}'_\epsilon(q(z-2))} g\big(\hat{x}(z-2),q(z-2),\hat{w}(z-2)\big),\end{aligned}\tag{4.7.15}$$

with $\hat{w}(m)=\lambda_k(\hat{x}(m),q(m),\rho'_m(\hat{x}(m),q(m)))$, $m\in\{z-1,z-2\}$. Therefore, continuing from (4.7.14) with (4.7.15) for all $z\in[2,k-1]$, one has

$$\begin{aligned}\theta_1(k)\leq\ &(1-\delta)^{k+1}\Big(\max_{\lambda_0\in\Lambda}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g\big(\hat{x}(0),q(0),\hat{w}(0)\big)\Big)\\ &\times\prod_{z=2}^{k}\Big(\max_{\lambda_{z-1}\in\Lambda}\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q^*(z-1))} g\big(\hat{x}^*(z-1),q^*(z-1),\hat{w}(z-1)\big)\Big)f\big(\hat{x}^*(k),q^*(k),\hat{w}(k)\big).\end{aligned}\tag{4.7.16}$$

with $\hat{w}(m)=\lambda_m(\hat{x}(m),q(m),\rho'_m(\hat{x}(m),q(m)))$, $m\in[0,k-1]$. Similar to the idea of going from (4.7.11) to (4.7.16) with $x^*(z-1)$ and $q^*(z-1)$ for all $z\in[2,k-1]$, starting from (4.7.10), one has

$$\begin{aligned}\theta_2(k)\leq\ &\delta\Big(1+(1-\delta)\Big(\max_{\lambda_0\in\Lambda}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g(\hat{x}(0),q(0),\hat{w}(0))\Big)+(1-\delta)^2\Big(\max_{\lambda_0\in\Lambda}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g(\hat{x}(0),q(0),\hat{w}(0))\Big)\\ &\times\Big(\max_{\lambda_1\in\Lambda}\sum_{\hat{x}(2)\in\hat{X}'_\epsilon(q^*(1))} g(\hat{x}^*(1),q^*(1),\hat{w}(1))\Big)+\dots+(1-\delta)^k\Big(\max_{\lambda_0\in\Lambda}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g(\hat{x}(0),q(0),\hat{w}(0))\Big)\\ &\times\prod_{z=2}^{k}\Big(\max_{\lambda_{z-1}\in\Lambda}\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q^*(z-1))} g(\hat{x}^*(z-1),q^*(z-1),\hat{w}(z-1))\Big)\Big).\end{aligned}\tag{4.7.17}$$

with $\hat{w}(m) = \lambda_m(\hat{x}(m), q(m), \rho'_m(\hat{x}(m), q(m)))$, $m \in [0, k-1]$. Finally. combining (4.7.16) and (4.7.17), one has

$$\begin{aligned}
&1 - \bar{V}_H^{\rho',\lambda_*(\rho')}(\hat{x}_0, \bar{q}_0)\\
\leq&(1-\delta)^{k+1}\Big(\max_{\lambda_0\in\Lambda}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g(\hat{x}(0), q(0), \hat{w}(0))\Big)\prod_{z=2}^{k}\Big(\max_{\lambda_{z-1}\in\Lambda}\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q^*(z-1))} g(\hat{x}^*(z-1), q^*(z-1),\\
&\hat{w}(z-1))\Big)f(\hat{x}^*(k), q^*(k), \hat{w}(k)) + \delta\Big(1 + \sum_{j=1}^{k}(1-\delta)^j \max_{\lambda_0\in\Lambda}\sum_{\hat{x}(1)\in\hat{X}'_\epsilon(q(0))} g(\hat{x}(0), q(0), \hat{w}(0))\\
&\times\prod_{z=2}^{j}\Big(\max_{\lambda_{z-1}\in\Lambda}\sum_{\hat{x}(z)\in\hat{X}'_\epsilon(q^*(z-1))} g(\hat{x}^*(z-1), q^*(z-1), \hat{w}(z-1))\Big)\Big).
\end{aligned}$$

Note that $\bar{\omega}'_k = \big(\hat{x}(0), q(0), \rho_0(\hat{x}(0), q(0)), \hat{x}^*(1), q^*(1), \rho_1(\hat{x}^*(1), q^*(1)), \ldots, \hat{x}^*(k), q^*(k)\big)$ is one of the history paths of the memory state of the Safe-visor architecture up to the time instant k. By applying the history-based supervisor as in Definition 4.3.6, one can ensure that for an arbitrary path $\bar{\omega}_k$, one has

$$\begin{aligned}
&(1-\delta)^{k+1}\prod_{z=1}^{k}\Big(\max_{\lambda_{z-1}\in\Lambda}\sum_{\bar{\omega}_{\hat{x}k}(z)\in\hat{X}'_\epsilon(\bar{\omega}_{qk}(z-1))} g(\bar{\omega}_{\hat{x}k}(z-1), \bar{\omega}_{qk}(z-1)), \hat{w}(z-1)\Big)f(\bar{\omega}_{\hat{x}k}(k), \bar{\omega}_{qk}(k), \hat{w}(k))\\
&+\delta\Big(1 + \sum_{j=1}^{k}(1-\delta)^j\prod_{z=1}^{j}\Big(\max_{\lambda_{z-1}\in\Lambda}\sum_{\bar{\omega}_{\hat{x}k}(z)\in\hat{X}'_\epsilon(\bar{\omega}_{qk}(z-1))} g(\bar{\omega}_{\hat{x}k}(z-1), \bar{\omega}_{qk}(z-1), \hat{w}(z-1))\Big)\Big) \leq \eta,
\end{aligned}$$

with $\hat{w}(m) := \lambda_m(\hat{x}(m), q(m), \rho'_m(\hat{x}(m), q(m)))$, $m \in [0, k-1]$. Therefore, one gets

$$\bar{V}_H^{\rho',\lambda_*(\rho')}(\hat{x}_0, \bar{q}_0) \geq 1 - \eta,$$

with the supervisor as in Definition 4.3.6. According to Theorem 3.3.4, one has $\mathbb{P}_{\mathfrak{D}}\Big\{y_{\omega H} \models \mathcal{A}\Big\} \geq 1 - \eta$, which completes the proof.

■

5 Abstraction-free Controller Synthesis against ω-Regular Properties

5.1 Introduction

In Chapter 3 and 4, abstraction-based approaches for constructing the Safe-visor architecture are proposed. These approaches require building finite abstractions of the original systems, which sometimes encounters so-called *the curse of dimensionality*, leading to exponential growth in computational complexity with the dimension of the systems. In this chapter, abstraction-free methodologies are proposed for designing controllers enforcing ω-regular properties [183] over discrete-time linear control systems affected by bounded disturbances. As a key insight, these controllers are constructed based on so-called *hybrid controlled invariant (HCI) sets* over the state set of a hybrid system. These controllers and sets are used for constructing the Safe-visor architecture following the basic idea of state-based approaches, which is described as in Section 1.3.

5.1.1 Related Works

In the computer science community, reactive synthesis [153] was introduced to synthesize controllers enforcing high-level logical properties, see e.g. [153, 139, 60]. However, these results are only applicable to systems with finite state and input sets. As for systems with continuous state and input sets, *Hamilton-Jacobi-based (HJ-based) methods* [16, 140] are applicable to synthesize controllers against invariance and reachability properties. However, it is challenging to apply these methods to enforce high-level logic properties, in general. To cope with high-level logic properties, *discretization-based approaches* have been proposed in the past two decades. Among them, *symbolic techniques* (see e.g. [209, 154, 161]) are widely applied for various types of properties, such as (safe-)LTL (see e.g. [164, 181]) and ω-regular properties (see e.g. [54, 104]). These techniques require the construction of symbolic models (a.k.a. finite abstractions) with finite state and input sets for the original systems. Since the finite state and input sets are constructed by gridding the original sets, the number of discrete states and inputs grow exponentially with respect to the dimensions of state and input sets, respectively. This issue is known as the *curse of dimensionality*, which is one of the main challenges of discretization-based approaches. Some recent results alleviate this issue partially by constructing abstractions in a compositional manner (see e.g. [180, 208, 155]), by leveraging a counterexample-guided abstraction refinement framework [202], or by applying a specification-guided framework (see e.g. [219, 53, 138]). However, these results require either specific properties of the systems (e.g. dissipativity, mixed-monotonicity, etc.), or additional assumptions regarding the properties (e.g. properties can be decomposed into several simpler ones).

Recently, other discretization-based approaches, which are developed based on interval analysis (referred to as *interval-analysis-based approaches*), have been proposed to enforce invariance properties [124], reach-and-stay properties [126], and properties modeled by deterministic Büchi automaton [127], which are subsets of ω-regular properties. Despite improvements in terms of space complexity compared with the symbolic techniques, interval-analysis-based approaches also suffer from the curse of dimensionality, since discretization of the state sets is still needed. Additionally, they are only applicable to systems without exogenous disturbances.

To avoid the curse of dimensionality introduced by discretizing the state and input sets, some *discretization-free approaches* have been proposed. Results in [23] propose a set-based approach to enforce invariance properties (i.e. the systems are expected to stay within a set). This result is further extended in [158, 30, 165, 129] in terms of termination and compositionality. *Control barrier functions* (CBF) [199] are also used to enforce invariance properties (e.g. [87, 147, 88, 8]), properties described by deterministic finite automata [85, 10], deterministic Büchi automata [86], LTL [176], and ω-regular properties [9]. Unfortunately, constructing valid CBFs is an NP-hard problem in general [46].

5.1.2 Contributions

In this chapter, new discretization-free approaches are proposed for synthesizing controllers against ω-regular properties over discrete-time linear control systems affected by bounded disturbances. Specifically, new set-based approaches are developed, which leverage new iterative schemes to construct these controllers by computing so-called *hybrid controlled invariant (HCI) sets.* Compared with existing set-based approaches (e.g., [165]) for computing control invariant sets over *continuous sets* against invariance properties, the technical contributions of this chapter are threefold:

- Here, ω-regular properties are considered instead of invariance properties. In particular, I elaborate on how to exploit the structure of the automata modeling the desired ω-regular properties to compute the HCI sets;
- I show the convergence of the new iterative schemes over *hybrid* sets. Moreover, rigorous results are provided for ensuring the termination of the iterative schemes over *hybrid* sets within a finite number of steps;
- A worst-case space and time complexities analysis is provided for the proposed set-based methods over hybrid sets. In particular, I show the relation between the complexities of these methodologies and the structure of the automata representing the desired ω-regular properties.

Moreover, in comparison with those discretization-based approaches, discretization over the state and input sets is not needed so that the proposed approaches can be efficient in some cases in terms of computation time (c.f. Section 5.5.4). Compared with the discretization-free approaches based on CBFs, the proposed methods are more systematic in the sense that given any linear control systems and desired ω-regular properties, one can readily compute the HCI sets and construct their corresponding controllers by leveraging the results in this chapter. Meanwhile, the results in [9] tackle the synthesis problems by first decomposing the synthesis task for the original property into several simpler ones and then computing CBFs for each simpler task by solving a series of sum-of-square (SOS) optimization problems. For each SOS optimization problem, one needs to choose the forms of the potential CBFs to be polynomials of fixed degrees and fix the forms of their corresponding controllers heuristically, which requires much manual effort.

5.1.3 Problem Formulation

In this chapter, I focus on synthesizing controllers enforcing ω-regular properties over dtLCS as in Definition 2.3.1. Here, ω-regular properties is modeled by deterministic Streett automata (DSA), as introduced in Definition 2.5.2. To synthesize such controllers, a *labeling function* is needed, which is used to connect a system S as in (2.3.1) to a DSA $\mathcal{A}$.

Definition 5.1.1. (Labeling function) *Consider a dtLCS $S = (X, X_0, U, W, f)$ and a DSA $\mathcal{A} = (Q, q_0, \Pi, \delta, Acc)$. A measurable labeling function $L : X \to \Pi$ is defined as follows: given an infinite state sequence $\xi_x = (x(0), x(1), \ldots) \in X^\omega$ of system S, the word of ξ_x over Π is $L(\xi_x) = (\sigma_0, \sigma_1, \ldots, \sigma_k, \ldots)$, where $\sigma_k = L(x(k))$ for all $k \in \mathbb{N}$. Accordingly, one has $L(\xi_x) \models \mathcal{A}$ if $L(\xi_x) \in \mathcal{L}(\mathcal{A})$, and $S \models \mathcal{A}$, if $L(\xi_x) \models \mathcal{A}$ holds for all possible ξ_x of S.*

Note that in Definition 5.1.1, I slightly abuse the notation by applying the map $L(\cdot)$ over the domain X^ω, i.e. $L((x(0), x(1), \ldots)) = (L(x(0)), L(x(1)), \ldots)$. However, the distinction is clear from the context. It is also worth noting that the set of alphabet $\Pi = \{\sigma_1, \sigma_2, \ldots, \sigma_M\}$ along with the labeling function $L : X \to \Pi$ provide a partition of the state set $X = \cup_{j=1}^{M} X_j$, where $X_j := L^{-1}(\sigma_j)$. Finally, two additional definitions related to the *strongly connected components* [13] in a DSA are introduced.

Definition 5.1.2. *Consider a DSA $\mathcal{A} = (Q, q_0, \Pi, \delta, Acc)$. A set $Q_1 \subseteq Q$ is* strongly connected *if any arbitrary pair of states $q_a, q_b \in Q_1$ are mutually reachable, i.e. $\exists (q_a, \ldots, q_b) \in Q^{d_1}, (q_b, \ldots, q_a) \in Q^{d_2}$ with $d_1, d_2 \in \mathbb{N}$. A set $Q_1 \subseteq Q$ is a* strongly connected component *in $\mathcal{A}$ if Q_1 is strongly connected, and $\nexists Q_2 \subseteq Q$, with $Q_1 \subset Q_2$, such that Q_2 is strongly connected. Additionally, $SCC(\mathcal{A}) \subset 2^Q$ denotes the set of all strongly connected components in $\mathcal{A}$.*

Definition 5.1.3. *(*reduced DSA*) Consider a DSA $\mathcal{A} = (Q, q_0, \Pi, \delta, Acc)$. A* reduced DSA *of $\mathcal{A}$ with respect to a set $\bar{Q} \subset Q$ is defined as $\mathcal{A}_{rd}(\bar{Q}) := (Q', q_0, \Pi', \delta', Acc')$, with $Q' \subseteq Q$, $\Pi' \subseteq \Pi$, $\delta' \subseteq \delta$, and $Acc' \subseteq Acc$ such that $\forall Q_{scc} \in SCC(\mathcal{A}_{rd}(\bar{Q}))$, $\nexists q \in \bar{Q}$ such that $q \in Q_{scc}$.*

Intuitively, the reduced DSA $\mathcal{A}_{rd}(\bar{Q})$ is constructed such that it does not have any strongly connected component containing the state within the set $\bar{Q}$. To formulate the main problem, the following definitions are also needed, which are borrowed from [71].

Definition 5.1.4. *(*Hyperplane*) A hyperplane in $\mathbb{R}^n$ is a set*

$$\{x \in \mathbb{R}^n | a^\top x = b\}, \tag{5.1.1}$$

where $a \in \mathbb{R}^n$ is non-zero and $b \in \mathbb{R}$.

Definition 5.1.5. *A* Polytope *is a bounded set in the form of*

$$\mathcal{P} = \{x \in \mathbb{R}^n | Px \leq p\}, \tag{5.1.2}$$

with $P \in \mathbb{R}^{n_p \times n}$, $p \in \mathbb{R}^{n_p}$, and $n_p \in \mathbb{N}$, where the inequality in (5.1.2) *is component-wise. Accordingly,*

$$\mathsf{numh}(\mathcal{P}) := n_p, \tag{5.1.3}$$

denotes the number of hyperplanes defining $\mathcal{P}$, and denoted by $\mathcal{P}(n)$ the set of all polytopes in $\mathbb{R}^n$.

Definition 5.1.6. *(*P-collection*) A P-collection $\mathcal{U}$ is a finite collection of polytopes in $\mathbb{R}^n$, i.e.*

$$\mathcal{U} = \cup_{a=1}^{\mathsf{N_c}} \mathcal{P}_a,$$

where $\mathsf{N_c} \in \mathbb{N}$, and $\mathcal{P}_a = \{x \in \mathbb{R}^n | P_a x \leq p_a\}$ are polytopes, with $a \in [1, \mathsf{N_c}]$, $P_a \in \mathbb{R}^{n_{p,a} \times n}$, and $p_a \in \mathbb{R}^{n_{p,a}}$. Additionally, for a P-collection $\mathcal{U}$, one has

$$\mathsf{larg}(\mathcal{U}) := \max_{a \in [1, \mathsf{N_c}]} \mathsf{numh}(\mathcal{P}_a), \tag{5.1.4}$$

and

$$\mathsf{num}(\mathcal{U}) := \mathsf{N_c}, \tag{5.1.5}$$

with $\mathsf{numh}(\cdot)$ as in (5.1.3).

With all notions above, the main problem in this chapter is formulated as below.

Problem 5.1.7. *Consider a dtLCS $S = (X, X_0, U, W, f)$ as in* (2.3.1)*, a DSA $\mathcal{A} = (Q, q_0, \Pi, \delta, Acc)$, and a labeling function $L : X \to \Pi$ as in Definition 5.1.1. Synthesize a controller (if existing) to enforce the property modeled by $\mathcal{A}$ over S.*

For a better illustration of the theoretical results, the following running example is deployed throughout this chapter.

Example 5.1.8. *(Running example) Consider a dtLCS as in* (2.3.1)*, in which $A = \begin{bmatrix} 0.9990 & 0.1846 \\ -0.0074 & 0.5265 \end{bmatrix}$; $B = \begin{bmatrix} 1.0209; 7.3830 \end{bmatrix}$; $x(k) = [x_1(k); x_2(k)]$ is the state; $X_0 = [105, 110] \times [-10, 10]$ is the initial state set; $u(k) \in [-0.32,\ 0.68]$ denotes the input; and $w(k) \in [-0.18,\ 0.18]^2$ denotes the disturbances affecting the system. Here, an ω-regular property*

ψ, which is modeled by a DSA $\mathcal{A}$ as in Figure 5.1, is considered. The temporal logic formula[1] *for $\mathcal{A}$ is given by $G((p_2 \Rightarrow FGp_2) \wedge (\neg p3))$, which, in English, requires that: 1) if the system enters the region $X_2 := L^{-1}(p_2)$, it must eventually stay within X_2; and 2) the system should not reach the region $X_3 := L^{-1}(p_3)$.*

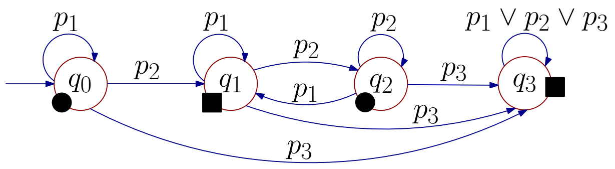


Figure 5.1: DSA $\mathcal{A}$ modeling ψ, with alphabet $\Pi = \{p_1, p_2, p_3\}$; labeling function $L : X \to \Pi$ with $L(x) = p_1$ when $x \in [105,\, 110] \times [-10, 10]$, $L(x) = p_2$ when $x \in (110, 115] \times [-10, 10]$, and $L(x) = p_3$ when $x \in \mathbb{R}^2 \backslash ([105, 115] \times [-10, 10])$; and accepting condition $\text{Acc} = \{\langle E_1, F_1\rangle, \langle E_2, F_2\rangle, \langle E_3, F_3\rangle\}$, in which $E_1 = \{q_3\}$, $F_1 = \emptyset$, $E_2 = \{q_1\}$, $F_2 = \{q_2\}$, $E_3 = \emptyset$, and $F_3 = \{q_0\}$. ■ and ● indicate the states that can be visited finitely and infinitely many times, respectively.

5.2 Controller Synthesis via Hybrid Controlled Invariant Sets

5.2.1 Product System

Consider a dtLCS S and a DSA $\mathcal{A} = (Q, q_0, \Pi, \delta, \text{Acc})$. To solve Problem 5.1.7, a product between a dtLCS S and a DSA $\mathcal{A}$ is required, which is formally defined as follows.

Definition 5.2.1. (Product of S and $\mathcal{A}$) *Consider a dtLCS $S = (X, X_0, U, W, f)$, a DSA $\mathcal{A} = (Q, q_0, \Pi,\, \delta, Acc)$, and a labeling function $L : X \to \Pi$. The product system between S and $\mathcal{A}$ is defined as*

$$S \otimes \mathcal{A} = (\underline{X}, \underline{X}_0, \underline{U}, \underline{W}, \underline{f}), \tag{5.2.1}$$

with state set $\underline{X} := \{(q, q', x) \in Q \times Q \times X | \exists \sigma \in \Pi, (q, \sigma, q') \in \delta,$ and $x \in L^{-1}(\sigma)\}$; the set of initial states $\underline{X}_0 := \{(q_0, q,\, x) \in \{q_0\} \times Q \times X_0 | \exists \sigma \in \Pi, (q_0, \sigma, q) \in \delta,$ with $x \in L^{-1}(\sigma)\} \subseteq \underline{X}$; the input set $\underline{U} := U$; and the disturbance set $\underline{W} := W$. The transition $\underline{f} : \underline{X} \times \underline{U} \times \underline{W} \to \underline{X}$ is defined as $\underline{x}' := \underline{f}(\underline{x}, u, w)$ with $\underline{x} = (q, q', x), \underline{x}' = (q', q'', x')$, $u \in \underline{U}$, and $w \in \underline{W}$ in which $x' = Ax + Bu + w$ and $(q', L(x'), q'') \in \delta$.

Consider the hybrid set $\underline{X}$ as in (5.2.1), and any set $\underline{X}' \subset \underline{X}$. The following definitions are also required in this chapter:

- *(Projection)* The projection of $\underline{X}'$ on X w.r.t. some $q, q' \in Q$ is denoted by

$$\underline{X}'(q, q') := \{x \in X | (q, q', x) \in \underline{X}'\}. \tag{5.2.2}$$

  Accordingly, $\big(q, q', \underline{X}'(q, q')\big) := \{(q_1, q_2, x) \in \underline{X}' \mid q_1 = q, q_2 = q'\}$.

[1]see [13, Section 5.1] for syntax and semantics of the formula.

- *(Hybrid Minkowski sum)* Consider a set $\mathsf{X} \subseteq X$. $\underline{X}' \oplus \mathsf{X}$ denotes the hybrid Minkowski sum between $\underline{X}'$ and X, which is defined as

$$\underline{X}' \oplus \mathsf{X} := \{(q, q', x) \in \underline{X} \mid \underline{X}'(q, q') \neq \emptyset, x \in \underline{X}'(q, q') + \mathsf{X}\}; \tag{5.2.3}$$

- *(ε-expansion set)* Consider an $\varepsilon \in \mathbb{R}_{\geq 0}$. The set $\underline{X}'_\varepsilon$ denotes the ε-expansion of $\underline{X}'$, which is defined as

$$\underline{X}'_\varepsilon := \underline{X}' \oplus \varepsilon \mathbb{B}^n; \tag{5.2.4}$$

- *(ε-contraction set)* Consider an $\varepsilon \in \mathbb{R}_{\geq 0}$. The set $\underline{X}'_{-\varepsilon}$ denotes the ε-contraction of $\underline{X}'$, which is defined as

$$\underline{X}'_{-\varepsilon} := \{(q, q', x) \in \underline{X} \mid \underline{X}'(q, q') \neq \emptyset, x \in \underline{X}'(q, q') - \varepsilon \mathbb{B}^n\}. \tag{5.2.5}$$

- *(ρ-contraction product)* Consider $\rho \in \mathbb{R}_{\geq 0}$.

$$(S \otimes \mathcal{A})_{-\rho} := (\underline{X}_{-\rho}, (\underline{X}_0)_{-\rho}, \underline{U} - \rho \mathbb{B}^m, \underline{W}, \underline{f}), \tag{5.2.6}$$

  denotes the ρ-contraction of $S \otimes \mathcal{A}$ as in (5.2.1).

- *(Distance)* Consider any $\underline{x}_1, \underline{x}_2 \in \underline{X}$, with $\underline{x}_1 = (q_1, q_1', x_1)$ and $\underline{x}_2 = (q_2, q_2', x_2)$. The distance between $\underline{x}_1$ and $\underline{x}_2$ is defined as

$$\mathsf{d}(\underline{x}_1, \underline{x}_2) := \begin{cases} +\infty & , \text{ if } q_1 \neq q_2 \text{ or } q_1' \neq q_2'; \\ \|x_1 - x_2\|, & \text{if } q_1 = q_2 \text{ and } q_1' = q_2'. \end{cases} \tag{5.2.7}$$

Additionally, *Hausdorf distance* between any two hybrid sets $\underline{X}', \underline{X}'' \subset \underline{X}$ is defined as follows.

Definition 5.2.2. *Consider two hybrid sets $\underline{X}', \underline{X}'' \subset \underline{X}$. The* Hausdorf distance *between $\underline{X}'$ and $\underline{X}''$ is defined as*

$$\mathsf{d}_H(\underline{X}', \underline{X}'') := \inf\{\varepsilon \in \mathbb{R}_{\geq 0} | \underline{X}' \subseteq \underline{X}''_\varepsilon \wedge \underline{X}'' \subseteq \underline{X}'_\varepsilon\}. \tag{5.2.8}$$

Remark 5.2.3. *Note that the ε-contraction set in* (5.2.5) *can be empty when ε is too large. Hence, the ρ-contraction products as in* (5.2.6) *are only meaningful for those ρ with which the sets $\underline{X}_{-\rho}$, $(\underline{X}_0)_{-\rho}$, and $\underline{U} - \rho \mathbb{B}^m$ are not empty.*

5.2.2 Synthesis via Hybrid Controlled Invariant Sets

This subsection shows that Problem 5.1.7 can be solved by computing HCI sets (cf. Definition 5.2.5) for the product system as in Definition 5.2.1. To this end, the next result is required.

Theorem 5.2.4. *Consider a dtLCS $S = (X, X_0, U, W, f)$, a DSA $\mathcal{A} = (Q, q_0, \Pi, \delta, Acc)$, a labeling function $L : X \to \Pi$, the product system $S \otimes \mathcal{A}$ as in Definition 5.2.1, and a set $\underline{E} \subset \underline{X}$ such that $\underline{X} \backslash \underline{E}$ is the state set of the product system $S \otimes \mathcal{A}_{rd}(E')$, with $\mathcal{A}_{rd}(E') := (Q_{rd}, q_0, \Pi_{rd}, \delta_{rd}, Acc_{rd})$, and*

$$E' := \{q \in Q | \exists r \in \{1, \dots, \mathsf{r}\}, q \in E_r\}. \tag{5.2.9}$$

One has $S \models \mathcal{A}$ if for any infinite state sequence $\underline{\xi}_x = (\underline{x}(0), \underline{x}(1), \dots, \underline{x}(k), \dots)$ of $S \otimes \mathcal{A}$, $\underline{x}(k) \notin \underline{E}$, $\forall k \in \mathbb{N}$.

One can show Theorem 5.2.4 by considering the accepting condition of $\mathcal{A}$ as in (2.5.1). As a key insight, if one can find a controller that keeps all infinite state sequences of $S \otimes \mathcal{A}$ evolving within the set $\underline{X} \backslash \underline{E}$, then any state $q \in E'$ would be visited *at most once* considering the definition of the reduced DSA $\mathcal{A}_{rd}(E')$. One can build such a controller by leveraging HCI sets for $S \otimes \mathcal{A}$, as defined next.

Definition 5.2.5. *(*HCI Set*) A set $\underline{I} \subseteq \underline{X} \backslash \underline{E}$ is an HCI set for $S \otimes \mathcal{A}$, if $\forall \underline{x} \in \underline{I}$, $\exists u \in \underline{U}$ such that $\forall w \in \underline{W}$, one has $\underline{x}' := \underline{f}(\underline{x}, u, w) \in \underline{I}$, with $\underline{E}$ being the set as in Definition 5.2.4. Additionally, $\underline{I}^*$ denotes the* maximal HCI set *in the sense that for any other HCI set $\underline{I}' \subset \underline{X} \backslash \underline{E}$, one has $\underline{I}' \subset \underline{I}^*$.*

Note that the HCI set defined here is similar to the *strongly reachable set* in [23, Definition 2], but defined on the hybrid set $\underline{X}$ instead of $\mathbb{R}^n$. Based on the definition for the HCI set, the definition of an *HCI-based controller* is provided as follows.

Definition 5.2.6. *(*HCI-based controller*) Consider a dtLCS $S = (X, X_0, U, W, f)$, a DSA $\mathcal{A} = (Q, q_0, \Pi, \delta, Acc)$, a labeling function $L : X \to \Pi$, the product system $S \otimes \mathcal{A}$ as in Definition 5.2.1, and a non-empty HCI set $\underline{I}$ for $S \otimes \mathcal{A}$. An HCI-based controller $\mu : \underline{X} \to \underline{U}$ is constructed as follows: given $\underline{x}(k) = (q, q', x)$, input $u(k) = \mu(\underline{x}(k))$ should be chosen such that $\forall x' \in Ax(k) + Bu(k) + W$, one gets $(q', q'', x') \in \underline{I}$, with $(q', \sigma, q'') \in \delta$ and $\sigma = L(x')$.*

With Definition 5.2.6 in hand, the next result shows that once there exists a non-empty HCI set $\underline{I}$, the construction of an HCI-based controller is always feasible.

Proposition 5.2.7. *Consider a dtLCS S, a DSA $\mathcal{A}$ modeling the desired ω-regular property, and the product system $S \otimes \mathcal{A}$ as in Definition 5.2.1. For any non-empty HCI set $\underline{I}$ of $S \otimes \mathcal{A}$, there exists an HCI-based controller μ as in Definition 5.2.6.*

The proof of Proposition 5.2.7 is shown in Section 5.7.1. By virtue of Definition 5.2.6 and Proposition 5.2.7, one can reduce Problem 5.1.7 to the computation of (maximal) HCI sets for $S \otimes \mathcal{A}$. In Section 5.2.3, the computation of such sets will be discussed in details.

Running example (continued). For computing HCI set as in Definition 5.2.5,

$$\underline{E} := \bigcup_{\forall q' \in \{q_1, q_2\}} (q', q_1, \underline{X}(q', q_1)) \cup \bigcup_{\forall q' \in Q} (q', q_3, \underline{X}(q', q_3)), \tag{5.2.10}$$

is chosen, for which the corresponding reduced DSA $\mathcal{A}_{rd}(E')$ is depicted in Figure 5.2 (left). Note that the selection of the set $\underline{E}$ is not unique. One can also choose $\underline{E}$

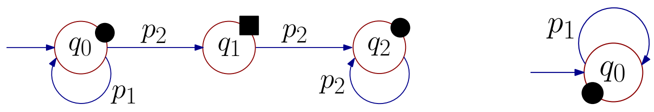


Figure 5.2: Reduced DSA $\mathcal{A}_{rd}(E')$ for different choices of $\underline{E}$.

such that $\underline{X} \backslash \underline{E} = \big(q_0, q_0, \underline{X}(q_0, q_0)\big)$, with the underlying reduced DSA as in Figure 5.2 (right). However, such a choice essentially prevents all the states in the set E' as in (5.2.9) from being reached, which is more conservative than the choice in (5.2.10).

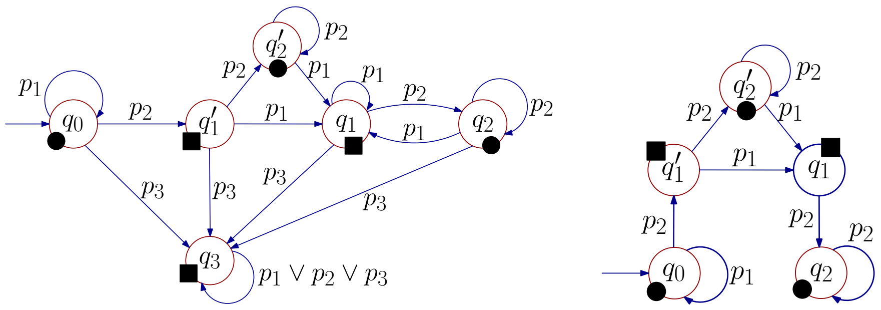


Figure 5.3: Left: DSA $\mathcal{A}'$ modeling ψ, with the same alphabet and labeling function as $\mathcal{A}$ in Figure 5.1, and accepting condition $\text{Acc} = \{\langle E_1, F_1\rangle, \langle E_2, F_2\rangle, \langle E_3, F_3\rangle\}$, with $E_1 = \{q_3\}$, $F_1 = \emptyset$, $E_2 = \{q_1, q_1'\}$, $F_2 = \{q_2, q_2'\}$, $E_3 = \emptyset$, and $F_3 = \{q_0\}$. Transition (q_2', p_3, q_3) is omitted to keep the figure less crowded. **Right**: The reduced DSA of $\mathcal{A}'$ with $\underline{E}$ selected as in (5.2.10).

Remark 5.2.8. *It is also worth mentioning that the results in Theorem 5.2.4 can readily be applied to synthesize controllers that allow some states in E' being visited* at most N' times, *where $N' \in \mathbb{N}_{\geq 1}$ is chosen by the users. For instance, to synthesize a controller that allows E_2 of $\mathcal{A}$ being visited at most twice (i.e. $N' = 2$), one can first reformulate $\mathcal{A}$ in Figure 5.1 to another DSA $\mathcal{A}'$ as in Figure 5.3 (Left). Then, one can apply Theorem 5.2.4 to $\mathcal{A}'$ by selecting $\underline{E}$ as in* (5.2.10)*, which corresponds to a reduced DSA as in Figure 5.3 (Right), and design an HCI-based controller accordingly (if existing).*

5.2.3 Computation of the Maximal HCI Sets

Inspired by the method proposed in [23] for computing maximal strongly reachable set, the following approach is proposed to compute the maximal HCI set.

Definition 5.2.9. *Consider a dtLCS S as in* (2.3.1)*, a DSA $\mathcal{A}$ modeling the desired ω-regular property, the product system $S \otimes \mathcal{A} = (\underline{X}, \underline{X}_0, \underline{U}, \underline{W}, \underline{f})$, and set $\underline{E} \subset \underline{X}$ selected as in Theorem 5.2.4. The maximal HCI set for $S \otimes \mathcal{A}$ can be computed with iteration* (5.2.11) *and stopping criterion* (5.2.12) *as:*

$$\underline{I}_0 = \underline{X} \backslash \underline{E},\ \underline{I}_{i+1} = \underline{I}_0 \cap \boldsymbol{P}(\underline{I}_i), \tag{5.2.11}$$

$$\underline{I}_i = \underline{I}_{i+1}, \tag{5.2.12}$$

where

$$\boldsymbol{P}(\underline{I}) = \{\underline{x} \in \underline{X} \mid \exists u \in \underline{U}, \forall w \in \underline{W},\ \textit{such that } \underline{f}(\underline{x}, u, w) \in \underline{I}\}, \tag{5.2.13}$$

denotes the set of states that reach $\underline{I}$ in one step. Once the iteration in (5.2.11) *is terminated by the stopping criterion in* (5.2.12)*, $\underline{I}_i$ is the maximal HCI set.*

To ensure the convergence of the iteration scheme in Definition 5.2.9, the following assumption is needed.

Assumption 5.2.10. *Consider a dtLCS S, a DSA $\mathcal{A}$ representing the desired ω-regular property, a labeling function $L : X \to \Pi$ as in Definition 5.1.1, and the corresponding product system $S \otimes \mathcal{A} = (\underline{X}, \underline{X}_0, \underline{U}, \underline{W}, \underline{f})$ as in* (5.2.1)*. It is assumed that:*

1. *Input set $\underline{U}$ and disturbance set $\underline{W}$ are of the form of polytopes in $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively;*
2. *The set $(\underline{X} \backslash \underline{E})(q, q')$, as defined in* (5.2.2)*, is compact and of the form of a P-collection in $\mathbb{R}^n$, $\forall q, q' \in Q$.*

With Definition 5.2.9 and Assumption 5.2.10, the following results show that $\underline{I}_i$ converges to maximal HCI set $\underline{I}^*$ as i goes to infinity.

Theorem 5.2.11. *Consider a dtLCS S as in Definition 2.3.1, and a DSA $\mathcal{A}$ modeling the desired ω-regular property such that Assumption 5.2.10 holds. Then, considering the iteration in* (5.2.11)*, one has $\underline{I}^* = \lim\limits_{i \to \infty} \underline{I}_i$, where the limit is in terms of the Hausdorff distance as in Definition 5.2.2.*

The proof of Theorem 5.2.11 is inspired by [23] and can be found in Section 5.7.1. Next, the implementation of (5.2.11) and (5.2.12) is discussed. Considering the dynamics as in (2.3.2), by the definition of $\underline{f}$, $\mathbf{P}(\underline{I})$ as in (5.2.13) can be rewritten as

$$\mathbf{P}(\underline{I}) = \{(q, q', x) \in \underline{X} \,|\, x \in \mathit{pre}(\underline{I}(q', q'')),\ \text{with } q, q', q'' \in Q \text{ s.t. } \exists \sigma \in \Pi, (q, \sigma, q') \in \delta\}, \tag{5.2.14}$$

with

$$\mathit{pre}(X') = \{x \in X | \exists u \in U,\ \forall w \in W, Ax + Bu + w \in X'\}, \tag{5.2.15}$$

Algorithm 5: Computing maximal HCI Set $\underline{I}^*$

```
Input: $\underline{X}\backslash\underline{E}$, $S \otimes \mathcal{A}$
Output: Maximal HCI set $\underline{I}^*$
1  $i = 0$, $\underline{I}_0 = \underline{X}\backslash\underline{E}$
2  while 1 do
3      $\underline{I}_{i+1} = \emptyset$, $Pr = \emptyset$;
4      foreach every $(q', q'')$ s.t. $\exists x$, $(q', q'', x) \in \underline{I}_i$ do
5          $Proj = pre(\underline{I}_i(q', q''))$;
6          foreach every $q \in Q$ s.t. $\exists \sigma \in \Pi$, $(q, \sigma, q') \in \delta$ do
7              $Pr = Pr \cup \{(q, q', x) | x \in Proj\}$;
8          end
9      end
10     foreach every $(q, q')$ s.t. $\exists x$, $(q, q', x) \in Pr$ do
11         $I_c = \underline{I}_0(q, q') \cap Pr(q, q')$;
12         $\underline{I}_{i+1} = \underline{I}_{i+1} \cup \{(q, q', x) | x \in I_c\}$;
13     end
14     if $\underline{I}_i = \underline{I}_{i+1}$ then
15         $\underline{I}^* = \underline{I}_i$;
16         Stop successfully;
17     else
18         if $\underline{I}_{i+1}$ is empty then
19             Stop unsuccessfully;
20         else
21             $i = i + 1$;
22         end
23     end
24 end
```

and $\underline{I}(q', q'') \neq \emptyset$ as defined in (5.2.2).

Roughly speaking, $pre(X')$ computes the *one-step-backward projection* of the set X' considering the linear dynamics as in (2.3.2). Based on (5.2.14), the implementation of Theorem 5.2.11 is provided in Algorithm 5. In each iteration, $\mathbf{P}(\underline{I}_i)$ is computed as in line 3-7, where line 7 and 5 correspond to (5.2.14) and (5.2.15), respectively; $\underline{I}_0 \cap \mathbf{P}(\underline{I}_i)$ is computed as in line 10-12. Note that one can readily employ existing toolboxes (e.g., multi-parametric toolbox `MPT` [78] and `BENSOLVE` [132]) to perform these polyhedral operations. The iteration proceeds until either: 1) $\underline{I}_i = \underline{I}_{i+1}$ (line 14-16); or 2) $\underline{I}_{i+1} = \emptyset$ (line 18-19), meaning a non-empty HCI set does not exist.

Remark 5.2.12. *If the set $\underline{X}\backslash\underline{E}$ is not compact, one can reselect the set $\underline{E}$ to ensure (if possible) the compactness of $\underline{X}\backslash\underline{E}$. Additionally, one can also (slightly) deflate the original set $\underline{X}\backslash\underline{E}$ such that one can start Algorithm 5 with a compact $\underline{X}\backslash\underline{E}$. Such deflation is shown using the running example.*

Running example (continued). With $\underline{E}$ selected as in (5.2.10), the set $\underline{X}\backslash\underline{E}$ is

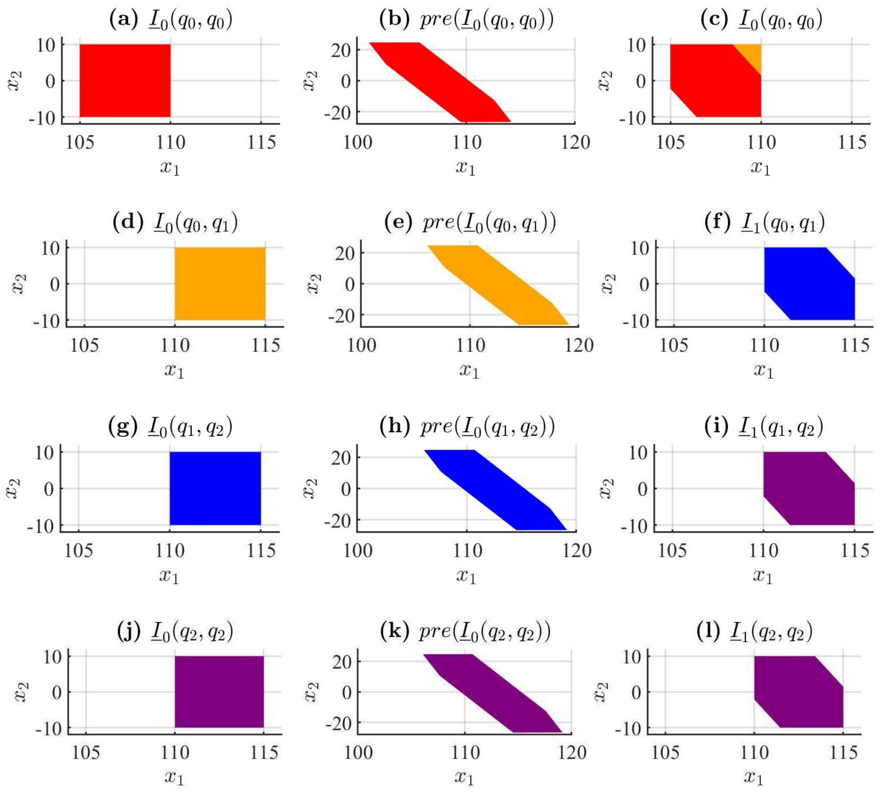


Figure 5.4: Computation of $\underline{I}_1$ based on $\underline{I}_0$ for the running example according to Algorithm 5.

not compact. Nevertheless, following the idea of Remark 5.2.12, one can ensure the compactness of $\underline{X}\backslash\underline{E}$ by slightly deflating it such that $\underline{X}\backslash\underline{E}(q_0, q_1) = \underline{X}\backslash\underline{E}(q_1, q_2) = [110+\epsilon, 115] \times [-10, 10]$, with $\epsilon \in \mathbb{R}_{>0}$ being any arbitrary positive real number. Here, $\epsilon = 0.01$ is selected for the computation as in Algorithm 5. To provide more intuition on how Algorithm 5 works, the computation of $\underline{I}_1$ based on $\underline{I}_0$ for the running example is demonstrated in Figure 5.4. Concretely, the iteration starts from $\underline{I}_0$ as depicted in Figure 5.4(a), (d), (g), and (j) (cf. line 1 in Algorithm 5). Then, by leveraging (5.2.15), the one-step-backward projection of $\underline{I}_0(q_0, q_0)$, $\underline{I}_0(q_0, q_1)$, $\underline{I}_0(q_1, q_2)$, and $\underline{I}_0(q_2, q_2)$ are computed, as shown in Figure 5.4(b), (e), (h), and (k), respectively (cf. line 5 in Algorithm 5). Based on these projections, $\mathbf{P}(\underline{I}_0)$ as in (5.2.14) are computed (cf. line 7 in Algorithm 5), in which

$$\begin{aligned}
&\mathbf{P}(\underline{I}_0)(q_0, q_0) = \big(q_0, q_0, pre(\underline{I}_0(q_0, q_0))\big);\\
&\mathbf{P}(\underline{I}_0)(q_0, q_1) = \big(q_0, q_0, pre(\underline{I}_0(q_0, q_1))\big);\\
&\mathbf{P}(\underline{I}_0)(q_1, q_2) = \big(q_0, q_1, pre(\underline{I}_0(q_1, q_2))\big);\\
&\mathbf{P}(\underline{I}_0)(q_2, q_2) = \big(q_1, q_2, pre(\underline{I}_0(q_2, q_2))\big) \cup \big(q_2, q_2, pre(\underline{I}_0(q_2, q_2))\big).
\end{aligned}$$

Finally, $\underline{I}_1$ is computed as in (5.2.11) based on $\mathbf{P}(\underline{I}_0)$ (cf. line 11 to 12 in Algorithm 5). Accordingly, one obtains

$$\underline{I}_1 = \big(q_0, q_0, \underline{I}_1(q_0, q_0)\big) \cup \big(q_0, q_1, \underline{I}_1(q_0, q_1)\big) \cup \big(q_1, q_2, \underline{I}_1(q_1, q_2)\big) \cup \big(q_2, q_2, \underline{I}_1(q_2, q_2)\big),$$

in which $\underline{I}_1(q_0, q_0) = \underline{I}_0(q_0, q_0) \cap \big(pre(\underline{I}_0(q_0, q_0)) \cup pre(\underline{I}_0(q_0, q_1))\big)$; $\underline{I}_1(q_0, q_1) = \underline{I}_0(q_0, q_1) \cap pre(\underline{I}_0(q_1, q_2))$; $\underline{I}_1(q_1, q_2) = \underline{I}_0(q_1, q_2) \cap pre(\underline{I}_0(q_2, q_2))$; and $\underline{I}_1(q_2, q_2) = \underline{I}_0(q_2, q_2) \cap pre(\underline{I}_0(q_2, q_2))$, as illustrated in Figure 5.4 (c), (f), (i), and (l), respectively.

It is worth mentioning that when invariance properties are of interest, the iteration in (5.2.11) terminates within a finite number of steps if there are additional assumptions on the system dynamics (see e.g. [191, Proposition 4]), or if X, U, and W have special shapes (see e.g. [73, Theorem 3.1], [191, Theorem 5], [30, Proposition 5.9],[158, Theorem 1 and Corollary 1]). However, there is no guarantee that (5.2.11) can be terminated within a finite number of iterations, in general. To cope with this issue, two alternative iterative schemes are proposed to compute approximations of $\underline{I}^*$ (if existing) within a finite number of iterations, which are introduced in Section 5.3.

5.3 Approximation of Maximal HCI Sets

In this section, two methods are proposed for computing approximations of $\underline{I}^*$ for $S \otimes \mathcal{A}$ within a finite number of iterations. For both methods, the following assumption for the dtLCS is required.

Assumption 5.3.1. *Consider a dtLCS S as in Definition 2.3.1. It is assumed that (A,B) in* (2.3.2) *is controllable.*

5.3.1 (ε_x,ε_u)-Contraction-based Approximation

In this subsection, I show how to compute an *(ε_x,ε_u)-contraction-based approximation* of $\underline{I}^*$ for $S \otimes \mathcal{A}$. This approximation is computed based on *a sequence of ($\varepsilon_x, \varepsilon_u$)-constraint i-step null-controllable sets*, as defined below.

Definition 5.3.2. *Consider a dtLCS S as in Definition 2.3.1 in which $W = \{\mathbf{0}_n\}$, and some $\varepsilon_x, \varepsilon_u \in \mathbb{R}_{>0}$. A sequence of* ($\varepsilon_x, \varepsilon_u$)-constraint i-step null-controllable sets, *denoted by $(\mathcal{N}_i(\varepsilon_x, \varepsilon_u))_{i\in\mathbb{N}}$, is recursively defined as*

$$\begin{aligned}\mathcal{N}_0(\varepsilon_x, \varepsilon_u) =&\{\mathbf{0}_n\},\\ \mathcal{N}_{i+1}(\varepsilon_x, \varepsilon_u) =&\{x \in \mathbb{R}^n | \exists u \in \varepsilon_u\mathbb{B}^m, Ax + Bu \in \mathcal{N}_i(\varepsilon_x, \varepsilon_u)\} \cap \varepsilon_x\mathbb{B}^n.\end{aligned} \tag{5.3.1}$$

Moreover, the following lemma for $(\mathcal{N}_i(\varepsilon_x, \varepsilon_u))_{i\in\mathbb{N}}$ is needed for computing the (ε_x,ε_u)-contraction-based approximation.

Lemma 5.3.3. *Consider a dtLCS S as in* (2.3.2) *in which $W = \{\mathbf{0}_n\}$ and (A,B) is controllable, and a sequence $(\mathcal{N}_i(\varepsilon_x, \varepsilon_u))_{i\in\mathbb{N}}$ as defined in* (5.3.1)*. Then, $\exists c_x, c_u \in \mathbb{R}_{>0}$ and $n' \in \mathbb{N}$ with $n' \leq n$ such that $\forall \gamma \in \mathbb{R}_{>0}$, one has*

$$\gamma\mathbb{B}^n \subseteq \mathcal{N}_{n'}(\varepsilon_x, \varepsilon_u), \tag{5.3.2}$$

with $\varepsilon_x = c_x\gamma$ and $\varepsilon_u = c_u\gamma$.

The proof of Lemma 5.3.3 is inspired by [165, Lemma 2] and given in Section 5.7.2. Note that c_x, c_u, and n' in Lemma 5.3.3 can be obtained by leveraging the next result.

Corollary 5.3.4. *Consider the vertices $z_i \in \mathbb{R}^n$ of $\mathbb{B}^n$, with $i \in [1, 2^n]$. One can select any $c_x, c_u \in \mathbb{R}^n$, and $n' \in \mathbb{N}$ for Lemma 5.3.3 such that* (5.3.2) *holds, if the following constraints are respected for all z_i:*

$$A^{n'}z_i + \sum_{j=0}^{n'-1} A^{n'-j-1}Bu_j = \mathbf{0}_n; \tag{5.3.3}$$

$$|u_j| \leq c_u, \forall j \in [0, n'-1]; \tag{5.3.4}$$

$$|A^d z_i + \sum_{j=0}^{d-1} A^{d-j-1}Bu_j| \leq c_x, \forall d \in [1, n'-1], \tag{5.3.5}$$

with $u_j \in \mathbb{R}^m$, $j \in [0, n'-1]$.

The proof of Section 5.3.4 is provided in Appendix 5.7.2. Next, the computation of $(\varepsilon_x,\varepsilon_u)$-contraction-based approximation is illustrated in Definition 5.3.5.

Definition 5.3.5. ($(\varepsilon_x,\varepsilon_u)$-contraction-based approximation) *Consider a dtLCS S as in Definition 2.3.1 such that Assumption 5.3.1 holds, a DSA $\mathcal{A}$ modeling the desired property, and the product system $S \otimes \mathcal{A} = (\underline{X}, \underline{X}_0, \underline{U}, \underline{W}, \underline{f})$. Given $c_x, c_u \in \mathbb{R}_{>0}$ as in Corollary 5.3.4, and any $\gamma \in \mathbb{R}$, the $(\varepsilon_x,\varepsilon_u)$-contraction-based approximation can be computed with iteration as in* (5.3.6) *and stopping criterion as in* (5.3.7)*:*

$$\underline{I}_0 = (\underline{X}\backslash\underline{E})_{-\varepsilon_x},\ \underline{I}_{i+1} = \underline{I}_0 \cap \boldsymbol{P}_{(\varepsilon_x,\varepsilon_u)}(\underline{I}_i), \tag{5.3.6}$$

$$\underline{I}_i \subseteq (\underline{I}_{i+n'})_{\gamma}, \tag{5.3.7}$$

where ε_x, ε_u, and n' are as in Lemma 5.3.3 s.t. (5.3.2) *holds, $\boldsymbol{P}_{(\varepsilon_x,\varepsilon_u)}(\underline{I})$ is defined similarly to $\boldsymbol{P}(\underline{I})$ as in* (5.2.14)*, with*

$$pre(X') = \{x \in X | \exists u \in U - \varepsilon_u\mathbb{B}^m, \forall w \in W, Ax + Bu + w \in X'\}. \tag{5.3.8}$$

By leveraging the iteration and stopping criterion as in Definition 5.3.5, one can construct the $(\varepsilon_x,\varepsilon_u)$-contraction-based approximation using the following result.

Theorem 5.3.6. *For any $\gamma \in \mathbb{R}_{>0}$ and the corresponding $\varepsilon_x, \varepsilon_u \in \mathbb{R}_{>0}$, $n' \in \mathbb{N}$ as in Lemma 5.3.3, there exists $i \in \mathbb{N}$ with which* (5.3.7) *holds. Moreover, consider $(\underline{I}_i)_{i\in\mathbb{N}}$ that is obtained through the iteration as in* (5.3.6)*, and the sequence $(\mathcal{N}_i(\varepsilon_x, \varepsilon_u))_{i\in\mathbb{N}}$ as in Definition 5.3.2. The set*

$$\underline{I}(\varepsilon_x, \varepsilon_u) = \bigcup_{i'\in[1,n']} (\underline{I}_{i_*+i'} \oplus \mathcal{N}_{i'}(\varepsilon_x, \varepsilon_u)), \tag{5.3.9}$$

is an HCI set for the product system $S \otimes \mathcal{A}$, with $i^* \in \mathbb{N}$ being the smallest index i for the given γ such that (5.3.7) *holds.*

The proof of Theorem 5.3.6 can be found in Section 5.7.2. Note that the existence of $i \in \mathbb{N}$ such that (5.3.7) holds indicates that the iteration in (5.3.6) can be terminated within finite number of iterations. Since $\underline{I}(\varepsilon_x, \varepsilon_u)$ in (5.3.9) is an HCI set for $S \otimes \mathcal{A}$, it is, by definition, an under-approximation of the maximal HCI set $\underline{I}^*$ according to Definition 5.2.5. The next result shows how close this approximation is. In brief, it is shown that given a $\rho \in \mathbb{R}_{>0}$ and a product system $(S \otimes \mathcal{A})_{-\rho}$ as defined in (5.2.6), one can construct an $(\varepsilon_x,\varepsilon_u)$-contraction-based approximation that contains the maximal HCI set for $(S \otimes \mathcal{A})_{-\rho}$ by selecting ε_x and ε_u properly.

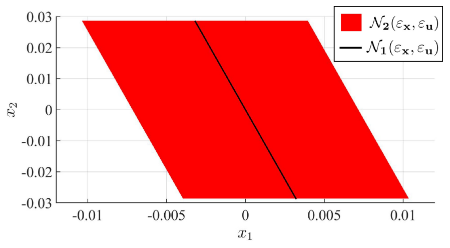


Figure 5.5: Sequences of $(\varepsilon_x, \varepsilon_u)$-constraint 2-step null-controllable sets.

Theorem 5.3.7. *Consider a dtLCS S as in Definition 2.3.1 such that Assumption 5.3.1 holds, a DSA $\mathcal{A}$ modeling the desired property, and the product system $S \otimes \mathcal{A} = (\underline{X}, \underline{X}_0, \underline{U}, \underline{W}, \underline{f})$. For any $\rho \in \mathbb{R}_{>0}$, there exists $\gamma \in \mathbb{R}_{>0}$, such that*

$$\underline{I}^*_\rho \subseteq \underline{I}(\varepsilon_x, \varepsilon_u), \tag{5.3.10}$$

where $\underline{I}^*_\rho$ is the maximal HCI set for $(S \otimes \mathcal{A})_{-\rho}$ as defined in (5.2.6)*, $\underline{I}(\varepsilon_x, \varepsilon_u)$ is as in* (5.3.9) *with ε_x and ε_u being computed as in Lemma 5.3.3 based on γ.*

The proof of Theorem 5.3.7 is provided in Section 5.7.2.

Remark 5.3.8. *Note that in* (5.3.6) *and* (5.3.8)*, one deploys a deflate version of U and $\underline{X}\backslash\underline{E}$ to compute the one-step-backward projection of the hybrid set $\underline{I}_i$ in* (5.3.6)*.*

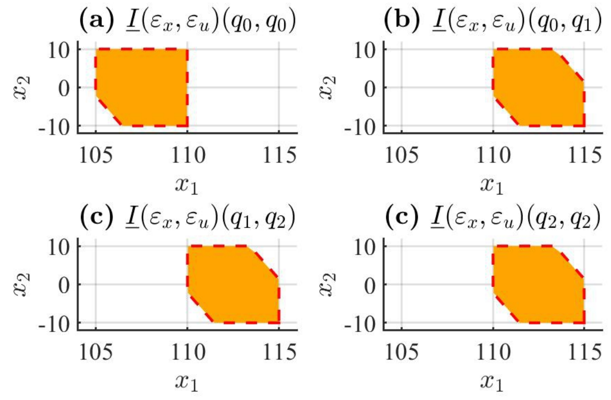


Figure 5.6: Result for $(\varepsilon_x,\varepsilon_u)$-contraction-based approximation (orange region), with $\varepsilon_x = 2.8636$ and $\varepsilon_u = 0.67251$, and the actual maximal HCI set $\underline{I}^*$ (red dashed lines).

Therefore, one should at least consider those ε_x and ε_u such that $U - \varepsilon_u \mathbb{B}^m \neq \emptyset$ and $\underline{I}_0 \cap \boldsymbol{P}_{(\varepsilon_x,\varepsilon_u)}(\underline{I}_0) \neq \emptyset$, so that the HCI set $\underline{I}(\varepsilon_x, \varepsilon_u)$ in (5.3.9) *would not trivially be empty. Note that both ε_x and ε_u are hyperparameters for the iterative scheme in* (5.3.6) *and* (5.3.7)*. In general, there is no guarantee that there exists $\varepsilon_x, \varepsilon_u \in \mathbb{R}_{>0}$ such that one can obtain* non-empty *$\underline{I}(\varepsilon_x, \varepsilon_u)$ as in* (5.3.9) *for any arbitrary system as in* (2.3.2)*. Therefore, in practice, one can first select $\varepsilon_x = \varepsilon_u = 0$ to compute the maximal HCI set $\underline{I}^*$. If one does not obtain $\underline{I}^*$ within a desirable number of steps (which may be the case; see discussion at the end of Section 5.2), one can select new $\varepsilon_u \in (0, \varepsilon'_u]$ and $\varepsilon_x \in (0, \varepsilon'_x]$ in a bisection manner, in which $\varepsilon'_x, \varepsilon'_u \in \mathbb{R}_{>0}$ are selected such that $U - \varepsilon'_u \mathbb{B}^m = \emptyset$ and $\underline{I}_0 \cap \boldsymbol{P}_{(\varepsilon'_x,\varepsilon'_u)}(\underline{I}_0) = \emptyset$.*

Running example (continued). To compute the $(\varepsilon_x,\varepsilon_u)$-contrac-tion-based approximation, $n{=}2$ and $\gamma{=}0.01$ are chosen. and get $\varepsilon_x{=}2.86$ and $\varepsilon_u{=}0.67$ considering Lemma 5.3.3 and Corollary 5.3.4. The corresponding sequences of $(\varepsilon_x, \varepsilon_u)$-constraint n-step null-controllable sets are depicted in Figure 5.5. Then, the approximation is computed by applying Definition 5.3.5 and Theorem 5.3.6. The computation ends within 1.36 seconds with 4 iterations. The approximation contains 49 hyperplanes, and it is depicted in Figure 5.6. For comparison purposes, the actual maximal HCI set $\underline{I}^*$ is also shown.

5.3.2 ε-Expansion-based Approximation

Then main topic of this subsection is the computation of an ε-expansion-based approximation of the maximal HCI set for $S \otimes \mathcal{A}$. Such approximations can be computed as in Definition 5.3.9.

Definition 5.3.9. (ε-expansion-based approximation) *Consider a dtLCS S as in* (2.3.2) *such that Assumption 5.3.1 holds, a DSA $\mathcal{A}$ modeling the desired property, and the product system $S \otimes \mathcal{A} = (\underline{X}, \underline{X}_0, \underline{U}, \underline{W}, \underline{f})$. Given $\varepsilon \in \mathbb{R}_{>0}$, one can compute the ε-expansion-based approximation with iteration as in* (5.3.11) *and stopping criterion as* *in* (5.3.12)*:*

$$\underline{I}_0 = \underline{X} \backslash \underline{E},\ \underline{I}_{i+1} = \underline{I}_0 \cap \boldsymbol{P}_\varepsilon(\underline{I}_i), \tag{5.3.11}$$

$$\underline{I}_i \subseteq (\underline{I}_{i+1})_\varepsilon, \tag{5.3.12}$$

in which $\boldsymbol{P}_\varepsilon(\underline{I})$ *is defined similarly to* $\boldsymbol{P}(\underline{I})$ *as in* (5.2.14)*, with*

$$pre(X') = \{x \in X | \exists u \ \in U, \forall w \in W', Ax + Bu + w \in X'\}, \tag{5.3.13}$$

and $W' := W + \varepsilon\mathbb{B}^n$*.*

Unlike (5.2.15), $pre(X')$ as in (5.3.13) is defined based on an *ε-expansion* of the set W, i.e. $W + \varepsilon\mathbb{B}^n$. With Definition 5.3.9, the next theorem shows the termination of (5.3.11) and the construction of the ε-expansion-based approximation.

Theorem 5.3.10. *Consider any* $\varepsilon \in \mathbb{R}_{>0}$*. There exists* $i \in \mathbb{N}$ *with which* (5.3.12) *holds. Additionally, the set*

$$\underline{I}(\varepsilon) := \underline{I}_{i^*+1}, \tag{5.3.14}$$

is an HCI set for the product system $S \otimes \mathcal{A}$*, with* $i^* \in \mathbb{N}$ *being the smallest index* i *for the given* ε *such that* (5.3.12) *holds.*

The proof of Theorem 5.3.10 can be found in Section 5.7.2. Note that $\underline{I}(\varepsilon)$ in (5.3.14) is an HCI set for $S \otimes \mathcal{A}$, it is therefore also an under-approximation of the maximal HCI set $\underline{I}^*$ according to Definition 5.2.5. Then, similar to Theorem 5.3.7, the next result illustrates how close this approximation is.

Theorem 5.3.11. *Consider a dtLCS* S *as in* (2.3.2) *such that Assumption 5.3.1 holds, a DSA* $\mathcal{A}$ *modeling the desired property, and the product system* $S \otimes \mathcal{A} = (\underline{X}, \underline{X}_0, \underline{U}, \underline{W}, \underline{f})$*. For any* $\rho \in \mathbb{R}_{>0}$*, there exists* $\varepsilon \in \mathbb{R}_{>0}$*, such that*

$$\underline{I}^*_\rho \subseteq \underline{I}(\varepsilon), \tag{5.3.15}$$

where $\underline{I}^*_\rho$ *is the maximal HCI set for* $(S \otimes \mathcal{A})_{-\rho}$ *as defined in* (5.2.6)*, and* $\underline{I}(\varepsilon)$ *is as in* (5.3.14)*.*

The proof of Theorem 5.3.11 can be found in Section 5.7.1.

Remark 5.3.12. *Note that an inflated version of the disturbance set* W*, i.e.,* $W + \varepsilon\mathbb{B}^n$*, is considered in* (5.3.13) *for computing the one-step-backward projection of the hybrid set* $\underline{I}_i$ *in* (5.3.11)*. To avoid obtaining an empty HCI set* $\underline{I}(\varepsilon)$ *in* (5.3.14)*, one should at least select those* $\varepsilon \in \mathbb{R}_{>0}$ *such that* $\underline{I}_0 \cap \boldsymbol{P}_\varepsilon(\underline{I}_0) \neq \emptyset$*. Note that* ε *is a hyperparameter for computing the HCI set in* (5.3.14)*. In practice, one can tune* ε *in a similar way as* $\varepsilon_x, \varepsilon_u$ *for computing the (*ε_x*,*ε_u*)-contraction-based approximation (c.f. Remark 5.3.8).*

Running example (continued). (Running example) Here, $\varepsilon = 0.1$ is selected to compute the ε-expansion-based approximation by applying Definition 5.3.9 and Theorem 5.3.10. The computation terminates within 1.26 seconds with 3 iterations. The approximation contains 36 hyperplanes and it is illustrated in Figure 5.7. Additionally, the actual maximal HCI set $\underline{I}^*$ is also depicted for comparison purposes.

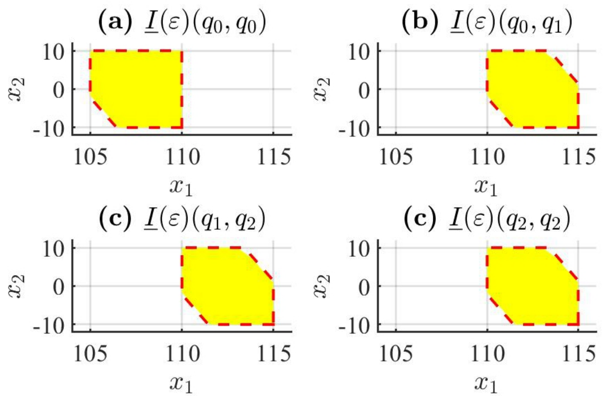


Figure 5.7: Result for ε-expansion-based approximation (yellow region), with $\varepsilon = 0.1$, and the actual maximal HCI set $\underline{I}^*$ (red dashed lines).

5.4 Complexity

In this section, the space and time complexities of the proposed approaches in this chapter are discussed. Note that the space and time complexities for the cases in which W has a non-empty interior is still open. As a key insight, considering a P-collection, denoted by $X' := \cup_{a=1}^{\mathsf{N}_c} X'_a$, one can verify that

$$\mathsf{larg}\Big(pre(X')\Big) = \mathsf{larg}\Big((X' - W) + (-BU)\Big), \tag{5.4.1}$$

holds by employing the results in [102, Section 3.3.3, pp. 44], in which $\mathsf{larg}(\cdot)$ is defined in (5.1.4), and BU denotes the linear mapping of the input set U regarding matrix B [102, Section 3.4.2]. However, if $\exists j, k \in [1, \mathsf{N}_c]$ such that $X'_j \cap X'_k \neq \emptyset$, i.e. X'_a are not pairwise disjoint, it is still an open problem as to what is the upper bound of the number of polytopes within $X' - W$, and what is the maximal number of hyperplanes defining each polytope within $X' - W$. Thus, in the remaining discussion, I only focus on the case in which $W = \{\mathbf{0}_n\}$. To derive the space and time complexities for this case, the following definitions are required.

Definition 5.4.1. *Consider a dtLCS $S = (X, X_0, U, W, f)$ with $W = \{\mathbf{0}_n\}$, and $p \in \mathbb{N}$. Function $\tilde{g}_S : \mathbb{N} \to \mathbb{N}$ is defined as*

$$\tilde{g}_S(p) := \max_{X' \in \mathcal{P}(n),\ with\ \mathsf{numh}(X') = p} \mathsf{numh}(pre(X')), \tag{5.4.2}$$

with $\mathsf{numh}(\cdot)$ *defined as in* (5.1.3), *pre*$(\cdot)$ *defined as in* (5.2.15), *n being the dimension of X, and $X' \subseteq X$.*

Definition 5.4.2. *Consider a dtLCS $S=(X, X_0, U, W, f)$ and a DSA $\mathcal{A} = (Q, q_0, \Pi, \delta, Acc)$ modeling the desired ω-regular property. The following set is defined:*

$$Q_{rd} := \{q \in Q | \exists q' \in Q, \text{ such that } \underline{X} \backslash \underline{E}(q', q) \neq \emptyset \text{ or } \underline{X} \backslash \underline{E}(q, q') \neq \emptyset\}, \tag{5.4.3}$$

with the set $\underline{E}$ being defined in Theorem 5.2.4.

Intuitively, $\tilde{g}_S(p)$ denotes the maximal number of hyperplanes defining $pre(X')$, with X' being any arbitrary polytope defined by p hyperplanes. The set Q_{rd} is the finite state set of the reduced DSA corresponding to the set $\underline{E}$. With these definitions, the next result is proposed to pave the way for deriving the worst-case space and time complexities.

Theorem 5.4.3. *Consider a dtLCS $S = (X, X_0, U, W, f)$ with $W = \{\mathbf{0}_n\}$, a DSA $\mathcal{A} = (Q, q_0, \Pi, \delta, Acc)$ modeling the desired ω-regular property, and the sequence of $\underline{I}_i$ with $i \in \mathbb{N}$ as defined in* (5.2.11) *and* (5.2.12). *One has*

$$\mathsf{num}(\underline{I}_i(q, q')) \leq \alpha^i \mathsf{M}^{i+1}, \tag{5.4.4}$$

$$\mathsf{larg}(\underline{I}_i(q, q')) \leq g^i(p'), \tag{5.4.5}$$

for any $q, q' \in Q_{rd}$, with Q_{rd} being defined as in (5.4.3), *where*

$$\alpha := \max_{q \in Q_{rd}} |\mathsf{out}(q)| \tag{5.4.6}$$

$$\mathsf{M} := \max_{q, q' \in Q_{rd}} \mathsf{num}(\underline{I}_0(q, q')), \tag{5.4.7}$$

$$p' := \max_{\mathcal{P}, \mathcal{P} \subset \underline{I}_0(q, q') \text{ with } q, q' \in Q_{rd}} \mathsf{numh}(\mathcal{P}), \tag{5.4.8}$$

in which $|\mathsf{out}(q)|$ is the cardinality of the set

$$\mathsf{out}(q) := \{q' \in Q \mid \exists \sigma \in \Pi, (q, \sigma, q') \in \delta\};$$

$\underline{I}_0$ is as in (5.2.11); $\mathsf{num}(\cdot)$, $\mathsf{larg}(\cdot)$, *and* $\mathsf{numh}(\cdot)$ *are defined in* (5.1.5), (5.1.4) *and* (5.1.3), *respectively; $\mathcal{P}$ is any arbitrary polytope within $\underline{I}_0(q, q')$; and $g^i : \mathbb{N} \to \mathbb{N}$, with $i \in \mathbb{N}$, is recursively defined as*

$$\begin{aligned} &g^i(p') = p', \text{ when } i = 0; \\ &g^i(p') = p' + \tilde{g}_S(g^{i-1}(p')), \text{ when } i \geq 1, \end{aligned} \tag{5.4.9}$$

where $\tilde{g}_S(\cdot)$ is defined in (5.4.2).

Remark 5.4.4. *As a key insight, Theorem 5.4.3 provides upper bounds on: 1) the number of polytopes within $\underline{I}_i(q, q')$; 2) the number of hyperplanes defining each polytope within $\underline{I}_i(q, q')$. These upper bounds are conservative since they are derived without*

Functions	Tasks
$c_1(a_1, b_1)$	Compute $pre(X')$, with X' beinga P-collection in $\mathbb{R}^n$ for which $\mathsf{num}(X') = a_1$, $\mathsf{larg}(X') = b_1$
$c_2(a_2, b_2)$	Concatenate matrices $\mathsf{P}_1 \in \mathbb{R}^{a_2\times(n+1)}$ with $\mathsf{P}_2 \in \mathbb{R}^{b_2\times(n+1)}$
$c_3(a_3, b_3, a'_3, b'_3)$	Check whether $X'_{i-1} \subseteq X'_i$ holds, with $X'_i, X'_{i-1} \subset \mathbb{R}^n$ being P-collections, where $\mathsf{num}(X'_i) = a_3$, $\mathsf{larg}(X'_i) = b_3$, $\mathsf{num}(X'_{i-1}) = a'_3$, $\mathsf{larg}(X'_{i-1}) = b'_3$

Table 5.1: Definition of c_1, c_2, and c_3.

considering the possibility of eliminating redundant hyperplanes and polytopes in practice. Concretely, intersections among polytopes in each iteration may contain some redundant hyperplanes, which can be eliminated by computing the minimal representations of these intersections [17]. Additionally, one can also reduce $\mathsf{num}(\underline{I}_i(q, q'))$ *by computing unions among some of the polytopes within* $\underline{I}_i(q, q')$*, in case these unions are in the form of polytopes.*

The proof of Theorem 5.4.3 is provided in Section 5.7.3. Based on Theorem 5.4.3, the worst-case space and time complexities of Algorithm 5 is proposed in the following corollary.

Corollary 5.4.5. *Consider a dtLCS* $S = (X, X_0, U, W, f)$ *with* $W = \{\mathbf{0}_n\}$*, a DSA* $\mathcal{A} = (Q, q_0, \Pi, \delta, Acc)$ *modeling the desired* ω*-regular property, and* $i \in \mathbb{N}_{>0}$ *the number of iterations. The worst-case space and time complexities of Algorithm 5 are*

$$\mathcal{O}\Big(|\delta|\alpha^i \mathsf{M}^{i+1} g^i(p')n\Big), \tag{5.4.10}$$

$$\begin{aligned}&\mathcal{O}\Big(|\delta|c_1\big(\alpha^{i-1}\mathsf{M}^i, g^{i-1}(p')\big) + |\delta|\alpha^{i-1}\mathsf{M}^{i+1}c_2\big(p', \tilde{g}_S(g^{i-1}(p'))\big)\\&\quad + |\delta|c_3\big(\alpha^i\mathsf{M}^{i+1}, g^i(p'), \alpha^{i-1}\mathsf{M}^i, g^{i-1}(p')\big)\Big),\end{aligned} \tag{5.4.11}$$

respectively, in which $|\delta|$ *is the number of transitions among* $q, q' \in Q_{rd}$*, with* Q_{rd} *as defined in* (5.4.3)*;* α*,* M*,* p' *and* $g^i(p')$ *are defined in* (5.4.6)-(5.4.9)*, respectively;* $\tilde{g}_S(\cdot)$ *is defined in* (5.4.2)*;* c_1*,* c_2*, and* c_3 *represent the computation costs for accomplishing different tasks as defined in Table 5.1.*

Remark 5.4.6. *For each* $i \in \mathbb{N}_{>0}$*, the tasks for the iteration in* (5.2.11) *and* (5.2.12) *include: 1) computing the one-step-backward projection* $\boldsymbol{P}(\underline{I}_{i-1})$ *of* $\underline{I}_{i-1}$*; 2) computing the intersection* $\underline{I}_0 \cap \boldsymbol{P}(\underline{I}_{i-1})$*; 3) checking whether* $\underline{I}_{i-1} \subseteq \underline{I}_i$ *holds*[2]*. Their computation costs correspond to the first, second, and third term in* (5.4.11)*, respectively. Here, the closed-form expressions of* c_1*,* c_2*, and* c_3 *depend on the concrete methods that are deployed for their associated tasks. For instance, given a polytope* $X' \subset \mathbb{R}^n$*, computing*

[2]One can verify that $\underline{I}_i \subseteq \underline{I}_{i-1}$ always holds based on the way of computing $\underline{I}_i$.

$pre(X')$ includes the computation of inverse image of a polytope and polyhedral projection [159]. For linear systems as in (2.3.2)*, the inverse image of a polytope can be obtained via simple matrix multiplications as in [165, Section 4], while different approaches can be used to compute the projection of a polytope [90, 96, 206]. Similarly, various results can be applied to check whether $\underline{I}_{i-1}\subseteq\underline{I}_i$ holds, e.g. [19, 17].*

Remark 5.4.7. *With slight modifications, Definition 5.4.1, Theorem 5.4.3, and Corollary 5.4.5 can also be leveraged to analyze the space and time complexities of the computation of (ε_x,ε_u)-contraction-based approximation. Concretely, $pre(\cdot)$ in* (5.4.2) *should be defined as in* (5.3.8) *(instead of* (5.2.15)*), and $\underline{I}_0$ in* (5.4.7) *and* (5.4.8) *should be defined as in* (5.3.6) *(instead of* (5.2.11)*).*

The proof of Corollary 5.4.5 is provided in Section 5.7.3. Then, the following results can be leveraged to obtain the closed-form expressions of $\tilde{g}_S(p)$ in (5.4.2) and $g^i(p')$ in (5.4.10) and (5.4.11).

Proposition 5.4.8. *Consider a dtLCS $S = (X, X_0, U, W, f)$ as in Definition 2.3.1, where n is the dimension of X, $W = \{\mathbf{0}_n\}$, and $p_{\mathsf{U}} := \mathsf{numh}(BU)$. Given p' as in* (5.4.8)*, and $i \in \mathbb{N}$ the number of iterations, one has*

$$\tilde{g}_S(p') \leq \begin{cases} 2, & \text{when } n = 1; \\ p_{\mathsf{U}} + p', & \text{when } n = 2; \\ (4p_{\mathsf{U}} - 9)p' + 26 - 9p_{\mathsf{U}}, & \text{when } n = 3. \end{cases} \tag{5.4.12}$$

Accordingly, one gets

$$g^i(p') \leq \begin{cases} 2(i+1), & \text{when } n = 1; \\ p' + i(p' + p_{\mathsf{U}}), & \text{when } n = 2; \\ \dfrac{1-\tilde{a}^{i+1}}{1-\tilde{a}}p' + \dfrac{1-\tilde{a}^i}{1-\tilde{a}}\tilde{b}, & \text{when } n = 3. \end{cases} \tag{5.4.13}$$

with $\tilde{a} = 4p_{\mathsf{U}} - 9$, and $\tilde{b} = 26 - 9p_{\mathsf{U}}$.

Note that one can verify $p_{\mathsf{U}} \leq \mathsf{numh}(U)+2(n-rank(B))$ according to [102, Corollary 3.5]. Considering (5.4.1), solving the closed-form expressions of $\tilde{g}_S(p')$ is equivalent to answering the following question: *given polytopes $\mathcal{P}_1$ and $\mathcal{P}_2$ defined by p' and p_{U} hyperplanes, respectively, what is the upper bound of the number of hyperplanes defining $\mathcal{P}_1 + \mathcal{P}_2$?* Trivially, 2 is the upper bound for the case $n = 1$. Additionally, one has $p_{\mathsf{U}} + p'$ being the upper bound for the case $n = 2$ according to [188, Theorem 13.5], and $(4p_{\mathsf{U}} - 9)p' + 26 - 9p_{\mathsf{U}}$ being the upper bound for the case $n = 3$ according to [198, Theorem 5.2.1]. Then, (5.4.13) can accordingly be derived. As for the cases $n \geq 4$, to the best of my knowledge, there is no result providing the upper bounds of the number of hyperplanes defining $\mathcal{P}_1 + \mathcal{P}_2$ based on p' and p_{U}. However, once the results for these upper bounds are available, the space complexities for the cases $n \geq 4$ can readily be derived based on Corollary 5.4.5.

Finally, I also want to point out the difficulties in having a fair comparison between those discretization-based approaches and ours in terms of worst-case space complexity. It is well-known that the space complexities of discretization-based approaches grow exponentially with respect to the dimension of the state (and input) sets (see [127, Section 5-A] for detailed discussion) since they require the discretization of the original state and input sets in order to construct the finite state and input sets. On the one hand, the space complexity of the proposed approaches does not have exponential growth regarding the dimensions since such discretization is not required. On the other hand, the complexity of proposed approaches grows exponentially with respect to the number of iterations in the worst case. It is worth noting that, however, such exponential growth is not observed in the case studies (see Figure 5.12). As a key insight, at each iteration step $i \in \mathbb{N}$, for all $q, q' \in Q_{rd}$, one can reduce $\mathsf{num}(\underline{I}_i(q, q'))$ and $\mathsf{larg}(\underline{I}_i(q, q'))$ in (5.4.4) and (5.4.5) by computing the minimal representations [17] and the union of (some of) the polytopes in $\underline{I}_i(q, q')$.

5.5 Case Studies

To show the effectiveness of proposed results, the running example is first simulated with the HCI-based controllers, which have already been computed in Section 5.3. Then, the results in this chapter are applied to a cruise control example and a quadrotor helicopter. Finally, the proposed approaches are compared with some existing tools in terms of computational time. The synthesis and simulation are performed on a computer equipped with Quad-Core Intel Core i7 (2.7 GHz) and 16 GB of RAM running macOS Big Sur (Version 11.5.2), using `MATLAB2019b` along with multi-parametric toolbox `MPT` [78] and optimization software `MOSEK` (version 9.3.6) [143]. It is also worth noting that controllers in all cases can be applied over an infinite time horizon. The numbers of time steps for the simulation are selected only for demonstration purposes.

5.5.1 Running Example

Here, 10 different initial states are randomly selected from $\underline{I}^*(q_0, q_0)$, $\underline{I}(\varepsilon_x, \varepsilon_u)(q_0, q_0)$, and $\underline{I}(\varepsilon)(q_0, q_0)$ (cf. Figure 5.6 and Figure 5.7), respectively, and simulate the running example for 30 time steps. In the simulation, the disturbances affecting the system are randomly generated at each time instant following a uniform distribution within the disturbance set. The simulation results for the maximal HCI set, the $(\varepsilon_x, \varepsilon_u)$-contraction-based and ε-expansion-based approximation are shown in Figure 5.8. One can verify that the desired property is respected.

5.5.2 Cruise Control

Here, a cruise control problem for a truck with a trailer as in Figure 5.9, with dynamics as in (2.3.2), where

$$A := \begin{bmatrix} 0.8855 & -0.3628 & 0.3628 \\ 0.4081 & 0.4683 & 0.5317 \\ 0 & 0 & 1.0000 \end{bmatrix}, \; B := \begin{bmatrix} 0.1018 \\ 0.1372 \\ 0.5000 \end{bmatrix}, \tag{5.5.1}$$

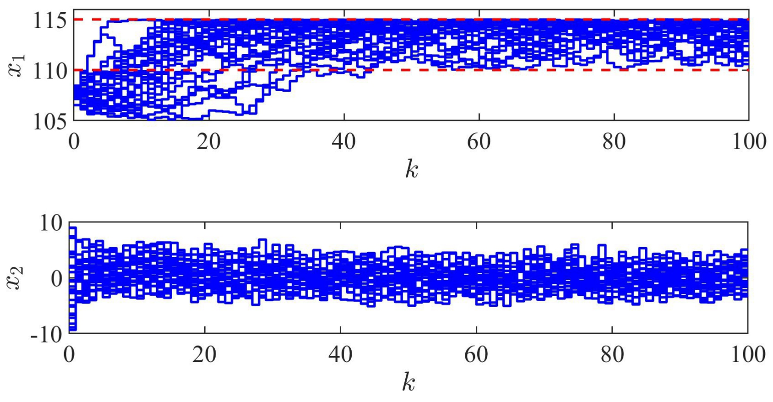


Figure 5.8: Simulation of the running example with the controllers associated with the maximal HCI set, the (ε_x,ε_u)-contraction-based approximation and the ε-expansion-based approximation.

$x(k) = [x_1(k); x_2(k); x_3(k)]$ is the state of the system, in which $x_1(k)$, $x_2(k)$, and $x_3(k)$

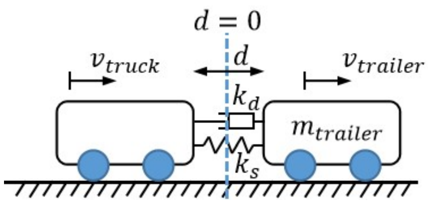


Figure 5.9: Cruise control problem for a truck with a trailer, with $m_{trailer} = 4000$kg the mass of the trailer, $k_s = 4500$N/kg and $k_d = 4600$Ns/m the constants for the spring-damper system, and d the distance between the truck and the trailer, where $d = 0$m is the position at which there is no deformation on the spring.

are the distance between the truck and the trailer, the velocity of the trailer, and the velocity of the truck, respectively. Moreover, $u(k) \in [-5, 5]\mathrm{m/s}^2$ denotes the acceleration of the truck that is used as the control input; and $w(k) \in [-0.04, 0.04] \times [-0.02, 0.02]^2$ denotes the exogenous disturbances encompassing the model uncertainty and unexpected interferences. The model as in (5.5.1) is adapted from [164] by discretizing it with a sampling time $\Delta t = 0.5s$ and including exogenous disturbances. In this case study, the distance between the truck and the trailer should be within $[-1, 1]$m to protect the spring-damper system, and the velocity of the truck and the trailer should be within $[5, 35]$m/s due to the traffic rules. Additionally, to increase the throughput of the road traffic, the truck is not allowed to move slower than 15m/s unless it has moved faster than 25m/s. Such a property, denoted by ψ_q, can be modeled by a DSA $\mathcal{A}_q$ as depicted in Figure 5.10. To synthesize controllers enforcing ψ_q, $\underline{E} := \cup_{\forall q' \in Q}\big(q', q_2, \underline{X}(q', q_2)\big)$ is selected. Additionally, to ensure the compactness of $\underline{X}\backslash\underline{E}$, $\underline{X}\backslash\underline{E}$ is slightly deflated such that $\underline{X}\backslash\underline{E}(q_0, q_1) := [-1, 1] \times [5, 35] \times [20 + \epsilon, 35]$,

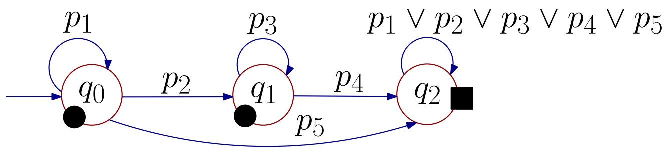


Figure 5.10: DSA $\mathcal{A}_q$ modeling ψ_q, with alphabet $\Pi = \{p_1, p_2, p_3,\ p_4, p_5\}$; labeling function $L : X \to \Pi$ with $L(x) = p_1$ when $x \in [-1, 1] \times [5, 35] \times [15, 25]$, $L(x) = p_2$ when $x \in [-1, 1] \times [5, 35] \times (25, 35]$, $L(x) = p_3$ when $x \in [-1, 1] \times [5, 35] \times [5, 35]$, $L(x) = p_4$ when $x \in \mathbb{R}^3 \backslash L^{-1}(p_3)$, and $L(x) = p_5$ when $x \in \mathbb{R}^3 \backslash (L^{-1}(p_1) \cup L^{-1}(p_2))$; and accepting condition $\text{Acc} = \{\langle E_1, F_1 \rangle\}$, with $E_1 = \{q_3\}$, $F_1 = \emptyset$. The temporal logics formula for ψ_q is given by $G((p_1 U p_2) \wedge (\neg p3))$.

	$\underline{I}^*$	$\underline{I}(\varepsilon_x, \varepsilon_u)$	$\underline{I}(\varepsilon)$
Number of iterations	5	6	4
Computation time (s)	21.34	19.59	16.12
Number of hyperplanes	120	259	149

Table 5.2: Synthesizing controllers for the cruise control problem by computing: 1) maximal HCI set $\underline{I}^*$; 2) contraction-based approximation $\underline{I}(\varepsilon_x,\varepsilon_u)$ with $n{=}3$ and $\varepsilon_x = \varepsilon_u {=} 0.036$; 3) expansion-based approximation $\underline{I}(\varepsilon)$ with $\varepsilon = 0.002$.

with $\epsilon = 0.001$. The results of controller synthesis are summarized in Table 5.2. Then, 10 initial states are randomly selected from $\underline{I}^*(q_0, q_0)$, $\underline{I}(\varepsilon_x, \varepsilon_u)(q_0, q_0)$, and $\underline{I}(\varepsilon)(q_0, q_0)$, respectively, and simulate the systems for 60 seconds (i.e. 120 time steps). Moreover, the disturbances are randomly generated at each time step following a uniform distribution within the disturbance set. The simulation results are shown in Figure 5.11, indicating that the desired property is enforced (note that trajectories of x_3 become red after x_3 has been larger than 25m/s). Additionally, Figure 5.12 shows that there is no exponential growth as in (5.4.4) and (5.4.5) in this case study.

5.5.3 Quadrotor Helicopter

In this subsection, a controller synthesis problem of a quadrotor moving in a 2-dimensional plane (x-y plane) with different regions (definition of these regions comes later) is considered. The model applied here is adapted from [68] by discretizing it with a sampling time $\Delta t{=}0.1s$ and including disturbances that encompass the model uncertainty and unexpected interference on the position and velocity of the quadrotor. Concretely, the dynamics of the quadrotor helicopter is as in (2.3.2), in which

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } B = \begin{bmatrix} \Delta t^2/2 & 0 \\ \Delta t & 0 \\ 0 & \Delta t^2/2 \\ 0 & \Delta t \end{bmatrix}.$$

Here, $x(k) = [x_1(k); x_2(k); x_3(k); x_4(k)]$ is the state of the system, where $x_1(k)$, $x_2(k)$, $x_3(k)$ and $x_4(k)$ are the position on the x axis, the velocity on the x axis, the position

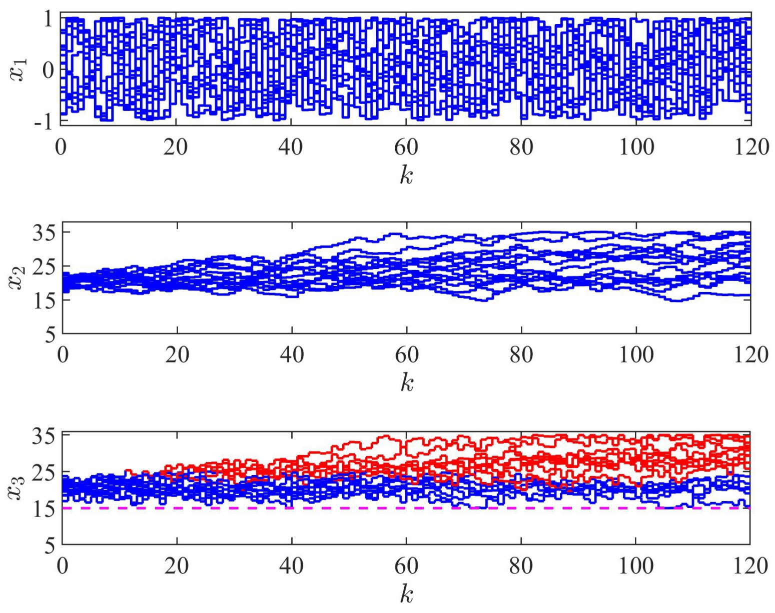


Figure 5.11: Simulation of the cruise control problem

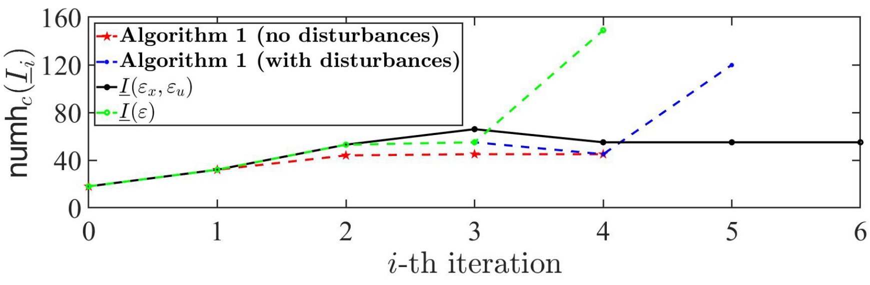


Figure 5.12: Evolution of the number of hyperplanes required to characterize $\underline{I}_i$, denoted by $\mathsf{numh}_c(\underline{I}_i)$, as i increases.

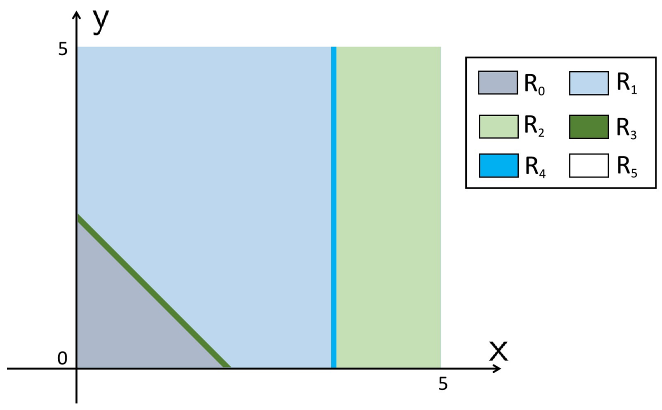


Figure 5.13: Different regions for the quadrotor on an x-y plane.

on the y axis, and the velocity on the y axis, respectively. Moreover, $u(k) \in [-0.9, 0.9]^2$ denotes the acceleration of the quadrotor on both axes and is used as the control input; and $w(k) \in [-0.02, 0.02]^4$ denotes the exogenous disturbances affecting the positions and velocities of the quadrotor helicopter.

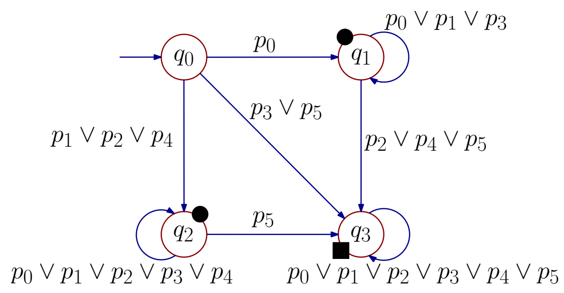


Figure 5.14: DSA $\mathcal{A}$ modeling ψ_q for the quadrotor helicopter, with ■ and ● indicate the states that can be visited finitely many and infinitely many times, respectively.

In this case study, a property ψ_q that is modeled by the DSA $\mathcal{A}$ depicted in Figure (5.14) is considered, with $E = \{q_3\}$, $F = \{q_1, q_2\}$, alphabet $\Pi = \{p_0, p_1, p_2, p_3, p_4, p_5\}$, and a labelling function $L : X \to \Pi$ with $L(x) = p_i$ when $x \in R_i \subset X$ for all $i \in [0, 5]$, where

$$R_0 := \begin{cases} x_1 \geq 0 \\ x_3 \geq 0 \\ x_1 + x_3 \leq 1.5, \\ -1 \leq x_2 \leq 1 \\ -1 \leq x_4 \leq 1 \end{cases} R_1 := \begin{cases} 0 \leq x_1 \leq 3.499 \\ 0 \leq x_3 \leq 5 \\ x_1 + x_3 \geq 1.501, \\ -1 \leq x_2 \leq 1 \\ -1 \leq x_4 \leq 1 \end{cases} R_3 := \begin{cases} x_1, x_3 \geq 0 \\ 1.5 < x_1 + x_3 < 1.501 \\ -1 \leq x_2 \leq 1 \\ -1 \leq x_4 \leq 1 \end{cases},$$

$R_2 := [3.5, 5] \times [-1, 1] \times [0, 5] \times [-1, 1]$, $R_4 := (3.499, 3.5) \times [-1, 1] \times [0, 5] \times [-1, 1]$, and $R_5 := \mathbb{R}^4 \backslash (\cup_{i=0}^4 X_i)$. To provide more intuition, the projection of R_i on the x-y plane, with $i \in [0, 5]$, are depict in Figure 5.13 . In English, property ψ_q requires that i) the helicopter should avoid $R_2 \cup R_4$ if it starts from R_0; ii) the helicopter should always stay away from R_5; iii) the velocity of the helicopter should be within $[-1, 1]$ m/s on both axes. Accordingly, a controller that enforces ψ_q is synthesized by computing the ε-expansion-based approximation of the maximal HCI set, denoted by $\underline{I}(\varepsilon)$, with $\varepsilon = 0.01$. The computation terminates within 840.81 seconds with 18 iterations, which results in an HCI-set containing 284 hyperplanes.

As for the simulation, 5 initial states are randomly selected from $\underline{I}(\varepsilon)(q_0, q_1)$ and $\underline{I}(\varepsilon)(q_0, q_2)$, respectively, and set the time horizon as 60 seconds (i.e. 600 time steps). Moreover, the disturbances affecting the positions and velocities of the quadrotor are randomly generated at each time step following a uniform distribution within the disturbance set. The simulation results are shown in Figure 5.15, which indicate that the desired property is enforced.

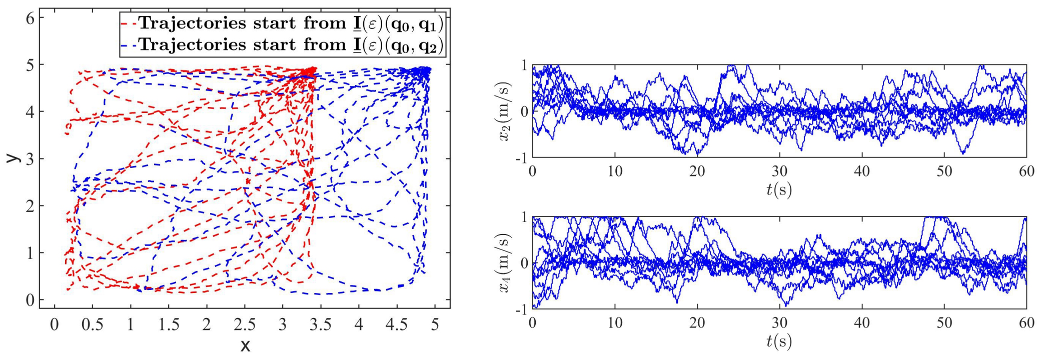


Figure 5.15: (**Left**): Trajectories of the quadrotor on the x-y plane; (**Right**): Evolutions of the quadrotor helicopter's velocity.

5.5.4 Comparison with Existing Results

In this subsection, the proposed set-based approaches are compared with existing results in terms of computation time for synthesizing controllers, including symbolic techniques (`OmegaThreads` [104] and `TuLiP` [64]), interval-analysis-based approaches (`ROCS` [125]), CBF-based approaches [87], and HJ-based approaches (`helperOC` [16] equipped with `toolboxLS` [141]). Moreover, since interval-based approaches do not handle systems with exogenous disturbances [127, Section 2.D], and HJ-based approaches do not handle ω-regular properties, for a fair comparison among these approaches, three different cases are considered: 1) enforcing ψ_q in Session 5.5.2 over the system in (5.5.1); 2) enforcing ψ_q over the system in (5.5.1), but *without exogenous disturbance*; 3) ensuring the system in (5.5.1) reaches the region $[-1,1]\times[5,35]\times[25,35]$ from the region $[-1,1]\times[5,35]\times[5,25]$ within 3 time steps.

Following the same settings as in Table 5.2, $n = 3$ and $\varepsilon_x = \varepsilon_u = 0.036$ are chosen to compute the (ε_x, ε_u)-contraction-based approximation of the maximal HCI-set, and $\varepsilon = 0.002$ is selected to compute the ε-expansion-based approximation. For applying `ROCS`, $\varepsilon = 0.001$ and $\mu = 0.001$ are chosen as the lower bounds of discretization parameters for state and input sets, respectively, (see [127, Section 4-A] for their definitions) for a fair comparison with the setting of ε-expansion-based approach [127, Lemma 1 and Theorem 1]. Moreover, considering the limitation of the computer used for the computation here, 0.2 is used as the discritization parameter for discretizing the state and input sets when deploying `OmegaThreads`, `TuLiP`, and `helperOC`. The computation time for synthesizing controllers with different approaches is summarized in Table 5.3, which indicates that approaches proposed in this chapter require less computation time than other ones. Concretely, >6 h means that the corresponding synthesis procedures did not terminate within 6h, and that the actual computation time is undecided. Additionally, when applying `OmegaThreads`, no controller was found in all cases with the current discretization parameters. Therefore, smaller discretization parameters for the state and input sets are needed to potentially obtain controllers, which would, however,

Methods	Maximal HCI-set	$(\varepsilon_x,\varepsilon_u)$-contraction	ε-expansion	`ROCS`	`TuLiP`	`OmegaThreads`	CBF-based	HJ-based
Case 1	21.34 s	19.59 s	16.12 s	N/A	>6 h	2899.85 s	N/A	N/A
Case 2	2.92 s	7.44 s	7.04 s	>6 h	>6 h	1933.60 s	N/A	N/A
Case 3	51.02 s	86.27 s	44.30 s	N/A	>6 h	7123.81 s	N/A	415.15 s

Table 5.3: Comparison among the proposed HCI-based methods and existing results in terms of computation time for synthesizing controllers enforcing: 1) (Case 1) ψ_q over system in (5.5.1) with disturbances; 2) (Case 2) ψ_q over system in (5.5.1) without disturbances; 3) (Case 3) system in (5.5.1) with disturbances reaching a target set within 3 time steps.

result in longer computation time. As for using CBF-based methods in [9], although the potential control barrier function, the multipliers, and the controller are set as polynomials of up to degree eight, no controller was found in any cases.

5.6 Summary

In this chapter, I proposed for the first time a notion of so-called hybrid controlled invariant set (HCI set), based on which one can synthesize controllers to enforce ω-regular properties over linear control systems affected by bounded disturbances. Given a linear control system and a deterministic Streett automata (DSA) modeling the desired ω-regular property, a product system is first constructed between the linear control system and the DSA. Then, the maximal HCI set is computed by utilizing a set-based approach over the hybrid state set of the product system. Additionally, two approaches are provided to compute approximations of the maximal HCI sets within a finite number of iterations: one by deflating the original state and input sets, the other by expanding the disturbance set. The effectiveness of the proposed methods is shown by three case studies, and by comparison with existing tools.

5.7 Proof of Statements in Chapter 5

5.7.1 Proof of Statements: Section 5.2

Proof of Proposition 5.2.7 Consider any controller sequence $\mu' = \{\mu'_0, \mu'_1, \ldots, \mu'_i, \ldots\}$, with $i \in \mathbb{N}$, associated with $\underline{I}$ such that for any initial state $\underline{x}(0) \in \underline{I}$, and infinite state sequence $\underline{\xi} = \{\underline{x}(0), \underline{x}(1), \ldots, \underline{x}(i), \ldots\}$, one has $\underline{x}(i) \in \underline{I}, \forall i \in \mathbb{N}$, when μ' is applied. Note that such controller sequence exists according to the definition of the HCI set as in Definition 5.2.5. Hence, at time instant $i = 0$, $\forall \underline{x} \in \underline{I}$, $\forall w \in \underline{W}$, one gets $\underline{x}' := \underline{f}(\underline{x}, u, w) \in \underline{I}$ with $u = \mu'_0(\underline{x})$. Since one has $\underline{x}' \in \underline{I}$, then $\forall w' \in \underline{W}$, one again has $\underline{x}'' := \underline{f}(\underline{x}', u, w') \in \underline{I}$ with $u = \mu'_0(\underline{x})$ at time instant $i = 1$. Therefore, one can verify that with the sequence of controller $\mu'' := \{\mu''_0, \mu''_1, \ldots, \mu''_i, \ldots\}$ with $\mu''_i = \mu'_0$, $\forall i \in \mathbb{N}$, one also has $\underline{x}(i) \in \underline{I}, \forall i \in \mathbb{N}$, when μ'' is utilized. Note that μ'' is a stationary controller as in Definition 5.2.6, which completes the proof. ■

Next, I proceed with showing Theorem 5.2.11, which requires additional definitions and lemmas. First, the following set is defined

$$G(\underline{X}') := \{(\underline{x}, u) \in \underline{X} \times \underline{U} \mid \forall w \in \underline{W}, \underline{f}(\underline{x}, u, w) \in \underline{X}'\}, \tag{5.7.1}$$

where $\underline{X}' \subseteq \underline{X}$. Accordingly, consider $\underline{I}_0$ along with the iteration of $\underline{I}_i$ as in (5.2.11), G_i with $i \in \mathbb{N}_{>0}$ are defined as:

$$G_1(\underline{I}_0) := G(\underline{I}_0); \tag{5.7.2}$$

$$G_i(\underline{I}_0) := G(\underline{I}_{i-1}),\ i \geq 2. \tag{5.7.3}$$

Now, based on theses definitions, Lemma 5.7.1 and Lemma 5.7.2 are proposed for proving Theorem 5.2.11.

Lemma 5.7.1. *Consider a dtLCS S as in* (2.3.2)*, a DSA $\mathcal{A}$ modeling the desired ω-regular property, and the product system $S \otimes \mathcal{A} = (\underline{X}, \underline{X}_0, \underline{U}, \underline{W}, \underline{f})$ such that Assumption 5.2.10 holds. Then, $G_i(\underline{I}_0)$ are compact for all $i \in \mathbb{N}_{>0}$, with G_i as defined in* (5.7.2) *and* (5.7.3)*, and $\underline{I}_0$ as in* (5.2.11)*.*

Proof of Lemma 5.7.1 In case that $G_i(\underline{I}_0) = \emptyset$, the assertion of Lemma 5.7.1 holds trivially. Therefore, it is assumed that $G_i(\underline{I}_0) \neq \emptyset$, for all $i \in \mathbb{N}_{>0}$. To this end, $\underline{I}_i$ are shown to be compact for all $i \in \mathbb{N}$. Then, $G_i(\underline{I})$ is shown to be compact when $\underline{I}$ is compact, and the compactness of $(G_i(\underline{I}_0))_{i \in \mathbb{N}_{>0}}$ follows by the compactness of $(\underline{I}_i)_{i \in \mathbb{N}}$. Firstly, $\underline{I}_0$ as in (5.2.11) is compact according to Assumption 5.2.10. Since intersection of two compact sets are still compact, the compactness of $(\underline{I}_i)_{i \in \mathbb{N}}$ can be verified by showing $\mathbf{P}(\underline{I})$ as in (5.2.13) is compact if $\underline{I}$ is compact. For this purpose, $\mathbf{P}(\underline{I})$ is rewritten as $\mathbf{P}(\underline{I}) = \{\underline{x} \in \underline{X} \mid \exists u \in \underline{U}, \text{ s.t. } \underline{f}(\underline{x}, u, \mathbf{0}_n) \in \underline{I}'\}$, with $\underline{I}'$ a bounded set being defined as $\underline{I}' = \{\underline{x}' \mid \{\underline{x}'\} \oplus \underline{W} \subseteq \underline{I}\}$. Consider any $\underline{x}'' := (q, q', x') \notin \underline{I}'$. By definition of $\underline{I}'$, there exists at least one $w \in \underline{W}$ such that one gets $\underline{z} := (q, q', x'+w) \notin \underline{I}$. Since $\underline{I}$ is compact (and therefore closed), then there exists an open ball B in the sense of the distance as defined in (5.2.7) centered at $(q, q', 0)$ such that $(\underline{z}+\mathsf{B}) \cap \underline{I} = \emptyset$, with $\underline{z}+\mathsf{B}$ denotes the Minkowski sum between $\underline{z}$ and B. Accordingly, for any $\underline{x}'' := (q, q', x') \notin \underline{I}'$, one gets $(\underline{x}''+\mathsf{B}) \cap \underline{I}' = \emptyset$, which implies that $\underline{I}'$ is closed. Therefore, $\forall q, q' \in Q$ such that $\underline{I}'(q, q') \neq \emptyset$, $\underline{I}'(q, q')$ is closed and bounded (and therefore compact). Note that Q is a finite set, and finite union of compact sets is still compact. Hence, it is straightforward that $\underline{I}' = \cup_{q,q' \in Q} \underline{I}'(q, q')$ is also compact. Since the dynamics of dtLCS S as in (2.3.2) is continuous, mapping $\underline{f}$ is also continuous. Then, the compactness of $\mathbf{P}(\underline{I})$ follows by the compactness of $\underline{U}$, $\underline{I}$, and $\underline{I}'$.

As for the compactness of $G_i(\underline{I})$ given $\underline{I}$ is compact, $G_i(\underline{I})$ as in (5.7.1) is rewritten as $G(\underline{I}) = \{(\underline{x}, u) \in \underline{X} \times \underline{U} \mid \underline{f}(\underline{x}, u, \mathbf{0}_n) \in \underline{I}'\}$. Then, the compactness of $G(\underline{I})$ can be proved similarly to that of the compactness of $\mathbf{P}(\underline{I})$. ∎

Lemma 5.7.2. *Consider G_i as defined in* (5.7.2) *and* (5.7.3)*, and $\underline{I}_0$ as in* (5.2.11)*. One has*

$$\pi_{\underline{X}}\Big(\bigcap_{i=1}^{\infty} G_i(\underline{I}_0)\Big) = \lim_{i \to \infty} \underline{I}_i, \tag{5.7.4}$$

with $\pi_{\underline{X}}(G_i(\underline{I}_0))$ the projection of $G_i(\underline{I}_0)$ on to $\underline{X}$.

Proof of Lemma 5.7.2 To show Lemma 5.7.2, I show that

1) (**Cond 1**) $\pi_{\underline{X}}\Big(\bigcap_{i=1}^{\infty} G_i(\underline{I}_0)\Big) \subseteq \lim\limits_{i \to \infty} \underline{I}_i$;

2) (**Cond 2**) $\lim\limits_{i \to \infty} \underline{I}_i \subseteq \pi_{\underline{X}}\Big(\bigcap_{i=1}^{\infty} G_i(\underline{I}_0)\Big)$ hold.

First, I show that (**Cond 1**) holds. Let's denote by $\underline{\xi} = \{\underline{x}(0), \underline{x}(1), \ldots, \underline{x}(i), \ldots\}$ an infinite state sequence of $S \otimes \mathcal{A}$. On one hand, according to the definition of $G_i(\underline{I}_0)$, $\pi_{\underline{X}}\Big(\bigcap_{i=1}^{\infty} G_i(\underline{I}_0)\Big)$ denotes the set of $\underline{x} \in \underline{I}_0$, from which there exists a stationary controller $\bar{\mu} = \{\mu, \mu, \ldots\}$ such that $\underline{x}(i) \in \underline{I}_0$, for all $i \in \mathbb{N}$. On the other hand, according to the iteration in (5.2.11), $\lim\limits_{i\to\infty} \underline{I}_i$ denotes the set of all $\underline{x} \in \underline{I}_0$, from which there exists a controller (either stationary or non-stationary) $\bar{\mu}' = \{\mu_1', \mu_2', \ldots\}$ such that $\underline{x}(i) \in \underline{I}_0$ for all $i \in \mathbb{N}$. Therefore, (**Cond 1**) holds.

Next, I show that (**Cond 2**) holds. According to the definition of $G_i(\underline{I}_0)$ and $\underline{I}_i$, one gets

$$\lim_{i\to\infty} \underline{I}_i = \bigcap_{i=1}^{\infty} \pi_{\underline{X}}(G_i(\underline{I}_0)). \tag{5.7.5}$$

Therefore, I proceed with proving

$$\bigcap_{i=1}^{\infty} \pi_{\underline{X}}(G_i(\underline{I}_0)) \subseteq \pi_{\underline{X}}\Big(\bigcap_{i=1}^{\infty} G_i(\underline{I}_0)\Big). \tag{5.7.6}$$

Consider an $\underline{x} \in \bigcap_{i=1}^{\infty} \pi_{\underline{X}}(G_i(\underline{I}_0))$. Then, there exists a sequence $\{u_i\}_{i\in\mathbb{N}}$, such that $(\underline{x}, u_i) \in G_i(\underline{I}_0)$, $\forall i \in \mathbb{N}$. On one hand, according to the computation of $\underline{I}_i$, $i \in \mathbb{N}$ as in (5.2.11), it is straightforward that $\underline{I}_0 \supseteq \underline{I}_1 \supseteq \ldots \supseteq \underline{I}_i \supseteq \ldots$. Then, considering the definition of $G_i((\underline{I}_0))$ as in (5.7.2) and (5.7.3), for all $i \in \mathbb{N}_{>0}$, one has $G_1(\underline{I}_0) \supseteq G_2(\underline{I}_0) \supseteq \ldots \supseteq G_i(\underline{I}_0) \supseteq \ldots$. Hence, $\forall i' \geq i > 0$, if one has $(\underline{x}, u_{i'}) \in G_{i'}(\underline{I}_0)$, then one gets $(\underline{x}, u_{i'}) \in G_i(\underline{I}_0)$. On the other hand, since $G_i(\underline{I}_0)$ are compact according to Lemma 5.7.1, any sequences of elements within $G_i(\underline{I})$ has at least one limit point $(\underline{x}, u) \in G_i(\underline{I})$. This indicate that $\exists (\underline{x}, u) \in G_i(\underline{I}_0)$, $\forall i \in \mathbb{N}_{>0}$, i.e. one has $(\underline{x}, u) \in \bigcap_{i=1}^{\infty} G_i(\underline{I}_0)$. This indicates that $\underline{x} \in \pi_{\underline{X}}\Big(\bigcap_{i=1}^{\infty} G_i(\underline{I}_0)\Big)$, which implies that (5.7.6) holds, and as a result (**Cond 1**) holds. Then, the proof is completed by combining (**Cond 1**) and (**Cond 2**). ■

With Lemma 5.7.1, Lemma 5.7.2, and Proposition 5.2.7, I am ready to prove Theorem 5.2.11.

Proof of Theorem 5.2.11 In case that $\underline{I}^* = \emptyset$, then there exists $i \in \mathbb{N}$ such that for all $i' \geq i$, $\underline{I}_{i'} = \emptyset$ according the iteration as in (5.2.11). Therefore, $\underline{I}^* = \lim\limits_{i\to\infty} \underline{I}_i$ holds trivially. This assertion can be proved by contradiction. Suppose $\underline{I}' := \lim\limits_{i\to\infty} \underline{I}_i \neq \emptyset$. Then, $\forall \underline{x} \in \underline{I}'$, there exists an infinite sequence of inputs $\xi_u(u(0), u(1), \ldots)$ such that the corresponding infinite state sequence $\xi_x(x(0), x(1), \ldots)$ can be enforced within $\underline{I}_0$, i.e. $\underline{I}'$ is an HCI set for $S \otimes \mathcal{A}$. However, this is contradictory to the fact that the maximal HCI set $\underline{I}^*$ is empty.

Next, the case in which $\underline{I}^* \neq \emptyset$ is considered. Considering (5.7.5), $\bigcap_{i=1}^{\infty} \pi_{\underline{X}}(G_i(\underline{I}_0))$ is first shown to be an HCI set for $S \otimes \mathcal{A}$, which implies that

$$\bigcap_{i=1}^{\infty} \pi_{\underline{X}}(G_i(\underline{I}_0)) \subseteq \underline{I}^*, \tag{5.7.7}$$

holds. Consider a controller $\mu : \underline{X} \to \underline{U}$ such that for all $\underline{x} \in \bigcap_{i=1}^{\infty} \pi_{\underline{X}}(G_i(\underline{I}_0))$, $(\underline{x}, \mu(\underline{x})) \in \bigcap_{i=1}^{\infty} G_i(\underline{I}_0)$ (such controller exists according to Lemma 5.7.2 by considering (5.7.4) and (5.7.5)). Then, by definition of $G_i(\underline{I}_0)$, $\forall \underline{x} \in \bigcap_{i=1}^{\infty} \pi_{\underline{X}}(G_i(\underline{I}_0))$, and $\forall w \in \underline{W}$, one gets $\underline{f}(\underline{x}, \mu(\underline{x}), w) \in \bigcap_{i=1}^{\infty} \pi_{\underline{X}}(G_i(\underline{I}_0))$. Therefore, $\bigcap_{i=1}^{\infty} \pi_{\underline{X}}(G_i(\underline{I}_0))$ is an HCI set for $S \otimes \mathcal{A}$ so that (5.7.7) holds according to Definition 5.2.5.

Next, I proceed with showing that

$$\underline{I}^* \subseteq \bigcap_{i=1}^{\infty} \pi_{\underline{X}}(G_i(\underline{I}_0)), \tag{5.7.8}$$

also holds. On one hand, according to Proposition 5.2.7, there exists a HCI-based controller μ, s.t. for all $\underline{x} \in \underline{I}^*$, and for all $w \in \underline{W}$, one gets $\underline{f}(\underline{x}, \mu(\underline{x}), w) \in \underline{I}^*$. On the other hand, by definition of $G_i(\underline{I}_0)$ and the HCI-based controller, one has $(\underline{x}, \mu(\underline{x})) \in \bigcap_{i=1}^{\infty} G_i(\underline{I}_0)$, indicating that $\underline{x} \in \pi_{\underline{X}}\Big(\bigcap_{i=1}^{\infty} G_i(\underline{I}_0)\Big)$. Meanwhile, by (5.7.4) and (5.7.5), one has $\underline{x} \in \bigcap_{i=1}^{\infty} \pi_{\underline{X}}(G_i(\underline{I}_0))$, and hence (5.7.8) also holds. Then, the proof is completed by combining (5.7.5), (5.7.7), and (5.7.8). ∎

5.7.2 Proof of Statements: Section 5.3

First, Proposition 5.7.3 is proposed, which facilitates the proof of Lemma 5.3.3 and Corollary 5.3.4.

Proposition 5.7.3. *If $\exists c_x, c_u \in \mathbb{R}_{>0}$ and $n' \in \mathbb{N}$ such that for all $x \in \mathbb{R}^n$, there exists $\nu : [0, n'] \to \mathbb{R}^m$ with which the following conditions hold:*

- ***(Cd.1)*** *$\xi_x(0) = x$ and $\xi_x(n') = \mathbf{0}_n$ with $\xi_x(k+1) = A\xi_x(k) + B\nu(k)$ for all $k \in [0, n']$;*
- ***(Cd.2)*** *$\nu(k) \leq c_u|x|$ holds for all $k \in [0, n']$;*
- ***(Cd.3)*** *$\xi_x(k) \leq c_x|x|$ holds for all $k \in [0, n']$;*

then, for all $\gamma \in \mathbb{R}_{>0}$, one has $\gamma\mathbb{B}^n \subseteq \mathcal{N}_{n'}(\varepsilon_x, \varepsilon_u)$, with $\varepsilon_x = c_x\gamma$ and $\varepsilon_u = c_u\gamma$.

Proof of Proposition 5.7.3 According to Definition 5.3.2, (Cd.1) in Proposition 5.7.3 indicates that there exists some $\varepsilon'_x, \varepsilon'_u \in \mathbb{R}_{>0}$ such that $\xi_x(t) \in \mathcal{N}_{n'-t}(\varepsilon'_x, \varepsilon'_u)$, and then (Cd.2) as well as (Cd.3) guarantee that $\xi_x(t) \in \mathcal{N}_{n'-t}(\varepsilon_x, \varepsilon_u)$ with $\varepsilon_x = c_x|x|$ and $\varepsilon_u = c_u|u|$. Therefore, one has $x \in |x|\mathbb{B}^n \subseteq \mathcal{N}_{n'}(\varepsilon_x, \varepsilon_u)$, which completes the proof. ∎

Now, I am ready to show the proof of Lemma 5.3.3.

Proof of Lemma 5.3.3 The proof of Lemma 5.3.3 is given by leveraging Proposition 5.7.3. Concretely, existence of c_x and c_u is shown when $n' = n$ such that (Cd.1), (Cd.2) and (Cd.3) are fulfilled. Considering any $x \in \mathbb{R}^n$, (Cd.1) requires that there exists a control sequence

$$\nu = [\nu(n-1)^\top; \nu(n-2)^\top; \ldots; \nu(0)^\top], \tag{5.7.9}$$

with $\nu(k) \in \mathbb{R}^m$ for all $k \in [0, n-1]$, such that $\xi_x(n) = A^n x + \mathcal{C}\nu = \mathbf{0}_n$, with $\mathcal{C} = [B; AB; \ldots; A^{n-1}B]^\top$ the controllability matrix. Since (A, B) is controllable, one has $rank(\mathcal{C}) = n$, indicating the existence of such control sequence. Therefore, (Cd.1) holds. Let $\mathcal{C}' \in \mathbb{R}^{n\times n}$ be a matrix that contains n linearly independent columns of $\mathcal{C}$. Here, ν is selected as in (5.7.9) by setting the entries $\underline{\nu}$ of ν associated with $\mathcal{C}'$ of $\mathcal{C}$ as $\underline{\nu} = -(\mathcal{C}')^{-1}A^n x$, and the remaining entries of ν as zero. Accordingly, one can verify that $\xi_x(n) = A^n x + \mathcal{C}\nu = \mathbf{0}_n$ holds with such ν. In this case, since $|\underline{\nu}| \leq |(\mathcal{C}')^{-1}A^n||x|$ holds, (Cd.2) also holds with

$$c_u = |(\mathcal{C}')^{-1}A^n|. \tag{5.7.10}$$

Meanwhile, by applying the same ν, one obtains

$$\xi_x(k) = A^k x + \sum_{t'=0}^{k-1} A^{k-t'-1}B\nu(t').$$

Accordingly, one has

$$\begin{aligned}|\xi_x(k)| = |A^k x + \sum_{t'=0}^{k-1} A^{k-t'-1}B\nu(t')| &\leq |A^k||x| + |\sum_{t'=0}^{k-1} A^{k-t'-1}B||\nu(t')| \\ &\leq (|A^k| + |\sum_{t'=0}^{k-1} A^{k-t'-1}B|c_u)|x|,\end{aligned}$$

with c_u as in (5.7.10) and $|A^k|$ the infinity norm of matrix A^k. Hence, (Cd.3) holds with

$$c_x = \max_{k\in[0,n]}(|A^k| + |\sum_{t'=0}^{k-1} A^{k-t'-1}B|c_u),$$

which completes the proof. ■

Proof of Corollary 5.3.4 Consider c_x, c_u, n', and u_j with $j \in [0, n'-1]$ such that (5.3.3) to (5.3.5) holds. Corollary 5.3.4 is proved by showing that (Cd.1), (Cd.2) and (Cd.3) in Proposition 5.7.3 also hold for all $x \in \mathbb{R}^n$ with the same c_x, c_u, and n'. For any $x' \in \mathbb{R}^n$ with $|x'| = \beta$ and $\beta \in \mathbb{R}_{\geq 0}$, $u'_j \leq \beta u_j$ is considered with $|u_j| \leq c_u$ for all $j \in [0, n'-1]$, and $z'_i \in \mathbb{R}^n$, with $i \in [1, 2^n]$, which are the vertices of the $\beta\mathbb{B}^n$. Firstly, one has

$$A^{n'}z'_i + \sum_{j=0}^{n'-1} A^{n'-j-1}Bu'_j = \beta(A^{n'}z_i + \sum_{j=0}^{n'-1} A^{n'-j-1}Bu_j) = \mathbf{0}_n. \tag{5.7.11}$$

As a result, (Cd.1) holds for all z'_i, with $i \in [1, 2^n]$. Secondly, one also has

$$|u'_j| = |\beta u_j| \leq c_u\beta, \forall j \in [0, n'-1], \tag{5.7.12}$$

which implies that condition (Cd.2) holds. Finally, for all $d \in [1, n'-1]$, one can verify that

$$|A^d z_i' + \sum_{j=0}^{d-1} A^{d-j-1} B u_j'| \le |A^d z_i + \sum_{j=0}^{d-1} A^{d-j-1} B u_j||\beta| \le c_x \beta, \tag{5.7.13}$$

hold. Hence, (Cd.3) also holds for all z_i', with $i \in [1, 2^n]$. Note that due to the convexity of $\beta\mathbb{B}^n$ and the linearity of (2.3.2), it is sufficient to show that (Cd.1) and (Cd.3) hold for all $x' \in \mathbb{R}^n$ with $|x'| = \beta$ by showing (5.7.11) and (5.7.13) hold for all z_i' with $i \in [1, 2^n]$. Therefore, the proof can be completed by combining (5.7.11), (5.7.12), and (5.7.13). ■

Proof of Theorem 5.3.6 First, the existence of $i \in \mathbb{N}$ such that (5.3.7) holds is shown. To this end, two cases are considered:

1. In case that $\underline{I}_i = \emptyset$ for some $i \in \mathbb{N}$, then $\forall i' \ge i$, one gets $\underline{I}_{i'} = \emptyset$, since $(\emptyset)_\gamma = \emptyset$ for any $\gamma \in \mathbb{R}_{>0}$ such that (5.3.7) holds.
2. In case that $\underline{I}_i \neq \emptyset$ for all $i \in \mathbb{N}$, one can verify from Theorem 5.2.11 that for any $\gamma \in \mathbb{R}_{>0}$, there exists $i \in \mathbb{N}$ such that for all $i' \ge i$, $\mathsf{d}_H(\underline{I}^*, \underline{I}_{i'}) < \gamma$. Additionally, considering the computation of $\underline{I}_i$, $i \in \mathbb{N}$ as in (5.2.11), one can verify that $\underline{I}_0 \supseteq \underline{I}_1 \supseteq \ldots \supseteq \underline{I}_i \supseteq \ldots$. Therefore, one has $\underline{I}_i \subseteq (\underline{I}_{i'})_\gamma$.

Thus, the proof of the existence of i is concluded by combining both cases above. Next, I proceed with showing that $\underline{I}(\varepsilon_x, \varepsilon_u)$ as in (5.3.9) is an HCI set for $\mathcal{S} \otimes \mathcal{A}$. Here, only the case in which $\underline{I}(\varepsilon_x, \varepsilon_u) \neq \emptyset$ is discussed since $\emptyset$ is a trivial solution of an HCI set for $\mathcal{S} \otimes \mathcal{A}$. Consider any $\underline{x} = (q, q', x) \in \underline{I}(\varepsilon_x, \varepsilon_u)$. Then, by definition of $\underline{I}(\varepsilon_x, \varepsilon_u)$ as in (5.3.9), there exists an $i' \in [1, n']$ such that $\underline{x} \in \underline{I}_{i_*+i'} \oplus \mathcal{N}_{i'}(\varepsilon_x, \varepsilon_u)$. Without loss of generality, it is assumed that $x = x_1 + x_2$, with $x_1 = \underline{I}_{i^*+i'}(q, q')$ and $x_2 \in \mathcal{N}_{i'}(\varepsilon_x, \varepsilon_u)$. On one hand, there exists $u_2 \in \varepsilon_u \mathbb{B}^m$ such that $x_2' := Ax_2 + Bu_2 \in \mathcal{N}_{i'-1}(\varepsilon_x, \varepsilon_u)$. On the other hand, let $\underline{x}_1 = (q, q', x_1)$. Considering the iteration in (5.3.6), there exists $u_1 \in U - \varepsilon_u \mathbb{B}^m$ such that for all $w \in W$, $\underline{x}_1' := (q', q'', x_1') \in \underline{I}_{i_*+i'-1}(q', q'')$ hold, with $x_1' = Ax_1 + Bu_1 + w$ and $(q', L(x_1'), q'') \in \delta$. Then, one can readily verify that for all $w \in W$, there exists $u = u_1 + u_2 \in U$ such that $\underline{x}' \in \underline{I}_{i_*+i'-1} \oplus \mathcal{N}_{i'-1}(\varepsilon_x, \varepsilon_u)$ for all $\underline{x}' = \underline{f}(\underline{x}, u, w)$. Now, one has the following two cases regarding different i':

1. (Case 1) If $i' \ge 2$, one has $\underline{x}' \in \underline{I}(\varepsilon_x, \varepsilon_u)$ by definition of $\underline{I}(\varepsilon_x, \varepsilon_u)$;
2. (Case 2) If $i' = 1$, then according to (5.3.7), one gets $\underline{x}' \in \underline{I}_{i_*} \subseteq (\underline{I}_{i_*+n})_\gamma$. Additionally, considering (5.2.3) and Lemma 5.3.3, $\gamma\mathbb{B}^n \subseteq \mathcal{N}_n(\varepsilon_x, \varepsilon_u)$ implies that $(\underline{I}_{i^*+n})_\gamma \subseteq \underline{I}_{i^*+n} \oplus \mathcal{N}_n(\varepsilon_x, \varepsilon_u)$. Therefore, $\underline{x}' \in \underline{I}(\varepsilon_x, \varepsilon_u)$ holds.

Combining Case 1 and Case 2, one can verify that $\underline{I}(\varepsilon_x, \varepsilon_u)$ is an HCI set for $\mathcal{S} \otimes \mathcal{A}$ according to Definition 5.2.5 . ■

Proof of Theorem 5.3.7 Consider any $\rho \in \mathbb{R}_{>0}$. Here, it is assumed that $\underline{I}^*_\rho \neq \emptyset$, since (5.3.10) holds trivially when $\underline{I}^*_\rho = \emptyset$. For the following discussion, I define

$$(S \otimes \mathcal{A})_{(-\varepsilon_x, -\varepsilon_u)} := (\underline{X}_{-\varepsilon_x}, (\underline{X}_0)_{-\varepsilon_x}, \underline{U} - \varepsilon_u \mathbb{B}^m, \underline{W}, \underline{f}).$$

Then, I show that the assertion of Theorem 5.3.7 holds if $\gamma = \min(\rho/c_x, \rho/c_u)$. If $\gamma = \rho/c_x$, this implies that $c_u \leq c_x$. Consider the maximal HCI set $\underline{I}^*_{(\varepsilon_x,\varepsilon_u)}$ for the product system $(S \otimes \mathcal{A})_{(-\varepsilon_x,-\varepsilon_u)}$. On one hand, one has

$$\underline{I}^*_\rho \subseteq \underline{I}^*_{(\varepsilon_x,\varepsilon_u)}, \tag{5.7.14}$$

according to the definition of $(S \otimes \mathcal{A})_{-\rho}$, since $\varepsilon_x = \rho$ and $\varepsilon_u \leq \rho$. On the other hand, in the view of the definition of an HCI set and the iteration as in (5.3.6), one has

$$\underline{I}^*_{(\varepsilon_x,\varepsilon_u)} \subseteq \underline{I}_i, \tag{5.7.15}$$

for all $i \in \mathbb{N}$, with $\underline{I}_i$ being obtained through the iteration as in (5.3.6). Then, one can readily see that (5.3.10) holds according to the definition of $\underline{I}(\varepsilon_x, \varepsilon_u)$ as in (5.3.9).

If $\gamma = \rho/c_u$, one can similarly show that (5.3.10) holds. As a key insight, $\gamma = \rho/c_u$ implies $c_x \leq c_u$, and hence one has $\varepsilon_x \leq \rho$ and $\varepsilon_u = \rho$ for $(S \otimes \mathcal{A})_{(-\varepsilon_x,-\varepsilon_u)}$. Then, one also has (5.7.14) and (5.7.15), which completes the proof. ■

Proof of Theorem 5.3.10 The existence of $i \in \mathbb{N}$ such that (5.3.12) holds can be proved similarly to the existence of i in Theorem 5.3.6. Therefore, I proceed with showing that $\underline{I}(\varepsilon)$ in (5.3.14) is an HCI set for $\mathcal{S} \otimes \mathcal{A}$. Here, only the case in which $\underline{I}(\varepsilon) \neq \emptyset$ is discussed since $\emptyset$ is a trivial solution of an HCI set for $\mathcal{S} \otimes \mathcal{A}$. On one hand, (5.3.12) implies that $\forall q, q' \in Q$ with $\underline{I}_{i^*}(q, q') \neq \emptyset$, $\underline{I}_{i^*}(q, q') \subseteq \underline{I}_{i^*+1}(q, q') \oplus \varepsilon\mathbb{B}^n$ hold. Hence, one has $(\underline{I}_{i^*})_{-\varepsilon} \subseteq \underline{I}_{i^*+1}$. On the other hand, (5.3.11) shows that $\forall \underline{x} := (q, q', x') \in \underline{I}_{i^*+1}$, $\forall w \in W + \varepsilon\mathbb{B}^n$, $\exists u \in U$ such that $\underline{x}' \in \underline{I}_{i^*}$ holds, with $\underline{x}' = \underline{f}(\underline{x}, u, w)$. This indicates that $\forall \underline{x} := (q, q', x') \in \underline{I}_{i^*+1}$ and $\forall w' \in W$, $\exists u \in U$ such that one has $\underline{x}'' \in (\underline{I}_{i^*})_{-\varepsilon} \subseteq \underline{I}_{i^*+1}$, with $\underline{x}'' = \underline{f}(\underline{x}, u, w')$. Therefore, $\underline{I}_{i^*+1}$ is an HCI set for the product system $\mathcal{S} \otimes \mathcal{A}$ according to Definition 5.2.5, which completes the proof. ■

Proof of Theorem 5.3.11 Consider any $\rho \in \mathbb{R}_{>0}$. If $\underline{I}^*_\rho = \emptyset$, (5.3.15) holds trivially. Therefore, I focus on the case in which $\underline{I}^*_\rho \neq \emptyset$. In the rest of this proof, I show that the assertion of Theorem 5.3.11 holds with

$$\varepsilon = \min(\frac{\rho}{n' c_x}, \frac{\rho}{n' c_u}), \tag{5.7.16}$$

in which c_x, c_u, and n' are those in Corollary 5.3.4 such that (5.3.3)-(5.3.5) hold. To this end, the following set is defined

$$\underline{X}' := \bigcup_{i' \in [1,n']} (\underline{I}^*_\rho \oplus \mathcal{N}_{i'}(\varepsilon_x, \varepsilon_u)), \tag{5.7.17}$$

in which ε_x and ε_u are computed based on $\gamma = \varepsilon$ as in Lemma 5.3.3, with ε as in (5.7.16). Accordingly, one can verify

$$\varepsilon\mathbb{B}^n \subseteq \mathcal{N}_{n'}(\varepsilon_x, \varepsilon_u), \tag{5.7.18}$$

by leveraging Lemma 5.3.3. Moreover, one gets $\mathcal{N}_{i'}(\varepsilon_x, \varepsilon_u) \subseteq \varepsilon_x\mathbb{B}^n$ according to (5.3.1), $\varepsilon_x\mathbb{B}^n \subseteq \frac{\rho}{n'}\mathbb{B}^n$ for all $i' \in [1, n']$ according to (5.7.16), and $\underline{I}^*_\rho \subseteq (\underline{I}_0)_{-\rho}$ according to the definition of HCI sets as in Definition 5.2.5. Therefore, one has

$$\underline{X}' \subseteq (\underline{I}_0)_{-\rho} \oplus (n' \times \frac{\rho}{n'}\mathbb{B}^n) = \underline{I}_0, \tag{5.7.19}$$

with $\underline{I}_0$ as in (5.3.11). Now, I start proving Theorem 5.3.11.

Consider any $\underline{x} := (q, q', x) \in \underline{X}'$. Without loss of generality, it is assumed that $x = \tilde{x} + \sum_{i=1}^{n'} x_i$, with $\tilde{x} \in \underline{I}^*_\rho(q, q')$, and $x_i \in \mathcal{N}_i(\varepsilon_x, \varepsilon_u)$ for all $i \in [1, n']$. Since $\underline{\tilde{x}} := (q, q', \tilde{x}) \in \underline{I}^*_\rho$, then $\exists \tilde{u} \in \underline{U} - \rho\mathbb{B}^m$ such that for all $w \in \underline{W}$, one gets $(q', q'', \underline{\tilde{x}}') := \underline{f}(\underline{\tilde{x}}, u, w) \in \underline{I}^*_\rho$, with $\underline{\tilde{x}}' = A\underline{\tilde{x}} + B\tilde{u} + w$ and $(q', L(\underline{\tilde{x}}'), q'') \in \delta$. Accordingly, considering (5.7.18), there also exists $\tilde{u} \in \underline{U} - \rho\mathbb{B}^m$ such that for all $w' \in \underline{W} + \varepsilon\mathbb{B}^n$,

$$(q', \tilde{q}'', \underline{\tilde{x}}'') := \underline{f}(\underline{\tilde{x}}, u, w') \in \underline{I}^*_\rho \oplus \mathcal{N}_{n'}(\varepsilon_x, \varepsilon_u), \tag{5.7.20}$$

hold, with $\underline{\tilde{x}}'' = A\underline{\tilde{x}} + B\tilde{u} + w'$ and $(q', L(\underline{\tilde{x}}''), \tilde{q}'') \in \delta$. Moreover, according to Definition 5.3.2, for any $x_i \in \mathcal{N}_i(\varepsilon_x, \varepsilon_u)$ with $i \in [1, n']$, there exists $u_i \in \varepsilon_u\mathbb{B}^m$ such that

$$Ax_i + Bu_i \in \mathcal{N}_{i-1}(\varepsilon_x, \varepsilon_u). \tag{5.7.21}$$

Combining (5.7.20) and (5.7.21), one can readily see that for any $\underline{x} := (q, q', x) \in \underline{X}'$, for all $w' \in \underline{W} + \varepsilon\mathbb{B}^n$, one gets $\underline{x}' := (q', q'', x') \in \underline{X}'$, with $x' = Ax + Bu + w'$, $(q', L(x'), q'') \in \delta$, and $u = \tilde{u} + \sum_{i=1}^{n'} u_i$. Additionally, since $\gamma \leq \frac{\rho}{n' c_u}$ according to (5.7.16), one obtains $\varepsilon_u \leq \frac{\rho}{n'}$ and as a result $u \in \underline{U}$. Hence, considering (5.7.19), one can readily conclude that the set $\underline{X}'$ is an HCI set for a product system $S' \otimes \mathcal{A}$ as defined in Definition 5.2.1, with $S' = (X, X_0, U, W + \varepsilon\mathbb{B}^n, f)$, and hence, one gets $\underline{X}' \subseteq \underline{I}^*(\varepsilon)$, with $\underline{I}^*(\varepsilon)$ being the maximal HCI set of $S' \otimes \mathcal{A}$. Moreover, according to (5.7.17), one can see that $\underline{I}^*_\rho \subseteq \underline{X}' \subseteq \underline{I}^*(\varepsilon)$, which completes the proof, since $\underline{I}^*(\varepsilon) \subseteq \underline{I}(\varepsilon)$ considering (5.2.11), (5.2.12), and (5.3.11). ■

5.7.3 Proof of Statements: Section 5.4

To prove Theorem 5.4.3, the following proposition is required.

Proposition 5.7.4. *Given P-collections $\mathcal{U}_1$ and $\mathcal{U}_2$, one has*

$$\mathsf{larg}(\mathcal{U}_1 \cap \mathcal{U}_2) \leq \mathsf{larg}(\mathcal{U}_1) + \mathsf{larg}(\mathcal{U}_2), \tag{5.7.22}$$

$$\mathsf{num}(\mathcal{U}_1 \cap \mathcal{U}_2) \leq \mathsf{num}(\mathcal{U}_1)\mathsf{num}(\mathcal{U}_2), \tag{5.7.23}$$

$$\mathsf{larg}(pre(\mathcal{U}_1)) \leq \tilde{g}_S(p), \tag{5.7.24}$$

$$\mathsf{num}(pre(\mathcal{U}_1)) \leq \mathsf{num}(\mathcal{U}_1), \tag{5.7.25}$$

in which $\mathsf{larg}(\cdot)$ *and* $\mathsf{num}(\cdot)$ *are defined in* (5.1.4) *and* (5.1.5)*, respectively; $pre(\cdot)$ is as in* (5.2.15)*, with exogenous disturbance set $W = \{\mathbf{0}_n\}$; $\tilde{g}_S(\cdot)$ is as in* (5.4.2)*, and $p = \max_{a \in [1, \mathsf{N}_c]} \mathsf{numh}(\mathcal{P}_a)$, with $\mathcal{U}_1 = \cup_{a=1}^{\mathsf{N}_c}(\mathcal{P}_a)$.*

Proof of Proposition 5.7.4 (5.7.22) and (5.7.23) hold trivially according to how the intersection between two P-collection is computed, and (5.7.24) holds according to the definition for $\tilde{g}_S(\cdot)$. As for (5.7.25), one can verify that

$$\mathsf{num}(pre(\mathcal{U}_1)) = \mathsf{num}(\cup_{a=1}^{\mathsf{N}_c} pre(\mathcal{P}_a)) \leq \cup_{a=1}^{\mathsf{N}_c} \mathsf{num}(pre(\mathcal{P}_a)) \leq \mathsf{N_c}.$$

Note that the last inequality holds since $pre(\mathcal{P}_a)$ is still a polytope given $\mathcal{P}_a$ is polytope [102, Section 3.3.3]. ■

Proof of Theorem 5.4.3 Here, (5.4.4) and (5.4.5) are shown by induction. When $i=1$, for any $q,q',q''\in Q_{rd}$ for which $\exists\sigma_1,\sigma_2\in\Pi$ s.t. $(q,\sigma_1,q')\in\delta$ and $(q',\sigma_2,q'')\in\delta$, one has

$$\begin{aligned}
\mathsf{num}(\underline{I}_1(q,q')) &= \sum_{q''\in Q_{rd}} \mathsf{num}\Big(\underline{I}_0(q,q')\cap pre(\underline{I}_0(q',q''))\Big)\\
&\le \sum_{q''\in Q_{rd}} \mathsf{num}(\underline{I}_0(q,q'))\mathsf{num}(pre(\underline{I}_0(q',q''))) && \text{(c1)}\\
&\le \sum_{q''\in Q_{rd}} \mathsf{M}^2 \le \alpha\mathsf{M}^2; && \text{(c2)}\\
\mathsf{larg}(\underline{I}_1(q,q')) &= \mathsf{larg}\Big(\underline{I}_0(q,q')\cap pre(\underline{I}_0(q',q''))\Big)\\
&\le \mathsf{larg}(\underline{I}_0(q,q')) + \mathsf{larg}(pre(\underline{I}_0(q',q''))) && \text{(c3)}\\
&\le p' + \tilde{g}_S(p') \le g^1(p'). && \text{(c4)}
\end{aligned}$$

Hence, (5.4.4) and (5.4.5) hold for $i=1$. Note that (c1)-(c4) hold according to Proposition 5.7.4. Suppose that (5.4.4) and (5.4.5) hold for $i=k$. Then, for $i=k+1$, one has

$$\begin{aligned}
\mathsf{larg}(\underline{I}_{i+1}(q,q')) &= \mathsf{larg}\Big(\underline{I}_0(q,q')\cap pre(\underline{I}_i(q',q''))\Big)\\
&\le \mathsf{larg}(\underline{I}_0(q,q')) + \mathsf{larg}(pre(\underline{I}_i(q',q''))) \le p' + \tilde{g}_S(g^i(p')) \le g^{i+1}(p').\\
\mathsf{num}(\underline{I}_{i+1}(q,q')) &= \sum_{q''\in Q_{rd}} \mathsf{num}\Big(\underline{I}_0(q,q')\cap pre(\underline{I}_i(q',q''))\Big)\\
&\le \sum_{q''\in Q_{rd}} \mathsf{num}(\underline{I}_0(q,q'))\mathsf{num}(pre(\underline{I}_i(q',q''))) \le \sum_{q''\in Q_{rd}} \mathsf{M}\alpha^i\mathsf{M}^{i+1} \le \alpha^{i+1}\mathsf{M}^{i+2};
\end{aligned}$$

Therefore, (5.4.4) and (5.4.5) also hold for $i=k+1$, which completes the proof. ■

Proof of Corollary 5.4.5 In the following discussion, considering a P-collection $\mathcal{U}=\cup_{a=1}^{\mathsf{N}_c}\mathcal{P}_a$, $\mathsf{numh}_c(\mathcal{U}) := \sum_{a=1}^{\mathsf{N}_c}\mathsf{numh}(\mathcal{P}_a)$ denotes the *total number of hyperplanes defining the polytopes within* $\mathcal{U}$. Then, based on (5.4.4) and (5.4.5), one has

$$\mathsf{numh}_c(\underline{I}_i(q,q')) \le \mathsf{num}(\underline{I}_i(q,q'))\mathsf{larg}(\underline{I}_i(q,q')) \le \alpha^i\mathsf{M}^{i+1}g^i(p').$$

Therefore, I_i contains at most $|\delta|\alpha^i\mathsf{M}^{i+1}g^i(p')$ hyperplanes. Meanwhile, the parameters of these hyperplanes can be stored in a $|\delta|\alpha^i\mathsf{M}^{i+1}g^i(p')$-by-$(n+1)$ matrix. Hence, (5.4.10) is a valid upper bound for the space complexities of Algorithm 5. Next, (5.4.11) is shown to be a valid upper bound for the time complexity of Algorithm 5. First, considering (5.4.4), (5.4.7), (5.7.24), and (5.7.25), one has

$$\begin{aligned}
&\mathsf{num}(pre(\underline{I}_{i-1}(q,q'))) \le \mathsf{num}(\underline{I}_{i-1}(q,q')) \le \alpha^{i-1}\mathsf{M}^i,\\
&\mathsf{larg}(pre(\underline{I}_{i-1}(q,q'))) \le \tilde{g}_S(g^{i-1}(p')).
\end{aligned}$$

Accordingly, in the worst case, one needs to compute the intersection of two P-collections, which contains M and $\alpha^{i-1}\mathsf{M}^i$ polytopes, respectively, to obtain $\underline{I}_0 \cap \mathbf{P}(\underline{I}_i)$. Therefore, the worst-case computation time for computing $\underline{I}_0 \cap \mathbf{P}(\underline{I}_i)$ is

$$|\delta|\alpha^{i-1}\mathsf{M}^{i+1}c_2\big(p', \tilde{g}_S(g^{i-1}(p'))\big)$$

considering the definition of c_2 and $|\delta|$. Then, one can readily verify that (5.4.11) is a valid upper bound for the time complexity of Algorithm 5 by considering the definitions of c_1 and c_3. ■

6 Data-driven Controller Synthesis against Invariance Properties

6.1 Introduction

The results proposed in the previous chapters require knowing the model for the system of interest. Nevertheless, in some cases, obtaining an accurate model requires a significant amount of effort [81], or even if a model is available, it may be too complex to be of any use. Such difficulties motivate researchers to enter the realm of data-driven control methods. In this chapter, I propose a data-driven method for constructing safety controllers along with their associated γ-robust safety invariant sets to enforce invariance properties over control systems (i.e., systems are expected to stay within a safe set). These controllers and sets can be used for constructing the Safe-visor architecture following the basic idea of state-based approaches, as discussed in Section 1.3.

6.1.1 Related Works

In general, data-driven control methods can be classified into indirect and direct approaches. *Indirect data-driven approaches* consist of a system identification phase followed by a model-based controller synthesis scheme. To achieve a rigorous safety guarantee, it is crucial to provide an upper bound for the error between the identified model and the real but unknown model (a.k.a. *identification error*). Among different system identification approaches, *least-squares methods* (see e.g. [130]) are frequently used for identifying linear models. In this case, *sharp error bounds* [173] relate the identification error to the cardinality of the finite data set which is used for the identification task. Computation of such bounds requires knowledge about the distributions of the disturbances (typically i.i.d. Gaussian or sub-Gaussian, see e.g. [136, 135], and references herein). Therefore, computation of these bounds is challenging when dealing with *unknown-but-bounded* disturbances [26], i.e., the disturbances are only assumed to be contained within a given bounded set, but their distributions are fully unknown. Note that *set-membership identification approaches* (see e.g. [113, 41]) can be applied to identify linear control systems with *unknown-but-bounded disturbances.* Nevertheless, it is still an open problem to provide an upper bound for the identification error when unknown-but-bounded disturbances are involved.

Different from indirect data-driven approaches, *direct data-driven approaches* directly map data into the controller parameters without any intermediate identification phase. Considering systems without being affected by exogenous disturbances, results in [49] propose a data-driven framework to solve linear quadratic regulation (LQR) problems for linear systems. Later on, similar ideas were utilized to design model-reference controllers (see [37, Section 2]) for linear systems [37], and to stabilize polynomial systems [72], switched linear systems [163], and linear time-varying systems [149]. When exogenous disturbances are also involved in the system dynamics, recent results, e.g., [50, 20, 21, 189], can be applied to LQR problems and robust controller design. However, none of these results considers state and input constraints. Hence, they cannot be leveraged to enforce invariance properties. When input constraints are considered, results in [25, 27] provide data-driven approaches for constructing state-feedback controllers to make a given *C-polytope* (i.e., *compact* polyhedral set containing the origin [30, Definition 3.10]) robustly invariant (see [27, Problem 1]). However, when such controllers do not exist for the given C-polytope, one may still be able to find controllers making a subset of this polytope robustly invariant, which is not considered in [25, 27]. Additionally, the approaches in [25, 27] require an individual constraint for each vertex of the polytope (see [25, Section 4] and [27, Theorem 1 and 2]). Unfortunately, given any arbitrary polytope, the number of its vertices grows exponentially with respect to its dimension and the number of hyperplanes defining it in the worst case [56, Section 1].

6.1.2 Contributions

In this chapter, I focus on enforcing invariance properties over unknown linear systems affected by unknown-but-bounded disturbances. Particularly, a direct data-driven approach is proposed for designing safety controllers against these properties. To this end, I first propose so-called *γ-robust safety invariant (γ-RSI) sets* and their associated state-feedback controllers enforcing invariance properties modeled by (possibly *unbounded*) polyhedral safety sets. Then, a data-driven approach is introduced for computing such sets, in which the numbers of constraints and optimization variables grow linearly with respect to the numbers of hyperplanes defining the safety set and the cardinality of the finite data set. Moreover, the relation between the proposed data-driven approach and the condition of *persistency of excitation* [200] is also elaborated, which is a crucial concept in most literature about direct data-driven approaches.

6.1.3 Problem Formulation

In this chapter, discrete-time linear control systems (dtLCS) as in Definition (2.3.1) are of interest, with $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ being some unknown constant matrices; $x(k) \in X$ and $u(k) \in U$, $\forall k \in \mathbb{N}$, being the state and the input vectors, respectively, in which $X \subseteq \mathbb{R}^n$ is the state set,

$$U = \{u \in \mathbb{R}^m | b_j u \leq 1, j = 1, \dots, \mathrm{j}\} \subset \mathbb{R}^m, \tag{6.1.1}$$

is the input set of the system, with $b_j \in \mathbb{R}^m$ being some known vectors; $w(k)$ denotes the exogenous disturbances, where $w(k) \in \Delta(\gamma)$, $\forall k \in \mathbb{N}$, with

$$\Delta(\gamma) = \{w \in \mathbb{R}^n | w^\top w \leq \gamma, \gamma \in \mathbb{R}_{\geq 0}\}. \tag{6.1.2}$$

Note that disturbances being in the form of (6.1.2) are also known as *unknown-but-bounded disturbance with instantaneous constraint* [26], with γ being the disturbance bound that is assumed to be a priori. Finally,

$$X_{1,N} := \begin{bmatrix} x(1) & x(2) & \dots & x(N) \end{bmatrix}, \tag{6.1.3}$$

$$X_{0,N} := \begin{bmatrix} x(0) & x(1) & \dots & x(N-1) \end{bmatrix}, \tag{6.1.4}$$

$$U_{0,N} := \begin{bmatrix} u(0) & u(1) & \dots & u(N-1) \end{bmatrix}, \tag{6.1.5}$$

denote the data collected offline, with $N \in \mathbb{N}$, in which $x(0)$ and $U_{0,N}$ are chosen by the users, while the rest are obtained by observing the state sequence generated by the system in Definition 2.3.1.

In this chapter, invariance properties is considered as the safety properties of interest, which can be modeled by (possibly unbounded) safety sets defined as

$$S := \{x \in \mathbb{R}^n | a_i x \leq 1, i = 1, \dots, \mathrm{i}\} \subset X, \tag{6.1.6}$$

where $a_i \in \mathbb{R}^n$ are some known vectors. With these notions, the main problem of this chapter is formulated as follows.

Problem 6.1.1. *Consider a dtLCS as in Definition 2.3.1, where matrices A and B are unknown, with input set as in* (6.1.1)*, and safety set as in* (6.1.6)*. Using data in* (6.1.3)- (6.1.5)*, design a* safety envelope $\bar{\mathcal{S}} \subseteq S$ *along with a* safety controller $u = Kx$ *(if existing) such that $x(k) \in \bar{\mathcal{S}}$, $\forall k \in \mathbb{N}_{>0}$, if $x(0) \in \bar{\mathcal{S}}$.*

6.2 γ-Robust Safety Invariant Sets

In this section, the computation of γ-robust safety invariant (γ-RSI) sets is proposed assuming matrices A and B in Definition 2.3.1 are known. These sets would be later employed as safety envelopes as defined in Problem 6.1.1. Then, these results would be used in the next subsection to provide the main direct data-driven approach to solve Problem 6.1.1. First, the definition of γ-RSI sets is present as follows.

Definition 6.2.1. *(γ-RSI set) Consider a linear control system as in Definition 2.3.1. A γ-RSI set $\mathcal{S}$ with respect to a safety set S as in* (6.1.6) *is defined as*

$$\mathcal{S} := \{x \in \mathbb{R}^n | x^\top P x \leq 1\} \subset S, \tag{6.2.1}$$

such that $\forall x \in \mathcal{S}$, one has $Ax + Bu + w \in \mathcal{S}$, $\forall w \in \Delta(\gamma)$, when the RSI-based controller

$$u = Kx, \tag{6.2.2}$$

associated with $\mathcal{S}$ is applied in the closed-loop, where $P \in \mathbb{R}^{n\times n}$ is a positive-definite matrix, and $K \in \mathbb{R}^{m\times n}$.

With this definition, the next straightforward result is proposed for Problem 6.1.1, which can readily been verified according to Definition 6.2.1.

Theorem 6.2.2. *Consider a dtLCS as in Definition 2.3.1. If there exists a γ-RSI set $\mathcal{S}$ as in* (6.2.1)*, then one has $x(k) \in \mathcal{S}$, $\forall k \in \mathbb{N}_{>0}$, when the RSI-based controller as in* (6.2.2) *associated with $\mathcal{S}$ is applied, and $x(0) \in \mathcal{S}$.*

Remark 6.2.3. *In this chapter, elliptical-type γ-RSI sets are computed to solve Problem 6.1.1, while computing γ-RSI sets of more general forms, e.g., polyhedral-type sets, is still open. One of the difficulties of computing polyhedral-type γ-RSI sets is to cast the volume of a polyhedral set as a convex objective function [103, Section 2], which is done easily in the elliptical case (cf. Remark 6.2.7). Additionally, consider an n-dimensional polytope $\mathcal{P} \subseteq \mathbb{R}^{\mathsf{n}}$, which is defined by m hyperplanes. The model-based approaches (see e.g. [29]) require an individual constraint for each vertex of $\mathcal{P}$ for synthesizing controllers that make $\mathcal{P}$ a γ-RSI set. Therefore, the exponential growth in the number of vertices with respect to n and m [56, Section 1] may also be a burden for extending the proposed data-driven approach to polyhedral-type γ-RSI sets.*

Using Theorem 6.2.2, the other question is how to compute γ-RSI sets. To do so, the following result is required.

Theorem 6.2.4. *Consider a system as in Definition 2.3.1. For any matrix $K \in \mathbb{R}^{m\times n}$, positive-definite matrix $P \in \mathbb{R}^{n\times n}$, and $\gamma \in \mathbb{R}_{\geq 0}$, one has*

$$\Big((A+BK)x+w\Big)^\top P\Big((A+BK)x+w\Big) \leq 1, \tag{6.2.3}$$

$\forall w \in \Delta(\gamma)$, and $\forall x \in \mathbb{R}^n$ satisfying $x^\top P x \leq 1$, iff $\exists \kappa \in (0,1]$, such that

1. ***(Cond.1)*** *$x^\top (A+BK)^\top P(A+BK)x \leq \kappa$ holds $\forall x \in \mathbb{R}^n$ satisfying $x^\top P x \leq 1$;*
2. ***(Cond.2)*** *$(y+\tilde{d})^\top P(y+\tilde{d}) \leq 1$ holds $\forall y \in \mathbb{R}^n$ satisfying $y^\top P y \leq \kappa$, and $\forall \tilde{d} \in \Delta(\gamma)$.*

The proof of Theorem 6.2.4 is provided in Section 6.6. Some intuitions for Theorem 6.2.4 is provided in Figure 6.1. With this notion, an optimization problem is proposed in the next definition for computing a γ-RSI set for dtLCS as in Definition 2.3.1, assuming that matrices A and B are known.

Definition 6.2.5. *Consider a dtLCS as in Definition 2.3.1 with input constraints in* (6.1.1)*, a safety set S in* (6.1.6)*, $\kappa \in (0,1]$, and $\gamma \geq 0$. An optimization problem, denoted by OP_m, is defined as:*

$$OP_m : \min_{Q,\bar{K}} - \mathit{log}(\mathit{det}(Q)) \tag{6.2.4}$$

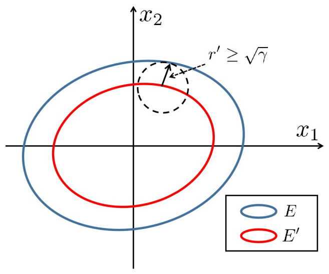


Figure 6.1: An envelope $E := \{x \in \mathbb{R}^n | x^\top P x \leq 1\}$ is a γ-RSI set, when there exists a controller $u = Kx$ that can steer any $x \in E$ into a smaller envelope $E' := \{x^+ \in \mathbb{R}^n | (x^+)^\top P x^+ \leq \kappa\}$ in which it is assumed that $d = \mathbf{0}$, i.e., $\forall x \in E$, one gets $x^+ \in E'$, with $x^+ = (A + BK)x$.

$$s.t. \begin{bmatrix} \kappa Q & Q^\top A^\top + \bar{K}^\top B^\top \\ AQ + B\bar{K} & Q \end{bmatrix} \succeq 0, \tag{6.2.5}$$

$$Q \succeq c\mathbf{I}, \tag{6.2.6}$$

$$a_i Q a_i^\top \leq 1, \, i = 1, \ldots, i, \tag{6.2.7}$$

$$\begin{bmatrix} 1 & b_j \bar{K} \\ \bar{K}^\top b_j^\top & Q \end{bmatrix} \succeq 0, \, j = 1, \ldots, j, \tag{6.2.8}$$

where $c = \frac{\gamma}{(1-\sqrt{\kappa})^2}$ *if* $\kappa \neq 1$, *and* $c = 0$ *otherwise;* $Q \in \mathbb{R}^{n \times n}$ *is a positive-definite matrix, and* $\bar{K} \in \mathbb{R}^{m \times n}$.

Based on Definition 6.2.5, one can construct an RSI-based controller enforcing invariance properties as in the next result.

Theorem 6.2.6. *Consider the optimization problem* OP_m *in Definition 6.2.5. For any* $\kappa \in (0, 1]$ *and* $\gamma \geq 0$, *the set* $\mathcal{S}' := \{x \in X | x^\top Q^{-1} x \leq 1\}$ *is a* γ*-RSI set with* $u = \bar{K} Q^{-1} x$ *being the associated RSI-based controller, if and only if* OP_m *is feasible for the given* γ *and* κ.

The proof for Theorem 6.2.6 can be found in Section 6.6. Note that the existence of $\kappa \in (0, 1]$ is a necessary and sufficient condition for the existence of a γ-RSI set with respect to the safety set S as in (6.1.6) according to Theorem 6.2.4. In practice, one can apply bisection to come up with the largest value of κ while solving OP_m.

Remark 6.2.7. *The objective function in* (6.2.4) *maximizes the volume of the* γ*-RSI set in Theorem 6.2.6, since its volume is proportional to det*(Q) *[36, p. 42].*

So far, an approach for computing γ-RSI sets is proposed by assuming matrices A and B are known. Before proposing the direct data-driven approach with the help of the results in this subsection, it is worthwhile to point out the challenge in solving Problem 6.1.1 using indirect data-driven approaches. Following the idea of indirect data-driven approaches, one needs to identify unknown matrices A and B based on data, and then applies Theorem 6.2.6 to the identified model

$$x(k+1) = \hat{A}x(k) + \hat{B}u(k) + \hat{d}(k),$$

where $\hat{A}$ and $\hat{B}$ are the estimation of A and B, respectively, and $\hat{d}(k) := (A-\hat{A})x(k) + (B-\hat{B})u(k) + w(k)$, with $w(k) \in \Delta(\gamma)$. Accordingly, one has $\|\hat{d}(k)\| \leq \Delta_A\|x(k)\| + \Delta_B\|u(k)\| + \gamma$, with $\Delta_A := \|A-\hat{A}\|$ and $\Delta_B := \|B-\hat{B}\|$. Here, Δ_A and Δ_B are known as *sharp error bounds* [173], which relate the identification error to the cardinality of the finite data set used for system identification. Note that the computation of these bounds requires some assumptions on the distribution of the disturbances (typically disturbances with symmetric density functions around the origin such as Gaussian and sub-Gaussian, see discussion in e.g. [136, 135] and references herein). To the best of my knowledge, it is still an open problem how to compute such bounds when considering unknown-but-bounded disturbances (also see the discussion in Section 6.1). Such challenges in leveraging indirect data-driven approaches motivated me to propose a direct data-driven approach for computing γ-RSI sets, in which the intermediate system identification step is not required.

6.3 Direct Data-driven Computation of γ-RSI Sets

In this subsection, a direct data-driven approach is proposed for computing γ-RSI sets. To this end, the following definition is required.

Definition 6.3.1. *Consider a dtLCS as in Definition 2.3.1 with input constraints as in* (6.1.1)*, a safety set S as in* (6.1.6)*, $X_{1,N}$, $X_{0,N}$, and $U_{0,N}$, as in* (6.1.3)-(6.1.5)*, respectively. Given $\kappa \in (0,1]$ and $\gamma \geq 0$, an optimization problem, denoted by OP_d, is defined as:*

$$OP_d : \min_{Q,\bar{Z},\epsilon_1,\ldots,\epsilon_N} \; -\mathit{log}(\mathit{det}(Q)) \tag{6.3.1}$$

$$s.t.\; Q \succeq c\mathbf{I}, \tag{6.3.2}$$

$$N_1 - \sum_{p=1}^{N} \epsilon_p N_p \begin{bmatrix} \gamma\mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} \end{bmatrix} N_p^\top \succeq 0; \tag{6.3.3}$$

$$a_i Q a_i^\top \leq 1,\; i = 1,\ldots,i, \tag{6.3.4}$$

$$\begin{bmatrix} 1 & b_j\bar{Z} \\ \bar{Z}^\top b_j^\top & Q \end{bmatrix} \succeq 0,\; j = 1,\ldots,j, \tag{6.3.5}$$

where $\epsilon_i > 0$, $\forall i \in [1,N]$,

$$N_1 = \begin{bmatrix} \kappa Q & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -Q & -\bar{Z}^\top & \mathbf{0} \\ \mathbf{0} & -\bar{Z} & \mathbf{0} & \bar{Z} \\ \mathbf{0} & \mathbf{0} & \bar{Z}^\top & Q \end{bmatrix}; N_p = \begin{bmatrix} \mathbf{I}_n & X_{1,N}(p) \\ \mathbf{0} & -X_{0,N}(p) \\ \mathbf{0} & -U_{0,N}(p) \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$\forall p \in [1, N]$; $c = \frac{\gamma}{(1-\sqrt{\kappa})^2}$ if $\kappa \neq 1$, and $c = 0$, otherwise; $Q \in \mathbb{R}^{n\times n}$ is a positive-definite matrix, and $\bar{Z} \in \mathbb{R}^{m\times n}$.

With the help of Definition 6.3.1, the following result can be proposed for building an RSI-based controller with respect to invariance properties.

Theorem 6.3.2. *Consider an optimization problem OP_d as in Definition 6.3.1 and the disturbance set $\Delta(\gamma)$ as in (6.1.2). For any $\kappa \in (0, 1]$, if OP_d is feasible, then the set $\mathcal{S}'_d := \{x \in X | x^\top Q^{-1} x \leq 1\}$ is a γ-RSI set, with $u = \bar{Z}Q^{-1}x$ being the RSI-based controller associated with $\mathcal{S}'_d$.*

The proof of Theorem 6.3.2 is provided in Section 6.6. It is also worth mentioning that the number of LMI constraints in OP_d grows linearly with respect to the number of inequalities defining the safety set in (6.1.6) and input set in (6.1.1). Meanwhile, the sizes of the (unknown) matrices on the left-hand sides of (6.3.2)-(6.3.5) are independent of the number of data, i.e., N, and grow linear with respect to the dimensions of the state and input sets. Additionally, the number of slack variables, i.e., ϵ_i, grows linearly with respect to N. As a result, the optimization problem OP_d in Definition 6.3.1 can be solved efficiently.

In the remainder of this section, the proposed direct data-driven approach is further discussed in terms of the condition of *persistency of excitation* [200] regarding the offline-collected data $X_{0,N}$ and $U_{0,N}$. To this end, the condition of *persistency of excitation* is recalled as below, which is adapted from [200, Corollary 2].

Lemma 6.3.3. *Consider the dtLCS as in Definition 2.3.1 with (A, B) being controllable, $X_{0,N}$ as in (6.1.4), and $U_{0,N}$ as in (6.1.5). One has*

$$rank\Big(\begin{bmatrix} X_{0,N} \\ U_{0,N} \end{bmatrix} \Big) = n + m, \tag{6.3.6}$$

with n and m being the dimensions of state and input sets, respectively, if $U_{0,N}$ is a persistently exciting *input sequence of order $n + 1$, i.e., $rank(U_{0,n+1,N}) = m(n + 1)$, where*

$$U_{0,n+1,N} := \begin{bmatrix} U_{0,N}(1) & U_{0,N}(2) & \dots & U_{0,N}(N-n) \\ U_{0,N}(2) & U_{0,N}(3) & \dots & U_{0,N}(N-n+1) \\ \vdots & \vdots & \ddots & \vdots \\ U_{0,N}(n+1) & U_{0,N}(n+2) & \dots & U_{0,N}(N) \end{bmatrix}.$$

The condition of *persistency of excitation* in Lemma 6.3.3 is common among direct data-driven approaches, since it ensures that the data in hand encode all information which is necessary for synthesizing controllers *directly* based on data [200]. Although Definition 6.3.1 and Theorem 6.3.2 do not require this condition, the next result points out the difficulties in obtaining a feasible solution for OP_d, whenever condition (6.3.6) does not hold.

Corollary 6.3.4. *Consider the optimization problem OP_d in Definition 6.3.1, and the set*

$$\mathcal{F} := \bigcap_{p=1}^{N} \mathcal{F}_p, \tag{6.3.7}$$

where $\mathcal{F}_p := \Big\{(\tilde{A}, \tilde{B}) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times m} \;\Big|\; X_{1,N}(p) = \tilde{A}X_{0,N}(p) + \tilde{B}U_{0,N}(p) + d, d \in \Delta(\gamma)\Big\}$*, in which* $p \in [1, N]$*. The set $\mathcal{F}$ is unbounded if and only if*

$$rank\Big(\begin{bmatrix} X_{0,N} \\ U_{0,N} \end{bmatrix}\Big) < n + m. \tag{6.3.8}$$

The proof of Corollary 6.3.4 can be found in Section 6.6. As a key insight, given data of the form of (6.1.3) to (6.1.5), the failure in fulfilling condition (6.3.6) indicates that these data do not contain enough information about the underlying unknown system dynamics for solving the optimization problem OP_d, since the set of systems of the form of (2.3.2) that can generate the same data is unbounded. Concretely, the optimization problem OP_d aims at finding a common γ-RSI set for any linear system as in Definition 2.3.1 such that $(A, B) \in \mathcal{F}$, with $\mathcal{F}$ as in (6.3.7). The unboundedness of the set $\mathcal{F}$ makes it very challenging to find a common γ-RSI set which works for all $(A, B) \in \mathcal{F}$. In practice, to avoid the unboundedness of $\mathcal{F}$ and ensure that (6.3.6) holds, one can increase the duration of the single input-state trajectory till the condition of persistency of excitation is fulfilled. Before proceeding with introducing the case study of this paper, a flowchart for applying the proposed direct data-driven approach is given in Figure 6.2.

6.4 Case Studies

To demonstrate the effectiveness of the proposed results, they are applied to two case studies. Although the direct data-driven approach proposed in Section 6.3 does not require any knowledge about matrices A and B of the model, model with known A and B are considered in both case studies mainly for collecting data, simulation, and computing the model-based gamma-RSI sets in Theorem 6.2.6 as baselines to evaluate the effectiveness of the proposed direct data-driven approaches (cf. Figure 6.6, 6.7, 6.10, and 6.11). When leveraging the direct data-driven method, it is assumed that A and B are fully unknown and the systems are treated as black-box ones. The experiments are performed via `MATLAB 2019b`, on a machine with Windows 10 operating system (Intel(R) Xeon(R) E-2186G CPU (3.8 GHz)) and 32 GB of RAM. The optimization

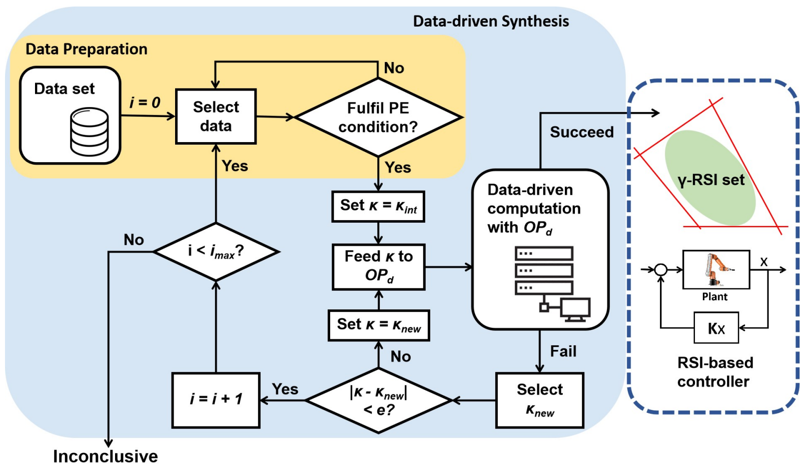


Figure 6.2: Flowchart of the proposed direct data-driven approach, with OP_d and κ as in Definition 6.3.1, $\kappa_{int} \in (0, 1]$, $e \in \mathbb{R}_{>0}$, and $i_{max} \in \mathbb{N}_{>0}$ being parameters which are manually selected by users, and PE condition referring to the condition of persistency of excitation as in Lemma 6.3.3.

problems in Section 6.2 and 6.3 are solved by using optimization toolboxes `YALMIP` [131] and `MOSEK` [143].

6.4.1 Inverted Pendulum

In the first case study, a four dimensional linearized model of the inverted pendulum as in Figure 6.3 is considered. The model of the inverted pendulum can be described by the difference equation as in (2.3.2), in which

$$A = \begin{bmatrix} 1 & 0.02 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1.0042 & 0.0194 \\ 0 & 0 & 0.4208 & 0.9466 \end{bmatrix}, B = \begin{bmatrix} 0.0002 \\ 0.0200 \\ -0.0004 \\ -0.0429 \end{bmatrix}, \tag{6.4.1}$$

where $x(k) = [x_1(k); x_2(k); x_3(k); x_4(k)]$ is the state of the system, with $x_1(k)$ being position of the cart, $x_2(k)$ being the velocity of the cart, $x_3(k)$ being the angular position of the pendulum with respect to the upward vertical axis, and $x_4(k)$ being the angular velocity of the pendulum; $u(k) \in [-5, 5]\ m/s^2$ is the acceleration of the cart that is used as the input to the system. The safety objective for the inverted pendulum case study is to keep the position of the cart within $[-1, 1]$ m, and the angular position of the pendulum within $[-\pi/12, \pi/12]$ rad. This model is obtained by discretizing a continuous-time linearized model of the inverted pendulum as in Figure 6.3 with a sampling time $\tau = 0.02s$, and including disturbances $w(k)$ that encompass unexpected interferences and model uncertainties. The disturbances $w(k)$ belong to the set $\Delta(\gamma)$ as in (6.1.2), with $\gamma = (0.05\tau)^2$, which are generated based on a non-symmetric probability

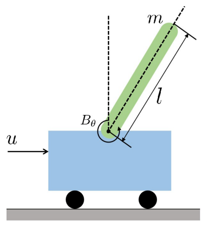


Figure 6.3: Inverted pendulum, where $m = 0.1314\,\mathrm{kg}$ is the mass of the pendulum, $l = 0.68\,\mathrm{m}$ is the length of the pendulum, $g = 9.81\,\mathrm{m/s}$ is the gravitational constant, and $B_\theta = 0.06\,\mathrm{Nm/s}$ is the damping coefficient of the connection between the cart (the blue part) and the pendulum (the green part).

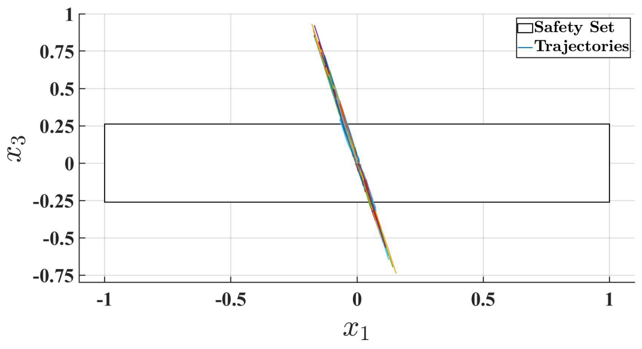


Figure 6.4: Projections of 1000 closed-loop trajectories on $x_1 - x_3$ plane when applying the controller obtained by leveraging indirect data-driven approach.

density function:

$$f(d) := \begin{cases} \dfrac{5}{\pi^2\gamma^2}, & \text{for } d \in D_1; \\ \dfrac{9}{5\pi^2\gamma^2}, & \text{for } d \in \Delta(\gamma)\backslash D_1, \end{cases} \tag{6.4.2}$$

with $D_1 := \{[d_1; d_2; d_3; d_4] \in \Delta(\gamma) | d_i \in \mathbb{R}_{\geq 0}, i \in [1, 4]\}$. Here, the distribution is selected as in (6.4.2) to mainly illustrate the difficulties in identifying the underlying unknown system dynamics when the exogenous disturbances are subject to a non-symmetric distribution, even though they are bounded. Meanwhile, the proposed direct data-driven approaches in this chapter can handle such disturbances since there is no assumption on the disturbance distribution, e.g., being Gaussian or sub-Gaussian. Moreover, this distribution is only used for collecting data and simulation, while the computation of data-driven γ-RSI sets does not require any knowledge of it.

First, the difficulties in applying indirect data driven approaches to solve Problem 6.1.1 is demonstrated, when the bounded disturbances are generated based on a non-symmetric probability density function as in (6.4.2). Here, least-squares approach as in [89] is deployed to identify matrices A and B. Data as in (6.1.3)- (6.1.5)

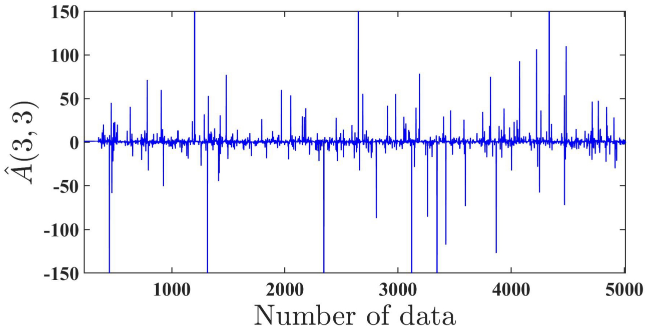


Figure 6.5: Evolution of the entry $\hat{A}(3,3)$ as number of data used for the system identification increases.

is collected, with $N = 500$, and the following estimation of A and B is obtained as

$$\hat{A} = \begin{bmatrix} 1 & 0.02 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0.0764 & -0.0888 & 2.3439 & -0.3745 \\ 0.0687 & -0.0798 & 1.6255 & 0.5924 \end{bmatrix}, \hat{B} = \begin{bmatrix} 0.0002 \\ 0.0200 \\ -0.0003 \\ -0.0422 \end{bmatrix},$$

respectively. Based on the estimated model, one obtains a controller $u = K_i x$ by applying Theorem 6.2.6, with $K_i = \begin{bmatrix} -9.8089; -3.3176; -112.7033; 25.7470 \end{bmatrix}^\top$. With this controller, the system is initialized at $x = \begin{bmatrix} 0; 0; 0; 0 \end{bmatrix}^\top$ and simulated within time horizon $H = 70$. The projections of closed-loop state trajectories on the $x_1 - x_3$ plane are shown in Figure 6.4, which indicate that the desired safety constraints are violated. Additionally, the evolution of the entry $\hat{A}(3,3)$ is also depicted in Figure 6.5 as an example to show that some of the entries in $\hat{A}$ keep fluctuating as the number of data used for system identification increases. In other words, $\hat{A}$ does not seem to converge to the real value in (6.4.1) by increasing the number of data used for system identification.

Next, the direct data-driven approach proposed in this chapter is demonstrated. To compute the data-driven γ-RSI set using Theorem 6.3.2, data as in (6.1.3)-(6.1.5) is collected, with $N = 107$, such that condition (6.3.6) holds. Then, a data-driven γ-RSI set is obtained within 4.165s. Here, the data-driven γ-RSI set is denoted by $\mathcal{S}_d := \{x \in \mathbb{R}^4 | x^\top P_d x \leq 1\}$, with

$$P_d = Q^{-1} = \begin{bmatrix} 3.3950 & 2.8786 & 12.1264 & 1.9861 \\ 2.8786 & 3.8224 & 15.6826 & 2.7404 \\ 12.1264 & 15.6826 & 81.9169 & 12.4079 \\ 1.9861 & 2.7404 & 12.4079 & 2.4531 \end{bmatrix},$$

in which Q is the solution of OP_d with $\kappa = 0.9813$. The RSI-based controller associated with $\mathcal{S}_d$ is $u = K_d x$, where $K_d = \begin{bmatrix} 3.2672; 4.9635; 38.1223; 4.9989 \end{bmatrix}^\top$.

As for the simulation, 100 initial states are randomly selected from $\mathcal{S}_d$ following a uniform distribution. Then, the RSI-based controller associated with $\mathcal{S}_d$ is applied in

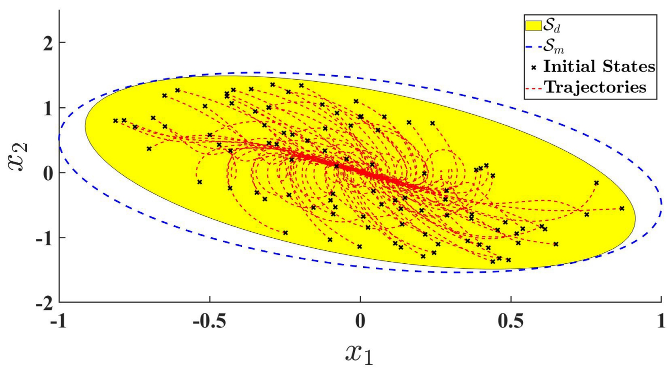


Figure 6.6: Projections of the data-driven γ-RSI set $\mathcal{S}_d$, the model-based γ-RSI set $\mathcal{S}_m$, initial states, and state trajectories on $x_1 - x_2$ plane.

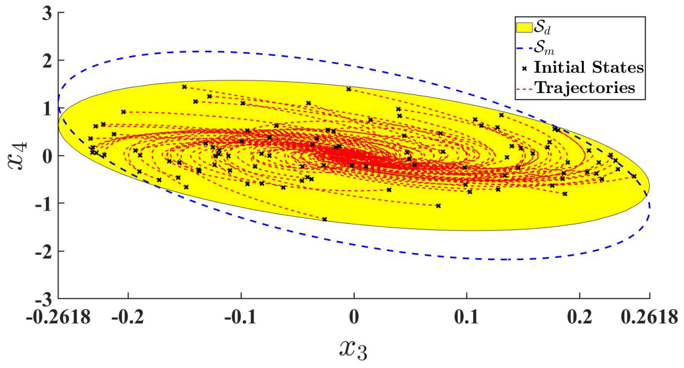


Figure 6.7: Projections of the data-driven γ-RSI set $\mathcal{S}_d$, the model-based γ-RSI set $\mathcal{S}_m$, initial states, and state trajectories on $x_3 - x_4$ plane.

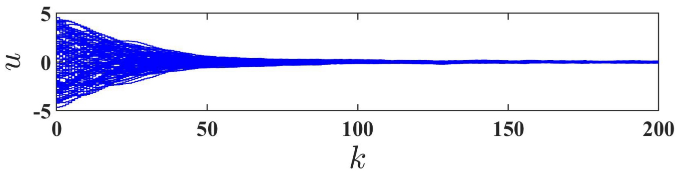


Figure 6.8: Input sequences for the inverted pendulum example.

the closed-loop and the system is simulated within the time horizon $H = 200$. In the simulation, disturbance at each time instant is injected following the distribution in (6.4.2). The projections[1] of the data-driven γ-RSI sets, and closed-loop state trajectories on the $x_1 - x_2$ and $x_3 - x_4$ planes are shown in Figure 6.6 and 6.7, respectively. For comparison, the model-based γ-RSI set is also computed with Theorem 6.2.6, denoted by $\mathcal{S}_m$, and project it onto relevant coordinates. One can readily verify that all trajectories are within the desired safety set, and input constraints are also respected, as displayed in Figure 6.8. It is also worth noting that, as shown in Figure 6.7, the data-driven γ-RSI set does not necessarily need to be inside the model-based one, since the γ-RSI set with the maximal volume (cf. Remark 6.2.7) do not necessarily contain all other possible γ-RSI sets with smaller volume.

6.4.2 3-DOF Helicopter

In the second case study, a case study of a 3-DOF helicopter, as depicted in Figure 6.9 (**Left**), is considered. Here, the 3-DOF helicopter is a simplified helicopter model equipped with two motors that generate upward or downward force according to the actuation voltage.

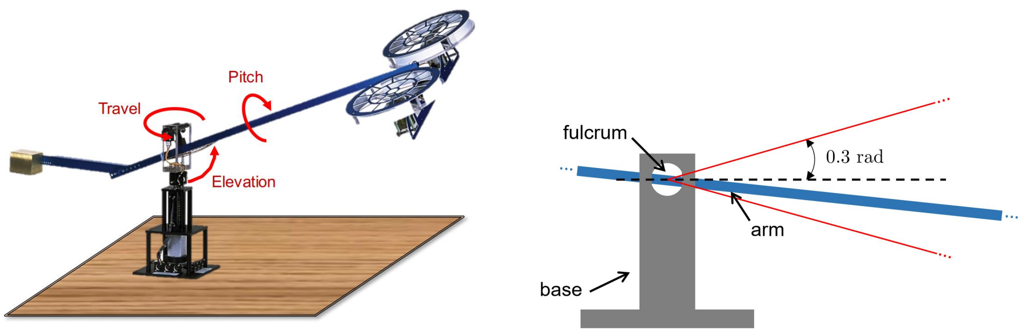


Figure 6.9: (**Left**): 3 Degree of freedom (3-DOF) helicopter; (**Right**): Region between two red lines in which the helicopter fans should stay.

The model of the 3-DOF helicopter can be described by (2.3.2), in which

$$A = \begin{bmatrix} 1 & 0 & 0 & 0.02 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0.02 & 0 \\ 0 & -0.0002 & 1 & 0 & -1.64e^{-6} & 0.02 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -0.0246 & 0 & 0 & -0.002 & 1 \end{bmatrix}, B = \begin{bmatrix} 1.72e^{-5} & 1.72e^{-5} \\ 1.16e^{-4} & -1.16e^{-4} \\ 0 & 0 \\ 0.0017 & 0.0017 \\ 0.0116 & -0.0116 \\ 0 & 0 \end{bmatrix},$$

where $x(k) = [x_1(k); x_2(k); x_3(k); x_4(k); x_5(k); x_6(k)]$ is the state of the system, with $x_1(k)$, $x_2(k)$, and $x_3(k)$, being the elevation, pitch, and travel angles, respectively,

[1]Here, the projections of the γ-RSI sets are computed by leveraging `Ellipsoidal Toolbox` [108].

while $x_4(k)$, $x_5(k)$, and $x_6(k)$, being the elevation, pitch, and travel angular velocities, respectively; $u(k) = [u_1(k); u_2(k)] \in [-24,\ 24] \times [-24\ 24]$ is the input to the system, with $u_1(k)$ and $u_2(k)$ being the actuation voltage of the motors. This model is obtained by discretizing a continuous-time linear model of a 3-DOF helicopter (see the manufacturer manual [157]) with a sampling time $\tau = 0.02s$, and including disturbances $w(k)$ that encompass unexpected interferences and model uncertainties. Moreover, disturbances $w(k)$ belong to the set $\Delta(\gamma)$ as in (6.1.2), with $\gamma = (0.005\tau)^2$, which are generated based on a non-symmetric probability density function:

$$f(d) := \begin{cases} \dfrac{219}{5\pi^3\gamma^3}, & \text{for } d \in D_2; \\ \dfrac{27}{5\pi^3\gamma^3}, & \text{for } d \in \Delta(\gamma)\backslash D_2, \end{cases} \tag{6.4.3}$$

with $D_2 := \{[d_1; d_2; d_3; d_4; d_5; d_6] \in \Delta(\gamma) | d_i \in \mathbb{R}_{\geq 0}, i \in [1, 6]\}$. The safety set is denoted by $S_{\mathsf{heli}} = \{x \in \mathbb{R}^6 | A_h x \leq b_h\}$, in which

$$A_h = \begin{bmatrix} -1 & -0.33 & 0 & 0 & 0 & 0 \\ -1 & 0.33 & 0 & 0 & 0 & 0 \\ 1 & 0.33 & 0 & 0 & 0 & 0 \\ 1 & -0.33 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \text{ and } b_h = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.3 \\ 0.4 \\ 0.4 \\ 1.5 \\ 1.5 \end{bmatrix}.$$

In English, the safety set S_{heli} requires that: 1) the helicopter fans should not exceed the region as illustrated in Figure 6.9 (**Right**); 2) the elevation angular velocity should not exceed 0.4 rad/s; 3) the pitch angular velocity should not exceed 1.5 rad/s.

As for the computation of the data-driven γ-RSI set using Theorem 6.3.2, data as in (6.1.3) - (6.1.5) with $N = 298$ are collected, and a data-driven γ-RSI set is obtained within 6.481s. Note that $N = 298$ is selected here such that condition (6.3.6) holds. Here, the data-driven γ-RSI set and the RSI-based controller associated with $\mathcal{S}_h$ is denoted by $\mathcal{S}_h := \{x \in \mathbb{R}^6 | x^T P_h x \leq 1\}$ and $u = K_h x$, respectively, where

$$P_h = Q^{-1} = \begin{bmatrix} 42.271 & -0.280 & -0.070 & 10.489 & -0.033 & 0.045 \\ -0.280 & 4.649 & -0.291 & -0.034 & 0.347 & -1.544 \\ -0.070 & -0.291 & 0.056 & -0.018 & -0.022 & 0.184 \\ 10.489 & -0.034 & -0.018 & 8.862 & -2.73e^{-5} & -0.037 \\ -0.033 & 0.347 & -0.022 & -2.73e^{-5} & 0.472 & -0.134 \\ 0.045 & -1.544 & 0.184 & -0.037 & -0.134 & 0.899 \end{bmatrix},$$

$$K_h = \begin{bmatrix} -58.318 & -11.075 & 0.7984 & -28.364 & -5.208 & 3.897 \\ -59.858 & 11.741 & -0.605 & -28.381 & 5.246 & -3.879 \end{bmatrix},$$

in which Q is a solution of OP_d, with $\kappa = 0.9938$. In the simulation, 100 initial states are randomly selected from $\mathcal{S}_h$ following a uniform distribution. Then, the RSI-based

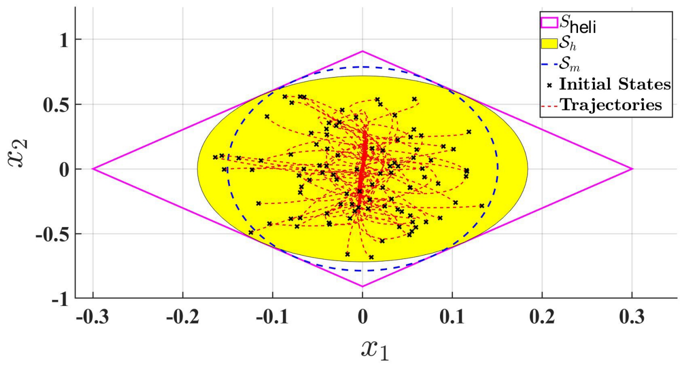


Figure 6.10: Projections of the safety set S_{heli}, the data-driven γ-RSI set $\mathcal{S}_h$, the model-based γ-RSI set $\mathcal{S}_m$, initial states, and state trajectories on $x_1 - x_2$ plane.

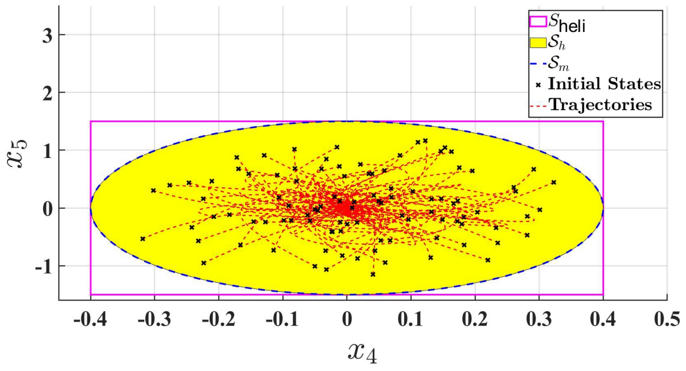


Figure 6.11: Projections of the safety set S_{heli}, the data-driven γ-RSI set $\mathcal{S}_h$, the model-based γ-RSI set $\mathcal{S}_m$, initial states, and state trajectories on $x_4 - x_5$ plane.

controller associated with $\mathcal{S}_h$ is applied in the closed-loop within the time horizon $H = 200$. The projections of the safety set, closed-loop state trajectories, the model-based, and data-driven γ-RSI sets on the $x_1 - x_2$ and $x_4 - x_5$ planes are depicted in Figure 6.10 and 6.11, respectively. Moreover, input sequences are shown in Figure 6.12. One can readily verify that the desired safety set and input constraints are respected.

6.5 Summary

In this chapter, a direct data-driven approach is proposed to synthesize safety controllers, which enforce invariance properties over unknown linear systems affected by unknown-but-bounded disturbances. To do so, a direct data-driven framework is introduced to compute γ-robust safety invariant (γ-RSI) sets. Moreover, the relation between the proposed data-driven approach and the condition of persistency of excitation is discussed, explaining the difficulties in finding a suitable solution when the

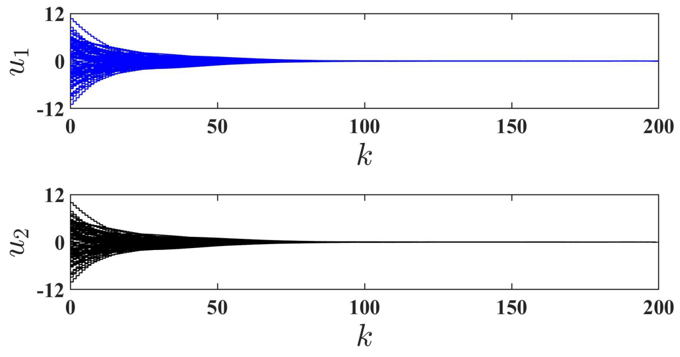


Figure 6.12: Input sequences for the 3DF helicopter example.

collected data do not fulfill such a condition. To show the effectiveness of the proposed results, the proposed results are applied to two case studies.

6.6 Proof of Statements in Chapter 6

Proof of Theorem 6.2.4: First, the statement regarding *if* is proved. If $\exists\kappa \in (0,1]$ such that $(y+\tilde{d})^\top P(y+\tilde{d}) \leq 1$ holds $\forall y \in \mathbb{R}^n$ with $y^\top Py \leq \kappa$, and $\forall\tilde{d}$ with $\tilde{d}^\top\tilde{d} \leq \gamma$, one can let $y = (A+BK)x$ with $x^\top Px \leq 1$ without loss of generality. This immediately implies that (6.2.3) holds $\forall x \in \mathbb{R}^n$ and $\forall w \in \mathbb{R}^n$ with $w^\top w \leq \gamma$.

Next, the statement regarding *only if* is shown by contradiction. Suppose that $\nexists\kappa \in (0,1]$ such that (***Cond.1***) holds. Then, $\exists x \in \mathbb{R}^n$, with $x^\top Px \leq 1$, such that $x^\top(A+BK)^\top P(A+BK)x > 1$. Accordingly, one has

$$\Big((A+BK)x+w\Big)^\top P\Big((A+BK)x+w\Big) > 1,$$

with $w = \mathbf{0}_n$, which results in a contradiction to the fact that (6.2.3) holds for $\forall w \in \Delta(\gamma)$. Therefore, one can see that there exists $\kappa \in (0,1]$ such that (***Cond.1***) holds if (6.2.3) holds $\forall w \in \Delta(\gamma)$, and $\forall x \in \mathbb{R}^n$, with $x^\top Px \leq 1$. In the following discussion, such κ is denoted by κ'. Similarly, assuming that $\nexists\kappa \in (0,1]$ such that (***Cond.2***) holds. This indicates that $\forall\kappa \in (0,1]$, $\exists y \in \mathbb{R}^n$, with $y^\top Py \leq \kappa$, or $\exists\tilde{d} \in \Delta(\gamma)$ such that $(y+\tilde{d})^\top P(y+\tilde{d}) > 1$. Let's consider $\kappa = \kappa'$ and one can let $y = (A+BK)x$ with $x^\top Px \leq 1$ without loss of generality. Then, $\exists x \in \mathbb{R}^n$, with $x^\top Px \leq 1$, or $\exists w \in \Delta(\gamma)$, such that $\Big((A+BK)x+w\Big)^\top P\Big((A+BK)x+w\Big) > 1$, which is contradictory to the fact that (6.2.3) holds $\forall w \in \Delta(\gamma)$, and $\forall x \in \mathbb{R}^n$ with $x^\top Px \leq 1$. Hence, there exists $\kappa \in (0,1]$ s.t. (***Cond.2***) holds, if (6.2.3) holds $\forall w \in \Delta(\gamma)$, and $\forall x \in \mathbb{R}^n$, with $x^\top Px \leq 1$, which completes the proof. ■

Proof of Theorem 6.2.6: First, it is shown that given a $\kappa \in (0,1]$, (***Cond.1***) in Theorem 6.2.4 holds if and only if (6.2.5) holds. By applying S-procedure [35, Section

B.2], (***Cond.1***) in Theorem 6.2.4 holds if and only if there exists $\lambda \in \mathbb{R}_{\geq 0}$ s.t.

$$\begin{bmatrix}(A+BK)^\top P(A+BK) & \mathbf{0} \\ \mathbf{0} & -\kappa\end{bmatrix} \preceq \lambda \begin{bmatrix}P & \mathbf{0} \\ \mathbf{0} & -1\end{bmatrix}, \tag{6.6.1}$$

holds. Accordingly, (6.6.1) holds if and only if $(A+BK)^\top P(A+BK) \preceq \lambda P$ with $\lambda \leq \kappa$. Hence, (6.6.1) holds if and only if (6.2.5) holds according to [211, Theorem 1.12], with $Q = P^{-1}$.

Next, it is proved that (***Cond.2***) in Theorem 6.2.4 holds if and only if (6.2.6) holds. First, considering the geometric properties of ellipsoids $x^\top Px \leq 1$ and $x^\top Px \leq \kappa$, the shortest distance between both ellipsoids is $\sqrt{\lambda_{min}} - \sqrt{\kappa\lambda_{min}}$, with λ_{min} the minimal eigenvalue of P^{-1}. Hence, to ensure (***Cond.2***), one needs to guarantee that

$$\sqrt{\lambda_{min}} - \sqrt{\kappa\lambda_{min}} \geq \sqrt{\gamma}. \tag{6.6.2}$$

Accordingly,

- If $\kappa \neq 1$, (6.6.2) requires that $\lambda_{min} \geq \frac{\gamma}{(1-\sqrt{\kappa})^2}$, which holds if and only if $P^{-1} \succeq \frac{\gamma}{(1-\sqrt{\kappa})^2}\mathbf{I}$;
- If $\kappa = 1$, (6.6.2) holds if and only if $\gamma = 0$, for any $\lambda_{min} \geq 0$. Hence, (6.6.2) holds if and only if $P^{-1} \succeq 0$.

Therefore, (***Cond.2***) in Theorem 6.2.4 holds if and only if (6.2.6) holds.

Finally, it is shown that (6.2.7) and (6.2.8) are respecting the safety set as in (6.1.6), and input constraints as in (6.1.1), respectively. According to [169, Lemma 4.1], (6.2.1) holds for S as in (6.1.6) if and only if (6.2.7) holds. Similarly, considering the RSI-based controller as in (6.2.2), (6.1.1) requires that $b_jKx \leq 1$ should hold for all $j = 1, \ldots, \mathrm{j}$. This can be enforced by (6.2.8) according to [169, Lemma 4.1] and [211, Theorem 1.12], which completes the proof. ■

Proof of Theorem 6.3.2: One can verify that (6.3.2), (6.3.4), and (6.3.5), are the same as (6.2.6), (6.2.7), and (6.2.8), respectively. Therefore, the proof can be completed by showing that (6.3.3), with $\epsilon_i > 0$, $\forall i \in [1, N]$, implies (6.2.5). According to [211, Theorem 1.12], (6.2.5) holds if and only if $(A+BK)Q(A+BK)^\top \preceq \kappa Q$, when considering the Schur complement of κQ of the matrix on the left hand side of (6.2.5), with $K = \bar{K}Q^{-1}$. Therefore, (6.2.5) holds if and only if

$$\begin{bmatrix}\mathbf{I} & A & B\end{bmatrix} \begin{bmatrix}\kappa Q & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -Q & -\bar{Z}^\top \\ \mathbf{0} & -\bar{Z} & -KQK^\top\end{bmatrix} \begin{bmatrix}\mathbf{I} \\ A^\top \\ B^\top\end{bmatrix} \succeq 0, \tag{6.6.3}$$

holds, with $\bar{Z} = KQ$. Next, it is shown that (6.3.3), with $\epsilon_i > 0$, $\forall i \in [1, N]$, implies (6.6.3) holds for any $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ such as $X_{1,N} = AX_{0,N} + BU_{0,N} + D$ holds with $D = [w(0) \ldots w(k) \ldots w(N-1)]$ and $w(k)^\top w(k) \leq \gamma$, $\forall k \in [0, N-1]$, indicating that (6.6.3) holds for the unknown A and B as in (2.3.2). Considering (2.3.2), since $w(k)w(k)^\top \preceq \gamma\mathbf{I}$, $\forall w(k) \in \Delta(\gamma)$, one has

$$\begin{bmatrix}\mathbf{I} & A & B\end{bmatrix} \bar{N}_p \begin{bmatrix}\gamma\mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}\end{bmatrix} \bar{N}_p^\top \begin{bmatrix}\mathbf{I} & A & B\end{bmatrix}^\top \succeq 0, \tag{6.6.4}$$

$\forall p \in [1, N]$, with

$$\bar{N}_p := \begin{bmatrix} \mathbf{I} & X_{1,N}(p) \\ \mathbf{0} & -X_{0,N}(p) \\ \mathbf{0} & -U_{0,N}(p) \end{bmatrix}. \tag{6.6.5}$$

Considering [211, Theorem 1.12], if $\exists \epsilon_i > 0$, $\forall i \in [1, N]$ such that (6.3.3) holds, then one gets

$$\begin{bmatrix} \kappa Q & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -Q & -\bar{Z}^\top \\ \mathbf{0} & -\bar{Z} & 0 \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \bar{Z} \end{bmatrix} Q^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \bar{Z} \end{bmatrix}^\top - \sum_{p=1}^{N} \epsilon_p \bar{N}_p \begin{bmatrix} \gamma \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -1 \end{bmatrix} \bar{N}_p^\top \succeq 0, \tag{6.6.6}$$

with $\bar{N}_p$ as in (6.6.5). According to [26, Lemma 2], (6.6.6) implies that (6.6.3) holds for all (6.6.4) with $p \in [1, N]$, which completes the proof. ■

Proof of Corollary 6.3.4: Consider the system as in (2.3.2), $X_{1,N}$ and $X_{0,N}$, and $U_{0,N}$ as in (6.1.3) to (6.1.5), respectively. Given any disturbance sequence $D := [d_1 \; d_2 \; \dots \; d_N] \in (\Delta(\gamma))^N$, with $(\Delta(\gamma))^N$ being the Cartesian product of N times of the set $\Delta(\gamma)$, it is defined $\tilde{X}_D := (X_{1,N} - D)^\top$, and $Y_{0,N} := \begin{bmatrix} X_{0,N}^\top \; U_{0,N}^\top \end{bmatrix}$. Then, by definition of the set $\mathcal{F}$ as in (6.3.7), one has

$$\mathcal{F} = \bigcup_{D \in (\Delta(\gamma))^N} \mathsf{W}(D)^\top, \tag{6.6.7}$$

with $\mathsf{W}(D) := \{W \in \mathbb{R}^{(n+m)\times n} | \tilde{X}_D = Y_{0,N} W, D \in (\Delta(\gamma))^N\}$.

Firstly, the statement regarding *if* is proved. To this end, one should first show that the set $\mathsf{W}(D)$ is either unbounded or empty, when (6.3.8) holds. Consider the equation $\tilde{X}_D = Y_{0,N} W$, in which $W \in \mathbb{R}^{(n+m)\times n}$ is an unknown matrix to be determined (note that there may not be suitable W, the discussion comes later). According to [177, Section 3.3], for any column $W(i)$, $i \in [1, n]$, if there exists

$${}^{i}\alpha := \begin{bmatrix} {}^{i}\alpha_1 & {}^{i}\alpha_2 & \dots & {}^{i}\alpha_{n+m} \end{bmatrix} \in \mathbb{R}^{n+m}, \tag{6.6.8}$$

such that

$$\tilde{X}_D(i) = \sum_{\mathsf{a}=1}^{n+m} {}^{i}\alpha_{\mathsf{a}} Y_{0,N}(\mathsf{a}) \tag{6.6.9}$$

holds, then one has $W(i) \in \{{}^{i}\alpha + w \mid w \in \mathsf{ker}(Y_{0,N})\}$, with $\mathsf{ker}(Y_{0,N})$ the kernel of $Y_{0,N}$; otherwise, one has $W(i) \in \emptyset$. Therefore, one has

$$\mathsf{W}(D) = \prod_{i=1}^{n} \{{}^{i}\alpha + w \mid w \in \mathsf{ker}(Y_{0,N})\} \neq \emptyset, \tag{6.6.10}$$

when for all $i \in [1, n]$, there exists ${}^{i}\alpha$ as in (6.6.8) such that (6.6.9) holds; and $\mathsf{W}(D) = \emptyset$ otherwise. Note that $\mathsf{ker}(Y_{0,N})$ is an r-dimension subspace of $\mathbb{R}^{n+m}$, with $\mathsf{r} = n + m -$

$\operatorname{rank}(Y_{0,N})$ according to [177, Section 3.5]. If (6.3.8) holds, then one has $\mathsf{r} > 0$. In this case, the set $\{{}^i\alpha + w \mid w \in \mathsf{ker}(Y_{0,N})\}$ is unbounded for any ${}^i\alpha \in \mathbb{R}^{n+m}$ due to the unboundedness of $\mathsf{ker}(Y_{0,N})$. As a result, the set $\mathsf{W}(D)$ is either unbounded or empty when (6.3.8) holds. Moreover, since $X_{0,N}$, $X_{1,N}$, and $U_{0,N}$ are data collected from the system as in (2.3.2), one always has $[A\ B] \in \mathsf{W}(D)$ for some $D \in \Delta(\gamma))^N$, with A and B the unknown matrices in (2.3.2). In other words, there always exists $D \in \Delta(\gamma))^N$ such that $\mathsf{W}(D)$ is not empty (and is therefore unbounded). Hence, it is then straightforward that the right-hand side of (6.6.7) is unbounded, so that statement regarding *if* holds.

Next, the statement regarding *only if* is also proved by showing $\mathcal{F}$ is bounded when

$$\operatorname{rank}\Big(\begin{bmatrix} X_{0,N} \\ U_{0,N} \end{bmatrix}\Big) = n + m. \qquad (6.6.11)$$

When (6.6.11) holds, then $\mathsf{ker}(Y_{0,N})$ only contains the origin. As a result, the set $\{{}^i\alpha + w \mid w \in \mathsf{ker}(Y_{0,N})\}$ is either a singleton set that only contains ${}^i\alpha$, or an empty set, so that the set $\mathsf{W}(D)$ is either a singleton set or an empty empty set, when (6.6.11) holds. Then, the boundedness the right-hand side of (6.6.7) follows by the boundedness of the set $\Delta(\gamma))^N$, which completes the proof. ■

7 Conclusions and Future Works

7.1 Conclusions

In the past decades, many high-performance, but unverified controllers, particularly those developed using artificial intelligence (AI) techniques, are expected to be deployed in modern Cyber-Physical Systems (CPS) for complex control missions. Nevertheless, applying unverified controllers in CPS makes it very challenging to ensure the overall safety of the systems. Meanwhile, it is of vital importance to provide safety guarantees for CPS since any malfunctions in these systems may lead to catastrophic consequences. The main contribution of this thesis is the design of a correct-by-construction controller architecture, namely *Safe-visor architecture*, for sandboxing those unverified controllers deployed in CPS so that a system-level safety guarantee can be provided. In particular, several methodologies based on formal methods have been developed and deployed for designing such an architecture, including abstraction-based approaches, abstraction-free approaches, and data-driven approaches. The results proposed in previous chapters of this thesis are reviewed here.

Chapters 3 and 4 provide abstraction-based methodologies to design the Safe-visor architecture, which enable the use of unverified controllers while enforcing complex logical safety properties expressed by deterministic finite automata (DFA). Concretely, in Chapter 3, new abstraction-based controller synthesis schemes were proposed for discrete-time non-cooperative stochastic games with continuous state and input sets. In this context, a finite abstraction should first be constructed over the original stochastic game by discretizing the original state and input sets. Then, an approximate probabilistic relation should be established between the original game and the finite abstraction to quantify their similarity. Here, I proposed an algorithmic procedure for establishing such a relation over a class of nonlinear stochastic games with slope restrictions on their nonlinearity. Then, new Bellman operators were proposed for synthesizing controllers over the finite abstractions obtained in the previous step with respect to the problems of *robust satisfaction* and *worst-case violation.* These controllers are finally refined back over the original stochastic games by leveraging an interface function associated with the approximate probabilistic relation so that formal probabilistic guarantees for satisfying the desired properties can be provided.

In Chapter 4, the abstraction-based controller synthesis techniques proposed in Chapter 3 were used for designing the safety advisor. On top of the abstraction-based design of the safety advisor, the design of a history-based supervisor was proposed over general Markov decision processes (gMDP) and non-cooperative stochastic games (gDTSGs). More precisely, by leveraging a history state-run of the system at runtime, the history-based supervisor estimates the risk of violating the desired safety properties presuming that the unverified controllers are accepted. Then, given the maximal tolerable probability of violating the desired safety specifications, the history-based supervisor would reject the unverified controller when: 1) the approximate probabilistic relation between the original system and its finite abstraction can not be maintained; 2) the risk for violating the safety specifications is higher than the maximal tolerable violation probability. The abstraction-based methodologies for constructing the Safe-visor architecture were validated through simulation over several case studies and experiments on a physical test-bed for quadrotor helicopters. Both simulation and experimental results showed that the formal probabilistic guarantees for satisfying the desired safety properties were well respected.

While abstraction-based approaches proposed in Chapters 3 and 4 can be applied to nonlinear systems, such methodologies require constructing finite abstractions over the original systems by discretizing the original continuous state and input sets. Such a discretization process often encounters so-called *the curse of dimensionality*, leading to exponential growth in computational complexity with the dimension of the system. In Chapter 5, an abstraction-free construction scheme was proposed for designing the Safe-visor architecture over uncertain linear systems, which does not require building finite abstractions. Specifically, ω-regular properties were deployed to model the desired specifications. Having deterministic Streett automata (DSA) representing the desired ω-regular properties, a hybrid system between the dynamical system and the DSA should first be constructed. Then, the notion of so-called *hybrid controlled invariant (HCI) sets* over the hybrid system was proposed to construct the Safe-visor architecture. Here, set-based approaches over hybrid sets were proposed to compute the maximal HCI sets leveraging a new iterative scheme. To ensure getting valid HCI sets within a finite number of iterations, two alternative iterative schemes were introduced to compute under-approximations of the maximal HCI sets. Additionally, the worst-case time and space complexities of the proposed abstraction-free set-based approaches were analyzed. To show the effectiveness of the proposed set-based approaches, they were applied to several case studies and compared with various existing tools for synthesizing controllers enforcing ω-regular properties.

Note that both abstraction-based (Chapters 3 and 4) and abstraction-free (Chapter 5) approaches require knowledge of the system models. However, in some cases, system models may be difficult to obtain, or the obtained models are too complex to be of any use. In Chapter 6, focusing on safety invariance properties, a direct data-driven approach was proposed for synthesizing safety controllers over uncertain linear systems with unknown system dynamics. These controllers can be used as safety advisors in the Safe-visor architecture. Using the direct data-driven scheme, the controllers can directly be constructed based on a single trajectory collected from the underlying unknown system without an intermediate phase for identifying the system model. Concretely, the notion of γ-robust safety invariant sets was first proposed for synthesizing safety controllers of interest. Then, a semi-definite programming (SDP) problem was provided to compute the γ-robust control invariant set and its associated safety controller directly based on offline collected data. Here, the SDP contains several linear matrix inequalities constraints, and the number of constraints grows linearly with the dimension of the state and input sets so that it can be solved efficiently using existing SDP solvers. Finally, the relations between the conditions of *persistency of excitation* and the proposed direct data-driven approaches were also investigated.

7.2 Future Directions

In the last part of this thesis, I propose several potential directions for extending the technical materials in this thesis. Exploring these directions are beneficial for improving the current construction schemes of the Safe-visor architecture, making it possible to construct such an architecture for large-scale, multi-agent autonomous systems with more complex dynamics. I believe that this architecture will be particularly meaningful in the golden age of AI in which AI techniques have made remarkable achievments in many domains, while safety concerns are still a stumbling block for their applications in safety-critical scenarios.

Construction scheme for systems with more general dynamics. In this thesis, several approaches have been proposed to construct the Safe-visor architecture for 1) nonlinear systems with slope restriction on the nonlinearity, and 2) uncertain linear systems. It is meaningful to further consider systems being in a more general form so that the Safe-visor architecture can be constructed for CPS with more complex system dynamics. Regarding different approaches proposed in this thesis, their corresponding open problems for such extensions are concluded here.

Abstraction-based approach. The Bellman operators proposed in Section 3.3 for synthesizing controllers do not have any restriction on the form of the system dynamics so that they can readily be applied to the general setting of gDTSGs (cf. Remark 3.2.1). Nevertheless, formal safety guarantees are provided on top of an (ϵ,δ)-approximate probabilistic relation between the original game and its finite abstraction. Meanwhile, the results proposed in Section 3.2 for systematically establishing such a relation require the gDTSGs to be in the form of (3.2.1). Therefore, the lack of a systematic framework for establishing (ϵ,δ)-approximate probabilistic relations for more general gDTSGs restricts the application of the current abstraction-based construction scheme to those gDTSGs with more complex dynamics.

Set-based approach. To guarantee the convergence of the set-based iterative scheme proposed in this thesis, one only needs to assume the discrete-time dynamics f to be continuous (cf. proof of Lemma 5.7.1). In this thesis, the reasons for restricting f to be in linear form as in (2.3.2) are twofold:

1. The proposed methodologies for computing the approximations of the maximal HCI sets require the uncertain system to be linear and controllable, as stated in Assumption 5.3.1.

2. For the iterative schemes proposed in Definitions 5.2.9, 5.3.5, and 5.3.9, one needs to compute the *exact* one-step-backward projection for a given continuous set. These computations can be done by leveraging existing toolboxes including

MPT [78] and BENSOLVE [132], if the system of interest is linear, and the set is a P-collection (cf. Definition 5.1.6). For nonlinear systems, one can only compute approximations of these projections with existing results; see [7] and reference herein. In fact, computing the exact one-step-backward projection with respect to arbitrary continuous nonlinear dynamics may not even be feasible in general.

Accordingly, to extend the current set-based approaches to nonlinear systems, one may consider: 1) exploiting the notion of controllability for nonlinear systems (e.g. [34]) to compute the approximations of the maximal HCI set; 2) extending the current iterative schemes by considering the case where only approximations of the one-step-backward projection are available.

Data-driven approach. As a key insight, by leveraging the direct data-driven approaches proposed in this thesis, a common γ-RSI set is computed for the set of linear systems as in Corollary 6.3.4 that could have generated the data in hand. Therefore, the obtained γ-RSI set and its associated controller can be applied to the underlying unknown system. Similarly, to further develop the current results to cope with more general dynamics, one needs to find a proper way to describe the set of nonlinear systems that can explain the collected data. Additionally, one also needs to derive new conditions for computing a common controlled invariant set for nonlinear systems based on data.

It is also worthwhile to deviate from the idea mentioned above and raise a question: *do we have to compute a common controlled invariant set for all systems that explain the given data while we only need a controller that works for one of these systems?* In some recent results, e.g., [190], direct data-driven approaches have been proposed while computing a common invariant set for a set of systems is not required. Although additional assumptions over the system dynamics are the prices, the idea proposed in these works may be helpful for deriving less restrictive computation methodologies for constructing the Safe-visor architecture using data.

Compositional synthesis of Safe-visor architecture. When using the methodologies proposed in this thesis to construct the Safe-visor architecture, the computational cost would rise as the dimension of the systems increases, particularly for those abstraction-based approaches proposed in Chapter 3 and 4. In practice, some CPS, such as smart power grids, are essentially interconnected systems containing several subsystems with independent control inputs in each subsystem. Accordingly, controllers for these CPS are typically implemented in a decentralized or distributed manner (instead of a centralized manner) to reduce the implementation cost and increase the robustness of the whole system with respect to failure in subsystems. In the context of formal synthesis over interconnected systems, several compositional synthesis approaches have been proposed, including abstraction-based approaches leveraging dissipativity theory [208, 122, 117] and small gain theory [180, 210], as well as abstraction-free approaches using set-based approaches [129, 59] and controller barrier functions [10, 9, 86]. Similar ideas may also be applied to the compositional construction of the Safe-visor architecture. Concretely, it would be worthwhile to investigate how to achieve a formal safety guarantee for the whole system by constructing the Safe-visor architecture *locally* for each subsystem instead of building a Safe-visor architecture for the whole system monolithically.

Computational-aware and hardware-aware construction scheme. In this thesis, mathematically rigorous results are proposed to construct the Safe-visor architecture and formal safety guarantees are provided accordingly. In practice, however, computational issues and hardware constraints also need to be handled carefully to maintain these guarantees. Here, several aspects in this context that are worth further investigation are discussed here.

Delay and deadline missing. The thesis implicitly assumes that all the control inputs can be computed and executed in time. However, implementing control software on a digital computer may suffer from *digital implementation delay* [168, Section 3.3] and *deadline missing* [137]. To cope with these timing issues, some recent results [152, 105, 192] also take those timing failures and deadline missing into consideration when synthesizing controllers. These results may be leveraged for improving the current construction scheme of Safe-visor architecture so that formal safety guarantees that are robust to timing failures can be provided.

Hardware constraints. When implementing controllers on a hardware, in particular on an embedded system, there would only be limited memory that can be allocated to the control software. This limitation may restrict the use of abstraction-based Safe-visor architecture proposed in Chapter 3 and 4. Concretely, the history-based supervisors require having access to $\hat{T}$ as in (3.2.14) and $\underline{V}_{*,H-k-1}$ as (3.3.22) (resp. $\bar{V}^*_{H-k-1}$ as in (3.3.9)) at runtime, which are essentially look-up tables obtained offline when synthesizing the safety advisors. In case that these look-up tables are too large for the given hardware, one of the possible solutions could be approximating these tables using DNNs. This idea has been applied to compress a large numerical table that is used in *Airbone Collision Avoidance System X* [91], and a significant size reduction of these tables was achieved. A similar idea could be applied here while new challenges need to be addressed in the context of correct-by-construction synthesis, for example,

1. How to provide a provable error bound for the approximation of these tables?
2. How to systematically improve the DNNs if the approximation error is too large?

Numerical error in computation. The safety guarantees provided in this thesis are valid given the computations of all components in the architecture (for instance, the HCI sets as in Chapter 5 and the γ-RSI sets as in Chapter 6) are mathematically correct. However, their computations require several numerical optimization algorithms, with which numerical errors may occur. Moreover, even if accurate parameters for these components can be obtained, numerical and rounding errors may still appear at either compile- or runtime due to the limitation of the hardware. In fact, these issues have started receiving attention in a few recent works in the communities of formal methods and Cyber-Physical Systems, see, e.g., [197]. To maintain the formal safety guarantees provided by the Safe-visor architecture over physical systems, it is worthwhile to investigate how those possible numerical errors affect the guarantees provided by the

Safe-visor architecture and how to mitigate those negative effects. Moreover, having a deeper understanding about these effects would also be helpful for investigating how to make a proper compromise between the amount of allocated computational resources and the level of safety guarantee that can be achieved.

Bibliography

[1] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini. Approximate model checking of stochastic hybrid systems. *European Journal of Control*, 16(6):624–641, 2010.

[2] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

[3] F. Abdi, C.-Y. Chen, M. Hasan, S. Liu, S. Mohan, and M. Caccamo. Preserving physical safety under cyber attacks. *IEEE Internet of Things Journal*, 6:6285–6300, 2018.

[4] F. Abdi, R. Tabish, M. Rungger, M. Zamani, and M. Caccamo. Application and system-level software fault tolerance through full system restarts. In *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*, pages 197–206. IEEE, 2017.

[5] S. Aberkane and V. Dragan. On a solution to the problem of time-varying zero-sum LQ stochastic difference game: A Riccati equation approach. In *2019 18th European Control Conference (ECC)*, pages 388–393. IEEE, 2019.

[6] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe reinforcement learning via shielding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[7] M. Althoff, G. Frehse, and A. Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:369–395, 2021.

[8] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[9] M. Anand, A. Lavaei, and M. Zamani. Compositional synthesis of control barrier certificates for networks of stochastic systems against ω-regular specifications. *arXiv:2103.02226*, 2021.

[10] M. Anand, A. Lavaei, and M. Zamani. From small-gain theory to compositional construction of barrier certificates for large-scale stochastic systems. *IEEE Transactions on Automatic Control*, 67, 2022.

[11] M. Arcak and P. Kokotovic. Observer-based control of systems with slope-restricted nonlinearities. *IEEE Transactions on Automatic Control*, 46(7):1146–1150, 2001.

[12] L. Asselborn and O. Stursberg. Probabilistic control of uncertain linear systems using stochastic reachability. *IFAC-PapersOnLine*, 48(14):167–173, 2015.

[13] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT press, 2008.

[14] S. Bak, T. T. Johnson, M. Caccamo, and L. Sha. Real-time reachability for verified simplex design. In *2014 IEEE Real-Time Systems Symposium*, pages 138–148. IEEE, 2014.

[15] S. Bak, K. Manamcheri, S. Mitra, and M. Caccamo. Sandboxing controllers for cyber-physical systems. In *2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*, pages 3–12. IEEE, 2011.

[16] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-acobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[17] M. Baotic. Polytopic computations in constrained optimal control. *Automatika, Journal for Control, Measurement, Electronics, Computing and Communications*, 50:119–134, 2009.

[18] N. Basset, M. Kwiatkowska, and C. Wiltsche. Compositional strategy synthesis for stochastic games with multiple objectives. *Information and Computation*, 261:536–587, 2018.

[19] A. Bemporad, K. Fukuda, and F. D. Torrisi. Convexity recognition of the union of polyhedra. *Computational Geometry*, 18(3):141–154, 2001.

[20] J. Berberich, A. Koch, C. W. Scherer, and F. Allgöwer. Robust data-driven state-feedback design. In *2020 American Control Conference (ACC)*, pages 1532–1538, 2020.

[21] J. Berberich, C. W. Scherer, and F. Allgöwer. Combining prior knowledge and data for robust controller design. *IEEE Transactions on Automatic Control*, 2022.

[22] J. M. Bernardo and A. F. M. Smith. *Bayesian theory*, volume 405. John Wiley & Sons, 2009.

[23] D. Bertsekas. Infinite time reachability of state-space regions by using feedback control. *IEEE Transactions on Automatic Control*, 17(5):604–613, 1972.

[24] S. Bharadwaj, R. Bloem, R. Dimitrova, B. Könighofer, and U. Topcu. Synthesis of minimum-cost shields for multi-agent systems. In *2019 American Control Conference (ACC)*, pages 1048–1055, 2019.

[25] A. Bisoffi, C. De Persis, and P. Tesi. Data-based guarantees of set invariance properties. *IFAC-PapersOnLine*, 53(2):3953–3958, 2020.

[26] A. Bisoffi, C. De Persis, and P. Tesi. Trade-offs in learning controllers from noisy data. *Systems & Control Letters*, 154:104985, 2021.

[27] A. Bisoffi, C. De Persis, and P. Tesi. Controller design for robust invariance from noisy data. *IEEE Transactions on Automatic Control*, 2022.

[28] E. Bıyık, D. P. Losey, M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022.

[29] F. Blanchini. Feedback control for linear time-invariant systems with state and control bounds in the presence of disturbances. *IEEE Transactions on automatic control*, 35(11):1231–1234, 1990.

[30] F. Blanchini and S. Miani. *Set-Theoretic Methods in Control.* Birkhäuser, 2015.

[31] R. Bloem, B. Könighofer, R. Könighofer, and C. Wang. Shield synthesis: Runtime enforcement for reactive systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 533–548. Springer Berlin Heidelberg, 2015.

[32] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv:1604.07316*, 2016.

[33] V. S. Borkar. *Probability theory: an advanced course.* Springer Science & Business Media, 2012.

[34] U. Boscain and M. Sigalotti. Introduction to controllability of nonlinear systems. *Contemporary Research in Elliptic PDEs and Related Topics*, pages 203–219, 2019.

[35] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[36] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15. SIAM, 1994.

[37] V. Breschi, C. De Persis, S. Formentin, and P. Tesi. Direct data-driven model-reference control with Lyapunov stability guarantees. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 1456–1461. IEEE, 2021.

[38] M. Breton, A. Alj, and A. Haurie. Sequential stackelberg equilibria in two-person games. *Journal of Optimization Theory and Applications*, 59(1):71–97, 1988.

[39] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[40] M. Cannon, Q. Cheng, B. Kouvaritakis, and S. V. Raković. Stochastic tube MPC with state estimation. *Automatica*, 48(3):536–541, 2012.

[41] V. Cerone, V. Razza, and D. Regruto. MIMO linear systems identification in the presence of bounded noise. In *2016 American Control Conference (ACC)*, pages 919–924. IEEE, 2016.

[42] K. Chatterjee and L. Doyen. Perfect-information stochastic games with generalized mean-payoff objectives. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 247–256, 2016.

[43] K. Chatterjee and R. Ibsen-Jensen. Qualitative analysis of concurrent mean-payoff games. *Information and Computation*, 242:2–24, 2015.

[44] K. Chatterjee, J.-P. Katoen, M. Weininger, and T. Winkler. Stochastic games with lexicographic reachability-safety objectives. In *International Conference on Computer Aided Verification*, pages 398–420. Springer, 2020.

[45] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[46] J. J. Choi, D. Lee, K. Sreenath, C. J. Tomlin, and S. L. Herbert. Robust control barrier-value functions for safety-critical control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6814–6821. IEEE, 2021.

[47] A. Clavière, E. Asselin, C. Garion, and C. Pagetti. Safety verification of neural network controlled systems. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 47–54, 2021.

[48] T. L. Crenshaw, E. Gunter, C. L. Robinson, L. Sha, and P. R. Kumar. The simplex reference model: Limiting fault-propagation due to unreliable components in cyber-physical system architectures. In *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, pages 400–412. IEEE, 2007.

[49] C. De Persis and P. Tesi. Formulas for data-driven control: Stabilization, optimality, and robustness. *IEEE Transactions on Automatic Control*, 65(3):909–924, 2019.

[50] C. De Persis and P. Tesi. Low-complexity learning of linear quadratic regulators from noisy data. *Automatica*, 128:109548, 2021.

[51] J. V. Deshmukh, J. P. Kapinski, T. Yamaguchi, and D. Prokhorov. Learning deep neural network controllers for dynamical systems with safety guarantees. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–7. IEEE, 2019.

[52] 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*, 49(9):2665–2674, 2013.

[53] M. Dutreix and S. Coogan. Specification-guided verification and abstraction refinement of mixed monotone stochastic systems. *IEEE Transactions on Automatic Control*, 66:2975–2990, 2020.

[54] M. Dutreix, J. Huh, and S. Coogan. Abstraction-based synthesis for stochastic systems with omega-regular objectives. *Nonlinear Analysis: Hybrid Systems*, 45:101204, 2022.

[55] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*, pages 121–138. Springer, 2018.

[56] M. E. Dyer. The complexity of vertex enumeration methods. *Mathematics of Operations Research*, 8(3):381–402, 1983.

[57] R. Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.

[58] Y. Y. Elboher, J. Gottschlich, and G. Katz. An abstraction-based framework for neural network verification. In *International Conference on Computer Aided Verification*, pages 43–65. Springer, 2020.

[59] A. Eqtami and A. Girard. A quantitative approach on assume-guarantee contracts for safety of interconnected systems. In *18th European Control Conference (ECC)*, pages 536–541. IEEE, 2019.

[60] J. Esparza, J. Křetínskỳ, J.-F. Raskin, and S. Sickert. From LTL and limit-deterministic Büchi automata to deterministic parity automata. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 426–442. Springer, 2017.

[61] C. Fan. *Formal methods for safe autonomy: Data-driven verification, synthesis, and applications.* PhD thesis, University of Illinois Urbana-Champaign, 2019.

[62] X. Fan and M. Arcak. Observer design for systems with multivariable monotone nonlinearities. *Systems and Control Letters*, 50(4):319–330, 2003.

[63] F. Faruq, D. Parker, B. Laccrda, and N. Hawes. Simultaneous task allocation and planning under uncertainty. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3559–3564. IEEE, 2018.

[64] I. Filippidis, S. Dathathri, S. C. Livingston, N. Ozay, and R. M. Murray. Control design for hybrid systems with tulip: The temporal logic planning toolbox. In *2016 IEEE Conference on Control Applications (CCA)*, pages 1030–1041. IEEE, 2016.

[65] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.

[66] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin. Bridging Hamilton-Jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019.

[67] S. K. S. Frederiksen and P. B. Miltersen. Monomial strategies for concurrent reachability games and other stochastic games. In *International Workshop on Reachability Problems*, pages 122–134. Springer, 2013.

[68] A. Ghaffari. Analytical design and experimental verification of geofencing control for aerial applications. *IEEE/ASME Transactions on Mechatronics*, 26:1106–1117, 2021.

[69] A. Girard and G. J. Pappas. Hierarchical control system design using approximate simulation. *Automatica*, 45(2):566–571, 2009.

[70] J. I. González-Trejo, O. Hernández-Lerma, and L. F. Hoyos-Reyes. Minimax control of discrete-time stochastic systems. *SIAM Journal on Control and Optimization*, 41(5):1626–1659, 2002.

[71] B. Grünbaum. *Convex Polytopes*, volume 221. Springer Science & Business Media, 2003.

[72] M. Guo, C. De Persis, and P. Tesi. Data-driven stabilization of nonlinear polynomial systems with noisy data. *IEEE Transactions on Automatic Control*, 2021.

[73] P. O. Gutman and M. Cwikel. Admissible sets and feedback control for discrete-time linear dynamical systems with bounded controls and states. *IEEE transactions on Automatic Control*, 31(4):373–376, 1986.

[74] S. Haesaert and S. Soudjani. Robust dynamic programming for temporal logic control of stochastic systems. *IEEE Transactions on Automatic Control*, 66:2496–2511, 2021.

[75] S. Haesaert, S. Soudjani, and A. Abate. Temporal logic control of general markov decision processes by approximate policy refinement. *IFAC-PapersOnLine*, 51(16):73–78, 2018. 6th IFAC Conference on Analysis and Design of Hybrid Systems.

[76] S. Haesaert, S. E. Zadeh Soudjani, and A. Abate. Verification of general Markov decision processes by approximate similarity relations and policy refinement. *SIAM Journal on Control and Optimization*, 55(4):2333–2367, 2017.

[77] T. A. Henzinger, L. de Alfaro, and K. Chatterjee. Strategy improvement for concurrent reachability games. In *Third International Conference on the Quantitative Evaluation of Systems*, pages 291–300. IEEE, 2006.

[78] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari. Multi-parametric toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013. `http://control.ee.ethz.ch/~mpt`.

[79] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[80] T. Hou, W. Zhang, and H. Ma. A game-based control design for discrete-time Markov jump systems with multiplicative noise. *IET Control Theory & Applications*, 7(5):773–783, 2013.

[81] Z.-S. Hou and Z. Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.

[82] S. Huh and I. Yang. Safe reinforcement learning for probabilistic reachability and safety specifications: A Lyapunov-based approach. *arXiv:2002.10126*, 2020.

[83] L. Humphrey, B. Könighofer, R. Könighofer, and U. Topcu. Synthesis of admissible shields. In *Hardware and Software: Verification and Testing. HVC 2016. Lecture Notes in Computer Science*, pages 134–151. Springer, 2016.

[84] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178, 2019.

[85] P. Jagtap, S. Soudjani, and M. Zamani. Formal synthesis of stochastic systems via control barrier certificates. *IEEE Transactions on Automatic Control*, 66:3097–3110, 2020.

[86] P. Jagtap, A. Swikir, and M. Zamani. Compositional construction of control barrier functions for interconnected control systems. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*. ACM, 2020.

[87] N. Jahanshahi, A. Lavaei, and M. Zamani. Compositional construction of safety controllers for networks of continuous-space POMDPs. *IEEE Transactions on Control of Network Systems*, 2022.

[88] M. Jankovic. Control barrier functions for constrained control of linear systems with input delay. In *2018 Annual American Control Conference (ACC)*, pages 3316–3321. IEEE, 2018.

[89] J. Jiang and Y. Zhang. A revisit to block and recursive least squares for parameter estimation. *Computers & Electrical Engineering*, 30(5):403–416, 2004.

[90] C. Jones, E. C. Kerrigan, and J. Maciejowski. Equality set projection: A new algorithm for the projection of polytopes in halfspace representation. Technical report, Cambridge University Engineering Dept, 2004.

[91] K. D. Julian, M. J. Kochenderfer, and M. P. Owen. Deep neural network compression for aircraft collision avoidance systems. *Journal of Guidance, Control, and Dynamics*, 42(3):598–608, 2019.

[92] N. Kalra and S. M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016.

[93] M. Kamgarpour, J. Ding, S. Summers, A. Abate, J. Lygeros, and C. Tomlin. Discrete time stochastic hybrid dynamical games: Verification & controller synthesis. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 6122–6127. IEEE, 2011.

[94] M. Kamgarpour, S. Summers, and J. Lygeros. Control design for specifications on stochastic hybrid systems. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 303–312, 2013.

[95] M. Kamgarpour, T. A. Wood, S. Summers, and J. Lygeros. Control synthesis for stochastic systems given automata specifications defined by stochastic sets. *Automatica*, 76:177–182, 2017.

[96] M. I. Karavelas and E. Tzanaki. The maximum number of faces of the Minkowski sum of two convex polytopes. *Discrete and Computational Geometry*, 55(4):748–785, 2016.

[97] M. Kattenbelt, M. Kwiatkowska, G. Norman, and D. Parker. A game-based abstraction -refinement framework for Markov decision processes. *Formal Methods in System Design*, 36(3):246–280, 2010.

[98] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[99] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*, pages 443–452. Springer, 2019.

[100] M. Kazemi and S. Soudjani. Formal policy synthesis for continuous-state systems via reinforcement learning. In *International Conference on Integrated Formal Methods*, pages 3–21. Springer, 2020.

[101] Z. Kenton, A. Filos, O. Evans, and Y. Gal. Generalizing from a few environments in safety-critical reinforcement learning. *arXiv:1907.01475*, 2019.

[102] E. C. Kerrigan. *Robust constraint satisfaction: Invariant sets and predictive control.* PhD thesis, University of Cambridge, 2001.

[103] L. Khachiyan. Complexity of polytope volume computation. In *New trends in discrete and computational geometry*, pages 91–101. Springer, 1993.

[104] M. Khaled and M. Zamani. Omegathreads: Symbolic controller design for ω-regular objectives. In *Proceedings of the 24th ACM International Conference on Hybrid Systems: Computation and Control (HSCC'21). ACM*, pages 1–7, 2021.

[105] M. Khayatian, M. Mehrabian, E. Andert, R. Grimsley, K. Liang, Y. Hu, I. McCormack, C. Joe-Wong, J. Aldrich, B. Iannucci, et al. Plan b: Design methodology for cyber-physical systems robust to timing failures. *ACM Transactions on Cyber-Physical Systems (TCPS)*, 6(3):1–39, 2022.

[106] P. Koopman and F. Fratrik. How many operational design domains, objects, and events? *Safeai@ aaai*, 4, 2019.

[107] O. Kupferman and M. Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

[108] A. Kurzhanskiy. Ellipsoidal Toolbox (ET) , 2021. MATLAB Central File Exchange. Retrieved October, 2021.

[109] M. Kwiatkowska, G. Norman, and D. Parker. Verification and control of turn-based probabilistic real-time games. In *The Art of Modelling Computational Systems: A Journey from Logic and Concurrency to Security and Privacy*, pages 379–396. Springer, 2019.

[110] M. Z. Kwiatkowska. Model checking and strategy synthesis for stochastic games: From theory to practice (invited talk). In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Schloss Dagstuhl, 2016.

[111] M. Z. Kwiatkowska. Safety verification for deep neural networks with provable guarantees (invited paper). In *30th International Conference on Concurrency Theory (CONCUR 2019)*, volume 140, pages 1–5, Dagstuhl, Germany, 2019.

[112] R. B. Larsen, A. Carron, and M. N. Zeilinger. Safe learning for distributed systems with bounded uncertainties. *IFAC-PapersOnLine*, 50(1):2536–2542, 2017.

[113] M. Lauricella. *Set Membership identification and filtering of linear systems with guaranteed accuracy.* PhD thesis, The Polytechnic University of Milan, 2020.

[114] A. Lavaei, M. Khaled, S. Soudjani, and M. Zamani. AMYTISS: Parallelized automated controller synthesis for large-scale stochastic systems. In *Computer Aided Verification*, pages 461–474. Springer, 2020.

[115] A. Lavaei, F. Somenzi, S. Soudjani, A. Trivedi, and M. Zamani. Formal controller synthesis for continuous-space MDPs via model-free reinforcement learning. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems*, pages 98–107. IEEE, 2020.

[116] A. Lavaei, S. Soudjani, A. Abate, and M. Zamani. Automated verification and synthesis of stochastic hybrid systems: A survey. *Automatica*, 146:110617, 2022.

[117] A. Lavaei, S. Soudjani, and M. Zamani. From dissipativity theory to compositional construction of finite markov decision processes. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, pages 21–30. ACM, 2018.

[118] A. Lavaei, S. Soudjani, and M. Zamani. Compositional construction of infinite abstractions for networks of stochastic control systems. *Automatica*, 107:125–137, 2019.

[119] A. Lavaei, S. Soudjani, and M. Zamani. Compositional abstraction of large-scale stochastic systems: A relaxed dissipativity approach. *Nonlinear Analysis: Hybrid Systems*, 36, 2020.

[120] A. Lavaei, S. Soudjani, and M. Zamani. Compositional (in)finite abstractions for large-scale interconnected stochastic systems. *IEEE Transactions on Automatic Control*, 65(12):5280–5295, 2020.

[121] A. Lavaei, S. Soudjani, and M. Zamani. Compositional abstraction-based synthesis of general MDPs via approximate probabilistic relations. *Nonlinear Analysis: Hybrid Systems*, 39, June 2021.

[122] A. Lavaei and M. Zamani. Compositional construction of finite mdps for large-scale stochastic switched systems: A dissipativity approach. *IFAC-PapersOnLine*, 52(3):31–36, 2019.

[123] X. Li and C. Belta. Temporal logic guided safe reinforcement learning using control barrier functions. *arXiv:1903.09885v1*, Mar. 2019.

[124] Y. Li and J. Liu. Invariance control synthesis for switched nonlinear systems: An interval analysis approach. *IEEE Transactions on Automatic Control*, 63(7):2206–2211, 2017.

[125] Y. Li and J. Liu. ROCS: A robustly complete control synthesis tool for nonlinear dynamical systems. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, pages 130–135, 2018.

[126] Y. Li and J. Liu. Robustly complete synthesis of memoryless controllers for nonlinear systems with reach-and-stay specifications. *IEEE Transactions on Automatic Control*, 66:1199–1206, 2020.

[127] Y. Li, Z. Sun, and J. Liu. A specification-guided framework for temporal logic control of nonlinear systems. *IEEE Transactions on Automatic Control*, pages 1–1, 2022.

[128] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations(Poster)*, 2016.

[129] S. Liu and M. Zamani. Compositional synthesis of almost maximally permissible safety controllers. In *American Control Conference (ACC)*, pages 1678–1683. IEEE, 2019.

[130] L. Ljung. *System Identification: Theory for the User*. Prentice Hall information and system sciences series. Prentice Hall PTR, 1999.

[131] J. Lofberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pages 284–289. IEEE, 2004.

[132] A. Löhne and B. Weißing. Equivalence between polyhedral projection, multiple objective linear programming and vector linear programming. *Mathematical Methods of Operations Research*, 84(2):411–426, 2016.

[133] X. Ma, K. Driggs-Campbell, and M. J. Kochenderfer. Improved robustness and safety for autonomous vehicle control with adversarial reinforcement learning. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1665–1671. IEEE, 2018.

[134] N. Maslej, L. Fattorini, E. Brynjolfsson, J. Etchemendy, K. Ligett, T. Lyons, J. Manyika, H. Ngo, J. C. Niebles, V. Parli, Y. Shoham, R. Wald, J. Clark, and R. Perrault. *The AI Index 2023 Annual Report*. AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, Stanford, CA, 2023.

[135] N. Matni, A. Proutiere, A. Rantzer, and S. Tu. From self-tuning regulators to reinforcement learning and back again. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 3724–3740. IEEE, 2019.

[136] N. Matni and S. Tu. A tutorial on concentration bounds for system identification. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 3741–3749. IEEE, 2019.

[137] D. Maxim, L. Cucu-Grosjean, and R. I. Davis. *Probabilistic Analysis*, pages 1–23. Springer Singapore, Singapore, 2019.

[138] P.-J. Meyer and D. V. Dimarogonas. Compositional abstraction refinement for control synthesis. *Nonlinear Analysis: Hybrid Systems*, 27:437–451, 2018.

[139] P. J. Meyer, S. Sickert, and M. Luttenberger. Strix: Explicit reactive synthesis strikes back! In *International Conference on Computer Aided Verification*, pages 578–586. Springer, 2018.

[140] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.

[141] I. M. Mitchell and J. A. Templeton. A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems. In *International workshop on hybrid systems: computation and control*, pages 480–494. Springer, 2005.

[142] J. Moon and T. Başar. Discrete-time stochastic stackelberg dynamic games with a large number of followers. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3578–3583. IEEE, 2016.

[143] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.3.6*, 2019.

[144] H. Mukaidani and H. Xu. Infinite horizon linear-quadratic Stackelberg games for discrete-time stochastic systems. *Automatica*, 76:301–308, 2017.

[145] S. Muntwiler, K. P. Wabersich, A. Carron, and M. N. Zeilinger. Distributed model predictive safety certification for learning-based control. *IFAC-PapersOnLine*, 53(2):5258–5265, 2020.

[146] National Transportation Safety Board. Preliminary report highway-wy18mh010, 2018.

[147] A. Nejati, S. Soudjani, and M. Zamani. Compositional construction of control barrier certificates for large-scale stochastic switched systems. *IEEE Control Systems Letters*, 4(4):845–850, 2020.

[148] A. Nejati and M. Zamani. Compositional construction of finite MDPs for continuous-time stochastic systems: A dissipativity approach. In *Proceedings of the 21st IFAC World Congress, to appear*, 2020.

[149] B. Nortmann and T. Mylvaganam. Data-driven control of linear time-varying systems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3939–3944. IEEE, 2020.

[150] K. Ogata et al. *Discrete-time control systems*, volume 2. Prentice Hall Englewood Cliffs, NJ, 1995.

[151] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[152] P. Pazzaglia, C. Mandrioli, M. Maggio, and A. Cervin. Dmac: Deadline-miss-aware control. In *31st Euromicro Conference on Real-Time Systems (ECRTS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[153] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 179–190. ACM, 1989.

[154] G. Pola, A. Girard, and P. Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[155] G. Pola, P. Pepe, and M. D. Di Benedetto. Decentralized supervisory control of networks of nonlinear control systems. *IEEE Transactions on Automatic Control*, 63(9):2803–2817, 2017.

[156] P. Prabhakar and Z. Rahimi Afzal. Abstraction based output range analysis for neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[157] Quanser Inc. 3 DOF Helicopter. Accessed: Oct 2021.

[158] S. V. Rakovic, P. Grieder, M. Kvasnica, D. Q. Mayne, and M. Morari. Computation of invariant sets for piecewise affine discrete time systems subject to bounded disturbances. In *2004 43rd IEEE Conference on Decision and Control (CDC)*, volume 2, pages 1418–1423. IEEE, 2004.

[159] S. V. Rakovic, E. C. Kerrigan, D. Q. Mayne, and J. Lygeros. Reachability analysis of discrete-time systems with disturbances. *IEEE Transactions on Automatic Control*, 51(4):546–561, 2006.

[160] C. Reis, A. Barth, and C. Pizano. Browser security: lessons from google chrome. *Communications of the ACM*, 52(8):45–49, 2009.

[161] G. Reissig, A. Weber, and M. Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62(4):1781–1796, 2017.

[162] U. Rieder. *Non-Cooperative Dynamic Games with General Utility Functions*, pages 161–174. Springer Netherlands, 1991.

[163] M. Rotulo, C. De Persis, and P. Tesi. Online learning of data-driven controllers for unknown switched linear systems. *Automatica*, 145:110519, 2022.

[164] M. Rungger, M. Mazo, and P. Tabuada. Specification-guided controller synthesis for linear systems and safe linear-time temporal logic. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 333–342. ACM, 2013.

[165] M. Rungger and P. Tabuada. Computing robust controlled invariant sets of linear systems. *IEEE Transactions on Automatic Control*, 62(7):3665–3670, 2017.

[166] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1525–1532. IEEE, 2014.

[167] W. Schilders. *Introduction to Model Order Reduction*, pages 3–32. Springer, 2008.

[168] D. Seto and L. Sha. A case study on analytical analysis of the inverted pendulum real-time control system. Technical report, Carnegie-Mellon Univ. Pittsburgh PA Software Engineering Inst., 1999.

[169] D. Seto and L. Sha. An engineering method for safety region development. Technical report, Carnegie-Mellon Univ. Pittsburgh PA Software Engineering Inst., 1999.

[170] L. Sha. Using simplicity to control complexity. *IEEE Software*, pages 20–28, 2001.

[171] S. E. Shreve. *Stochastic optimal control: The discrete time case.* Academic Press, 1978.

[172] S. Shyamsundar, T. Mannucci, and E.-J. van Kampen. Reinforcement learning based algorithm with safety handling and risk perception. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE, 2016.

[173] M. Simchowitz, H. Mania, S. Tu, M. I. Jordan, and B. Recht. Learning without mixing: Towards a sharp analysis of linear system identification. In *Conference On Learning Theory*, pages 439–473. PMLR, 2018.

[174] S. E. Z. Soudjani. *Formal Abstractions for Automated Verification and Synthesis of Stochastic Systems.* PhD thesis, Delft Center for Systems and Control, TU Delft, 2014.

[175] S. E. Z. Soudjani, A. Abate, and R. Majumdar. Dynamic bayesian networks as formal abstractions of structured stochastic processes. In *26th International Conference on Concurrency Theory (CONCUR 2015)*, volume 42 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 169–183, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[176] M. Srinivasan, S. Coogan, and M. Egerstedt. Control of multi-agent systems with finite time control barrier certificates and temporal logic. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1991–1996. IEEE, 2018.

[177] G. Strang. *Introduction to Linear Algebra.* Wellesley - Cambridge Press, 2016.

[178] R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and control*, 54(1-2):121–141, 1982.

[179] M. Svoreňová, J. Křetínskỳ, M. Chmelík, K. Chatterjee, I. Černá, and C. Belta. Temporal logic control for stochastic linear systems using abstraction refinement of probabilistic games. *Nonlinear Analysis: Hybrid Systems*, 23:230–253, 2017.

[180] A. Swikir and M. Zamani. Compositional synthesis of finite abstractions for networks of systems: A small-gain approach. *Automatica*, 107:551–561, 2019.

[181] P. Tabuada and G. J. Pappas. Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 51(12):1862–1877, 2006.

[182] B. Thananjeyan, A. Balakrishna, U. Rosolia, F. Li, R. McAllister, J. E. Gonzalez, S. Levine, F. Borrelli, and K. Goldberg. Safety augmented value estimation from demonstrations (SAVED): Safe deep model-based RL for sparse cost robotic tasks. *IEEE Robotics and Automation Letters*, 5(2):3612–3619, 2020.

[183] W. Thomas. Automata on infinite objects. In *Formal Models and Semantics*, pages 133–191. Elsevier, 1990.

[184] I. Tkachev, A. Mereacre, J.-P. Katoen, and A. Abate. Quantitative automata-based controller synthesis for non-autonomous stochastic hybrid systems. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 293–302. ACM, 2013.

[185] K.-C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3 —a MATLAB software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4):545–581, 1999.

[186] H.-D. Tran, X. Yang, D. Manzanas Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson. NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *International Conference on Computer Aided Verification*, pages 3–17. Springer, 2020.

[187] B. C. van Huijgevoort and S. Haesaert. Similarity quantification for linear stochastic systems as a set-theoretic control problem. *arXiv:2007.09052*, 2020.

[188] M. Van Kreveld, O. Schwarzkopf, M. de Berg, and M. Overmars. *Computational geometry algorithms and applications.* Springer, 3 edition, 2008.

[189] H. J. van Waarde, M. K. Camlibel, and M. Mesbahi. From noisy data to feedback controllers: non-conservative design via a matrix S-lemma. *IEEE Transactions on Automatic Control*, 2020.

[190] H. J. van Waarde, M. Kanat Camlibel, and H. L. Trentelman. Data-driven analysis and design beyond common lyapunov functions. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 2783–2788, 2022.

[191] R. Vidal, S. Schaffert, J. Lygeros, and S. Sastry. Controlled invariance of discrete time systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 437–451. Springer, 2000.

[192] N. Vreman, A. Cervin, and M. Maggio. Stability and performance analysis of control systems subject to bursts of deadline misses. In *33rd Euromicro Conference on Real-Time Systems (ECRTS 2021)*, 2021.

[193] K. P. Wabersich and M. N. Zeilinger. Linear model predictive safety certification for learning-based control. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 7130–7135. IEEE, 2018.

[194] K. P. Wabersich and M. N. Zeilinger. Safe exploration of nonlinear dynamical systems: A predictive safety filter for reinforcement learning. *arXiv:1812.05506*, 2018.

[195] X. Wang, N. Hovakimyan, and L. Sha. L1simplex fault-tolerant control of cyber-physical systems. In *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems*, pages 41–50, 2013.

[196] X. Wang, N. Hovakimyan, and L. Sha. Rsimplex: A robust control architecture for cyber and physical failures. *ACM Transactions on Cyber-Physical Systems*, 2(4):1–26, 2018.

[197] A. Weber, E. Macoveiciuc, and G. Reissig. Abs: A formally correct software tool for space-efficient symbolic synthesis. In *25th ACM International Conference on Hybrid Systems: Computation and Control*, pages 1–10, 2022.

[198] C. Weibel. *Minkowski sums of polytopes: combinatorics and computation.* PhD thesis, EPFL, 2007.

[199] P. Wieland and F. Allgöwer. Constructive safety using control barrier functions. *IFAC Proceedings Volumes*, 40(12):462–467, 2007.

[200] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. M. De Moor. A note on persistency of excitation. *Systems & Control Letters*, 54(4):325–329, 2005.

[201] C. Wiltsche. *Assume-guarantee strategy synthesis for stochastic games.* PhD thesis, University of Oxford, 2015.

[202] E. M. Wolff, U. Topcu, and R. M. Murray. Automaton-guided controller synthesis for nonlinear systems with temporal logic. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4332–4339. IEEE, 2013.

[203] W. Xiang, P. Musau, A. A. Wild, D. M. Lopez, N. Hamilton, X. Yang, J. Rosenfeld, and T. T. Johnson. Verification for machine learning, autonomy, and neural networks survey. *arXiv:1810.01989v1*, 2018.

[204] Y. Yang, Y. Yin, W. He, K. G. Vamvoudakis, H. Modares, and D. C. Wunsch. Safety-aware reinforcement learning framework with an actor-critic-barrier structure. In *2019 American Control Conference (ACC)*, pages 2352–2358. IEEE, 2019.

[205] J. Yao, X. Liu, G. Zhu, and L. Sha. Netsimplex: Controller fault tolerance architecture in networked control systems. *IEEE Transactions on Industrial Informatics*, 9(1):346–356, 2013.

[206] H. Yu and D. Monniaux. An efficient parametric linear programming solver and application to polyhedral projection. In *International Static Analysis Symposium*, pages 203–224. Springer, 2019.

[207] L. Zaccarian. DC motors: dynamic model and control techniques, 2012.

[208] M. Zamani and M. Arcak. Compositional abstraction for networks of control systems: A dissipativity approach. *IEEE Transactions on Control of Network Systems*, 5(3):1003–1015, 2018.

[209] M. Zamani, G. Pola, M. Mazo, and P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, 57(7):1804–1809, 2011.

[210] M. Zamani, M. Rungger, and P. M. Esfahani. Approximations of stochastic hybrid systems: A compositional approach. *IEEE Transactions on Automatic Control*, 62(6):2838–2853, 2017.

[211] F. Zhang. *The Schur complement and its applications*, volume 4. Springer Science & Business Media, 2006.

[212] B. Zhong, H. Cao, M. Zamani, and M. Caccamo. Towards safe ai: Sandboxing dnns-based controllers in stochastic games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 15340–15349, 2023.

[213] B. Zhong, A. Lavaei, H. Cao, M. Zamani, and M. Caccamo. Safe-visor architecture for sandboxing (AI-based) unverified controllers in stochastic cyber–physical systems. *Nonlinear Analysis: Hybrid Systems*, 43:101110, 2021.

[214] B. Zhong, A. Lavaei, M. Zamani, and M. Caccamo. Automata-based controller synthesis for stochastic systems: A game framework via approximate probabilistic relations. *Automatica*, 147:110696, 2023.

[215] B. Zhong, M. Zamani, and M. Caccamo. A set-based approach for synthesizing controllers enforcing ω-regular properties over uncertain linear control systems. In *Proceedings of the American Control Conference*, pages 1575–1581. IEEE, 2022.

[216] B. Zhong, M. Zamani, and M. Caccamo. Synthesizing safety controllers for uncertain linear systems: A direct data-driven approach. In *2022 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1278–1284. IEEE, 2022.

[217] B. Zhong, M. Zamani, and M. Caccamo. Formal synthesis of controllers for uncertain linear systems against ω-regular properties: A set-based approach. *IEEE Transactions on Automatic Control*, 2023.

[218] Q. Zhu and T. Basar. Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: games-in-games principle for optimal cross-layer resilient control systems. *IEEE Control Systems Magazine*, 35(1):46–65, 2015.

[219] M. H. Zibaeenejad and J. Liu. Auditor product and controller synthesis for nondeterministic transition systems with practical LTL specifications. *IEEE Transactions on Automatic Control*, 65(10):4281–4287, 2019.