Data-Driven Approaches for Formal Synthesis of Cyber-Physical Systems

Milad Kazemi Mehrabadi

School of Computing
Newcastle University

This dissertation is submitted for the degree of
Doctor of Philosophy

March 2023

I would like to dedicate this thesis to my loving parents.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others except as specified in the text and Acknowledgements. This dissertation contains fewer than 80000 words, including appendices, bibliography, footnotes, tables and equations, and has fewer than 100 figures.

Milad Kazemi Mehrabadi
March 2023

Acknowledgements

First and foremost, I would like to thank my supervisor Dr Sadegh Soudjani. Having the opportunity to work with him has been an exceptional intellectual and personal experience for me. I enjoy his wealth of ideas, clarity of thought, enthusiasm, endless energy, and constant encouragement. I feel very fortunate to have had him as an advisor and a teacher.

It has been a pleasure to be a part of the AMBER group at Newcastle University, where I have made many great friends. My deepest thanks go to my former and current office-mates at USB 6.030. I would like to especially acknowledge Ben Wooding for his collaboration and friendship.

In the course of my doctoral studies, I was fortunate enough to collaborate on a variety of research projects with several different people. For all the assistance and support they provided to me during this period, I would like to express my gratitude to Ashutosh, Fabio, Majid and Mateo from the University of Colorado Boulder, Alvaro from DARPA, Mahmoud and Rupak from the Max Planck Institute for Software Systems, Abolfazl from Newcastle University, and Vahid from Nottingham Trent University.

I am grateful to all my friends who have supported me through the years and have made my time at Newcastle so enjoyable. In particular, I would like to thank Armin, Desiree, Roberto, Roberta, Oliver, Paulius, Mazyounah, Felix, Jecel, Emnani, Artur, Dasha, Ivan, Ziqi, Alex, Marco, Mohammad Hossein, Mojtaba, Mostafa, and Radin. My special thanks go to my close friend Luca, with whom I share many unforgettable memories.

My deepest gratitude and love, of course, belong to my parents Alireza and Maryam, and my sisters Mobina and Mahdis for their unconditional love and support all through my life. To them, I owe all that I am and all that I have ever accomplished, and it is to them that I dedicate this thesis.

Abstract

The traditional view in control theory connects sensing, actuation, and computation in a feedback loop to provide stability, performance, and robustness. In recent years, the control community has started looking at controlling systems such as trustworthy autonomous systems and networked systems to satisfy complex requirements. The question has then changed to address dynamic, interconnection, and computing in a unified and scalable framework against high-level logical requirements including safety. Such a comprehensive framework is needed especially for the design of safety-critical systems.

The requirements on the system's behaviour can generally be expressed as temporal logic specifications. Such specifications express formally how the system should behave as time passes. Examples of logical specifications include: always staying in a safe region, reach a destination within a certain time, and visit a region infinitely often.

A prominent approach for formal control synthesis against logical specifications is to use *abstraction-based methods*. Due to the complex nature of the system dynamics that evolve over continuous or hybrid spaces, an abstract model is first constructed that approximates the dynamical system's behaviour with a simple finite model. Analysing the finite-state model is more accessible than the continuous-state model, and efficient computational methods are available from Computer Science literature using compact data structures. There exists a gap between the dynamical behaviour of the original model and the abstraction that is considered to guarantee the satisfaction of the specification on the original model. This gap is generally addressed by ensuring that the abstract model over-approximates the behaviour of the original model or by making the specification more conservative.

This thesis aims to *push the boundaries of abstraction-based methods by making them applicable to large-scale systems that operate in an uncertain environment using data-driven compositional learning approaches*. Formal abstraction-based synthesis schemes rely on a precise mathematical model of the system to build a finite state abstract model. Their usage is limited to small-scale models because the finite abstract model is generally constructed by state space discretisation. This thesis will address the above limitations by making the following contributions.

The first contribution of this thesis is to make abstraction-based schemes applicable when the system dynamics is unknown. We study the formal synthesis of controllers for continuous-space systems with unknown dynamics to satisfy requirements expressed as linear temporal logic (LTL) formulas. We propose a data-driven approach that computes the growth bound of the system using a finite number of trajectories. The growth bound gives the distance between the trajectories started from different initial states. The growth bound and the sampled trajectories are used to construct the abstraction and synthesise a controller. Our approach casts the computation of the growth bound as a robust convex optimisation program (RCP). Since the unknown dynamics appear in the optimisation, we formulate a scenario convex program (SCP) corresponding to the RCP using a finite number of sampled trajectories. We establish a sample complexity result that gives a lower bound for the number of sampled trajectories to guarantee the correctness of the growth bound computed from the SCP with a given confidence.

The second contribution of this thesis is to address the scalability of abstraction-based methods for systems that are influenced by random uncertainties. We design model-free reinforcement methods to satisfy temporal properties on unknown stochastic systems with continuous state spaces. We show how reinforcement learning (RL) can be applied for computing policies that are finite-memory and deterministic, using only the paths of the stochastic process. We address properties expressed in LTL and give a path-dependent reward function maximised via the RL algorithm. We develop the required assumptions and theories for the learned policy to converge to the optimal policy in the continuous state space.

The third contribution of this thesis is to provide a formal compositional synthesis approach designed for large-scale interconnected systems. We introduce a novel RL scheme to synthesise policies for *networks* of continuous-space stochastic control systems with unknown dynamics. The proposed *compositional* framework applies model-free *two-player* RL in an assume-guarantee fashion and *compositionally* compute strategies for continuous-space interconnected systems without explicitly constructing their finite-state abstractions. This approach gives a guaranteed lower bound for the probability of property satisfaction by the interconnected system based on those of individual controllers over subsystems.

As our last contribution, we address the use of average-reward RL for controller synthesis with formal convergence guarantees. Previous approaches rely on using discounted RL with formal guarantees that hold only when the discounting factor convergences to one. Discounted RL prioritises the short-term behaviour of the system over the long-term performance. To satisfy an LTL property, we need to choose the discounting factor close to one, leading to the instability of the learning algorithms. An alternative to discounted RL is to use the average objective without discounting, which inherently focuses on the system's

long-term behaviour. We restrict our attention to specifications corresponding to *absolute liveness* properties (i.e., those that cannot be invalidated by a finite prefix). We propose a translation from absolute liveness properties to an average reward objective for RL. This reduction can be made on the fly without full knowledge of the system, thereby enabling the use of model-free RL algorithms.

The contributions made during the course of this PhD enable us to perform formal control synthesis against high-level logical specifications on larger classes of systems. This is achieved by designing novel data-driven and learning methods with proper formal (convergence or closeness) guarantees.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

Driven by advancements in data analysis, artificial intelligence, and mathematical modelling, Cyber-Physical Systems (CPS) development and analysis are employed across many disciplines, from smart grids to medical devices to robotic systems. CPS integrate computing devices that interact with the physical world through sensors and actuators. The economic and societal potentials of these systems attract significant investments worldwide to develop the technology. The challenge of design and analysis to ensure the reliability of such systems has attracted researchers from academia and industry.

Central to the analysis of CPS is control theory, a well-studied field with many mathematical tools for design and analysis. The design of controllers requires modelling the dynamics of the physical systems. Although we are seeking to design controllers for specific tasks, we expect a high level of confidence in the correct performance of the system because errors can lead to unacceptable consequences such as loss of life. Specifically, in safety-critical systems, safety has a higher priority over other design objectives.

The traditional approach for checking the performance of a system is by extensive testing and validation. However, a more thorough approach involves writing mathematically precise requirements of the desired task and checking whether the system's model meets them. This framework leads to model-based approaches for controller design. The goal of modelling in system analysis is to provide mathematical abstractions to manage design complexity. Abstraction-based approaches use a mathematical abstraction as part of the framework for analysis and synthesis. However, constructing a sufficiently accurate model could be costly and time consuming. Data-driven approaches provide a solution for such cases when a model of the system is not available or is difficult to construct.

As CPS include numerous subsystems, the complex network structure, heterogeneity, and nonlinearity make it increasingly challenging to use traditional optimisation algorithms. This complexity motivates the effective use of artificial intelligence (AI) and machine learning (ML) methods.

Reinforcement learning (RL), as one essential ML paradigm in AI, has been adopted to address some of these issues. RL refers to learning through interaction and experience, which translates to exploration and exploitation. RL is a process through which an agent learns from its own experience. The agent has some states, actions, and rewards, an initial state and a task to accomplish. After it learns through mistakes, it can take an optimal action in each state that steers it toward accomplishing the task. In RL, we have the notion of reward, which makes it different from the traditional ML algorithms, such as supervised learning and unsupervised learning. The notion of reward validates the predicted output of the system. In this way, we can consider the reward as a lens to reaching the goal.

Traditional RL algorithms use an ad-hoc approach for designing the reward function. However, designing a reward function is not a trivial task. In this way, choosing an inappropriate reward structure may lead to finding an incorrect policy. A more thorough approach involves writing mathematically precise requirements of the desired task and translating them into an appropriate reward function.

To check that the RL algorithm works appropriately, the designer must represent the tasks in a mathematically precise manner. We can usually express these tasks as temporal logic formulas covering a range of specification formalisms and associated techniques for formal verification. We can classify these properties into two main groups: safety properties and liveness properties. A safety requirement asserts that “nothing bad ever happens”. On the other hand, a liveness requirement asserts that “something good eventually happens.” Most of the desired tasks for dynamical systems can be represented as a combination of safety and liveness properties.

In model-based approaches, analysis tools like model checkers allow the designer to check that the system accomplishes the task. The same reasoning applies when we intend to use data-driven approaches. The designer needs to represent the tasks in a mathematically precise manner. However, the goal is to provide a formally correct translation from these temporal logic formulas into a reward function. We are looking for a correct translation from a temporal logic formula to a reward function.

This thesis studies whether we can provide a translation from a task to a reward function with correctness guarantees. In this way, throughout this thesis, we examine different RL paradigms and how we can translate a task into a reward machine. The RL algorithms we

explore in this thesis do not require constructing an explicit model of the underlying system, and in this sense, they are all model-free.

1.2 Aims and objectives

This thesis studies formal model-free reinforcement learning algorithms while providing correctness guarantees for the translation of temporal properties into reward functions. The objectives of the research are as follows.

- Study the problem of model-free formal policy synthesis to satisfy a temporal property when we have an underlying stochastic continuous state dynamical system.
- Investigate the problem of model-free formal policy synthesis for temporal properties when we have a continuing task without any resetting and an average reward objective.
- Develop data-driven abstraction-based approaches for temporal properties and also providing a lower bound for the probability of satisfaction and sample size complexity.
- Applying compositional formal RL algorithms and using game-theoretic approaches to improve the scalability of traditional RL algorithms.
- Provide a data-driven approach for efficient construction of finite abstractions of continuous systems.

1.3 Contributions

The contributions of this thesis are as follows:

- We develop a data-driven abstraction-based algorithm for formal control synthesis of continuous systems. In comparison to existing results, our novel approach does not allow violation of the property on any subset of the state space, which leads to a correct lower bound for the probability of satisfaction. This algorithm and its sample complexity are presented in Chapter 3.
- We develop a compositional framework that applies model-free two-player RL in an assume-guarantee fashion. It compositionally computes strategies for continuous-space interconnected systems without explicitly constructing their finite-state abstractions. The algorithm and the associated guaranteed lower bound for the probability of satisfaction for the network are presented in Chapter 4.

- We develop a translation from a temporal task to an RL objective. We investigate reductions without full knowledge of the system and develop the required assumptions and theories for the convergence of the learned policy to the optimal policy in Chapter 5.
- We develop a translation from the temporal properties to an average reward objective for RL. We explore reductions that can be made on the fly without full knowledge of the environment, thereby enabling the use of model-free RL algorithms in Chapter 6.

1.4 Thesis outline

This thesis comprises seven chapters, including this introduction chapter. The remainder of this thesis chapters is organised as follows:

Chapter 2 In this chapter, the necessary background techniques on formal synthesis of cyber-physical systems are provided.

Chapter 3 This chapter studies data-driven formal abstraction-based synthesis schemes to design a controller for continuous-space systems with unknown dynamics. Additionally, it discusses the sample complexity that gives a lower bound for the number of sampled trajectories to guarantee the correctness of the controller with a given confidence. The result of this chapter is based on the work [78], which is under review in the Elsevier Journal on Nonlinear Analysis: Hybrid Systems. This work was the collaboration between Newcastle University and Max Planck Institute for Software Systems.

Chapter 4 This chapter provides formal Reinforcement Learning schemes to synthesise policies for a network of continuous-space stochastic control systems with unknown dynamics. Additionally, it explores compositional game-theoretic model-free RL framework for control synthesis. It also discusses computing a guaranteed lower bound for the probability of property satisfaction by the interconnected system based on those of individual controllers over subsystems. The result of this chapter is based on the work which is under review at Nonlinear Analysis: Hybrid Systems. This work was the collaboration between Newcastle University, ETH, and University of Colorado at Boulder.

Chapter 5 This chapter studies the satisfaction of temporal properties on unknown stochastic processes that have continuous state spaces. It also explores the required assumptions and theories for the convergence of the learned policy to the optimal policy in the continuous state space. Additionally, it discusses sequential learning procedures to guide the RL algorithm toward a policy that converges to an optimal policy under suitable assumptions on the process. The result of this chapter is based on the work [80].

Chapter 6 This chapter provides formal average-reward RL algorithms for control synthesis of continuing tasks. Additionally, it discusses empirically with various benchmarks a

comparison between the proposed method of using average reward RL for continuing tasks defined by omega-regular specifications and competing approaches that leverage discounted RL. The result of this chapter is based on the work [79]. This work was the collaboration between Newcastle University, University of Colorado at Boulder, and Air force research.
Chapter 7 This chapter presents the conclusion of this thesis. we summarise our contributions, limitations of our research and possible future research directions.

1.5 Publications

- Kazemi, Milad, and Sadegh Soudjani. "Formal policy synthesis for continuous-state systems via reinforcement learning." International Conference on Integrated Formal Methods. Springer, Cham, 2020.
- Kazemi, Milad, et al. "Translating omega-regular specifications to average objectives for model-free reinforcement learning." Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems. 2022.
- Kazemi, Milad. "Data-driven Approaches for Formal Synthesis of Dynamical Systems." Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems. 2022.
- Kazemi, Milad, et al. "Smart Charging and Operation of Electric Fleet Vehicles in a Smart City." Cyberphysical Smart Cities Infrastructures: Optimal Operation and Intelligent Decision Making (2022): 61-94.
- Wooding, Ben, et al. "Control and management of active buildings." Active Building Energy Systems. Springer, Cham, 2022. 161-192.
- Lavaei, A., et al. "Compositional Reinforcement Learning for Discrete-Time Stochastic Control Systems.", Nonlinear Analysis: Hybrid Systems.(Under Review)
- Kazemi, Milad, et al. "Data-Driven Abstraction-Based Control Synthesis", Nonlinear Analysis: Hybrid Systems. (Under Review)
- Kazemi, Milad, et al. "Assume-Guarantee Reinforcement Learning", Thirty-Seventh AAAI Conference on Artificial Intelligence. (Under Review)

Chapter 2

Background on formal synthesis of cyber-physical systems

2.1 Introduction

The goal of this chapter is to provide an introduction to the principles of control synthesis algorithms. We approach this subject in the context of the design and analysis of cyber-physical systems. Due to the distinguishing characteristics of cyber-physical systems, these approaches include a diverse set of disciplines such as model-based design, formal methods for specification and verification, control theory, and machine learning. In this chapter, we start by explaining the notion of cyber-physical systems, control synthesis, and formal methods, and then we discuss data-driven approaches. For data-driven approaches, we focus on abstraction-based approaches and reinforcement learning.

2.2 Cyber-physical systems

Ongoing advances in science and engineering and the attempts in linking the computation and physical elements leads to ubiquitous use of cyber-physical systems in many applications such as smart grids, robotics, autonomous automobile systems and medical monitoring devices. In a cyber-physical system, computing devices communicate and interact with sensors and actuators to control physical objects. Across various sectors, from smart buildings to medical devices to automobiles, such systems are becoming more common. In recent years, there has been a surge in developing tools to ensure the reliability of such systems.

Traditionally, system development includes designing and implementing the system, followed by extensive testing and validation to detect bugs. It is more principled to define

precise mathematical requirements for a system before it is developed. Furthermore, for design and analysis, a system model is necessary. Analysis tools are then used to verify whether the system model meets requirements. In comparison to traditional approaches, this methodology will detect design errors early and increase reliability.

Controlling a physical system is done by modelling it with a component called environment. It is possible to influence the evolution of the environment using actions, but it is equally dependent on uncontrollable environmental factors, known as disturbances. The agent responds to commands from the user based on specification and can make its decisions based on observation of the environment provided by the sensors, for example, for designing a policy to move a robot from a region to another region. The sensor is a GPS sensor on the robot that can measure the current position. The agent's task is to design the policy to move toward the target region. The agent can influence the position by adjusting the angle of the steering wheel and acceleration. The environment, in this case, needs to capture how the robot's position changes as a function of the steering angle and acceleration and wind, which is the uncontrolled disturbance.

It is worth noting that the vocabulary of reinforcement learning and control theory can be used synonymously. In this way we can call the environment as plant, the action as control input, and the agent as controller.

Every cyber-physical system has a physical underlying component. One way to model this dynamical system is to represent the system as a set of differential equations:

$$\dot{x} = F(x,u,w) \tag{2.2.1}$$

In Eq. (2.2.1), the differential equations represents the evolution of the dynamical system with time derivatives equal to F. $x \in \mathbb{R}^{n_x}$ represents the state of the system, $u \in \mathbb{R}^{n_u}$ represents the control input, and $w \in \mathbb{R}^{n_w}$ represents bounded disturbance input with known bounded interval. This model has a continuous uncountable state space, and any computational method needs to translate the model into a finite abstract model.

In CPS analysis, control theory is central. Control theory is a well-studied field of mathematics that gives many tools for design and analysis. Controller designs require modelling of the dynamics of physical systems. Modelling in system design helps manage the complexity of a system by creating mathematical abstractions. There are different paradigms in modelling dynamical systems. A model tries to capture the future behaviour of the dynamical system. In deterministic models, the parameter and initial values determine the model's future behaviour. However, in probabilistic (or stochastic) models, randomness plays a role in the system's future behaviour.

To verify whether the design of a system (or implementation of one) works as intended, one needs to first mathematically represents the precise requirements. We call these requirements as specifications. The designer can then check whether the system meets the requirements using analysis tools. Additionally, the designer can synthesise a controller under which the system satisfies the requirements. Given our access to a precise model of the system, we can use model-based approaches.

We divide the approaches for control synthesis in two important categories: model-based and data-driven approaches. In the following we start with model-based approaches. In model-based approaches, analysis tools like model checkers allow the designer to check that the system accomplishes the task. The fact that we are aiming to construct controllers for specific tasks does not absolve us from the responsibility of ensuring that the system performs as expected since errors can have catastrophic consequences, including loss of life. The designer needs to represent the tasks in a mathematically precise manner.

2.3 Model-based synthesis approaches

In this section, we discuss model-based approaches for formal synthesis of systems. There is an extensive body of literature on *model-based* formal synthesis for both deterministic and probabilistic systems. We refer the reader to the books [8, 161, 13] and seminal papers [49, 1]. In terms of specifications, computation tree logic (CTL) and linear temporal logic (LTL) are some of the most common temporal logics used to express the correctness of computer programs and digital circuits modelled as finite-state transition systems. Formal analysis or model checking, the process of analysing such models against a temporal logic formula, has received much attention over the past few decades. Several efficient algorithms and software tools are available for this task.

Automata-based and optimisation-based methods represent the major current approaches to combine optimality and correctness [12]. Automata-based approaches are based on the observation that an LTL formula can be translated into an automaton such that the language accepted by the automaton is the language satisfying the formula. There are different ways these automata can be represented, depending on the desired expressivity of the specification language. A control problem can then be reduced to a game. This game is played on the product between a system and an automaton generated from the specification. The winning condition, which ensures correctness, is the acceptance condition of the automaton. The main limitation of automata-based methods stems from their computational complexity, which leads to an exponential blowup of the partition-based abstraction for infinite-horizon systems.

In addition, due to the partition-based abstraction, such methods are generally conservative for infinite systems.

The basis of optimisation-based methods is temporal logic over finite-horizon signals. These finite-horizon logics not only indicate whether signals satisfy or violate a formula but also support quantitative semantics, which assesses the robustness of satisfactions. Moreover, it can be rewritten as the feasibility part of an optimisation problem. In this way we solve an optimisation problem on top of satisfying the property. It is possible to combine the cost of penalising deviations from the desired trajectory with the satisfaction's robustness to reach the optimisation's overall objective. Therefore, these types of methods benefit from combining correctness and optimality, robustness, and scalability.

2.3.1 Abstraction-based control synthesis

Consider the problem of finding a policy for a system to satisfy a temporal property expressed as an LTL formula. LTL specifications need to be translated into automata with appropriate acceptance conditions. A product automaton is created by synchronising a finite approximated version of the system and the automaton. The satisfying run of this product system visits a set of accepting states infinitely often. The existence of an optimal run with a prefix-suffix structure implies that it is sufficient to search for runs with a finite transient state followed by a periodic steady state. To find an optimal cycle containing an accepting state, we use a polynomial-time graph algorithm to find solutions to shortest-path problems [134].

Although the proposed approaches in [1, 71] have been promising, they rely on knowing a precise system model; thus, they cannot be applied when the model is not known. Accurate models for many physical systems are either unavailable or too complex to be of practical use; hence, one cannot employ model-based techniques to analyse these systems. Although there are results in the literature to handle various analysis and synthesis problems by learning approximate models utilising identification approaches [25, 69], acquiring an accurate model for complex systems is always very challenging, time-consuming, and expensive. This critical challenge motivated us to bypass the system identification phase and employ a data-driven approach for controller synthesis of complex stochastic systems with (fully or partially) unknown dynamics.

An approach to handle models of systems over continuous uncountable spaces is to approximate them with finite-state models conservatively. These finite models are then called *abstractions* of the original *concrete* model. In abstraction-based control synthesis, a controller for a continuous-state system is synthesised through a three-step process [108]. They abstract the continuous-state system into a finite transition system, solving the controller for the abstraction and refining the controller for the continuous system. We describe this

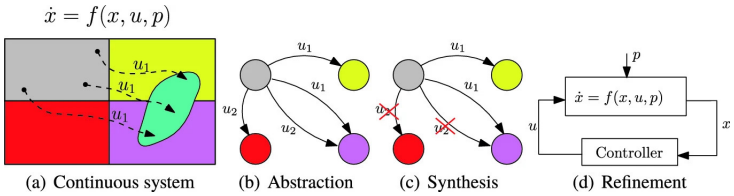


Fig. 2.1 Abstraction-based synthesis for control of a continuous system [108].

procedure next. The schematic of the abstraction-based control synthesis approach is shown in Fig. 2.1.

Abstraction

The abstraction-based control synthesis starts with constructing an abstraction of the dynamical system. We abstract the continuous system in Eq. (2.2.1) into a finite transition system defined by the triple (X,U,δ), where X is a set of symbols (also called states), U is a finite set of controls, and $\delta : X \times U \to 2^X$ is a non-deterministic transition relation that describes the set of possible next states for each input-state combinations in $X \times U$. To construct a finite transition system (X,U,δ), all continuous sets and behaviours of Eq. (2.2.1) need to be discretised. We begin by uniformly partitioning the continuous state space I^{n_x} into a finite set of smaller intervals. The partition parameter is represented as $\alpha_x \in \mathbb{N}$ and is the number of elements per dimension for the abstraction, meaning that the whole partition X contains $\alpha_x^{n_x}$ intervals, each denoting a discrete state of the abstraction. Our next step is to create a discretisation of the control set U. The computational complexity of the abstraction step depends on the number of discrete elements. While increasing the number of discrete elements is favourable for the control synthesis step, we want to avoid computational complexity in the abstraction step. It is not necessary to discretise the disturbance set w, which is included directly in the reachability analysis as discussed next.

Finally, we discretise the continuous behaviours of Eq. (2.2.1) to the finite and nondeterministic transition relation $\delta : X \times U \to 2^X$. Given a sampling time $\tau > 0$, we denote by $R(\tau,x,u) \subset I^{n_x}$ the reachable set

$$R(\tau,x,u) := \{\Phi(\tau;x,u,w) | x \in X, u \in U, w : [0,\tau] \to W\},$$

where $\Phi(\tau;x,u,w)$ is the state of the system at time τ starting from any state in the partition element $x \in X$, with constant control input $u \in U$ and for any disturbance taking values in W. Then, we can define the set of successors of the pair $(x,u) \in X \times U$ as

$$\delta(x,u) = \{x' \in X | x' \cap R(\tau,x,u) \neq \emptyset\}, \tag{2.3.1}$$

which determines a non-empty intersection between all partition elements in X and the reachable set.

Control synthesis

The second step that needs to be undertaken after constructing the abstraction is to synthesise a discrete controller on the finite abstract model to satisfy the desired specification. Graph search and model checking algorithms give us a powerful tool to combine the strength of the transition systems and wide range of high-level specifications, which can be expressed as temporal logic formulas.

Given a finite set of abstract states as target set $T \subseteq X$, the reachability game for T aims to find all abstract states that can be brought to the target set in a finite number of transitions of the abstraction.

Controller refinement

In the third final step of abstraction-based synthesis approaches, the discrete controller is refined to a controller for the continuous control system in Eq. (2.2.1). To achieve this, define the zero-order hold version of the discrete controller with the desired sampling period τ (which means keep the same value for the controller over the time period of length τ). The controller measures the current continuous state at each sampling time, find the partition element (or abstract state) that contains this continuous state, and apply the constant value of the discrete controller during the whole sampling period.

2.4 Data-driven control synthesis approaches

The more complex the dynamical system gets, the more difficult it is to analyse. This uncertainty and difficulty of having access to accurate models led to the introduction of data-driven approaches. Data-driven approaches for analysing, verifying, and synthesising systems have recently received significant attention [73]. They try to improve the efficiency and scalability of model-based approaches and to study problems in which a model of the system is either not available or costly and time-consuming to construct. These data-driven approaches are usually classified into two main categories: model-based and model-free approaches.

The model-based approaches usually rely on constructing a model from the available data and synthesising a controller using the constructed model. In classical control theory, this approach is the equivalent of the identification and control approach [67]. In recent years

there has been a surge in providing end-to-end approaches for guaranteeing the controller's performance [33].

In model-free data-driven approaches, the goal is to directly compute an optimal policy without generating a model. One of the approaches in model-based synthesis is to use an approximate model of the system. Given a prior inaccurate knowledge about the model of the system, a research line is to use data for refining the model and then synthesise a controller. This refining relies on the assumption over the underlying model of system. In doing so, such approaches assume a class of models and improve the estimation of the uncertainty within the model class. These approaches range from using Gaussian processes [110, 9], differential inclusions [38], rapidly-exploring random graphs [51], piecewise affine models [131], and model-based reinforcement learning algorithms [27]. On the other hand, data-driven model-free approaches compute the solution of the synthesis problem directly from data without constructing a model.

2.4.1 Data-driven abstraction-based control synthesis

Abstraction-based synthesis is an an effective approach to the synthesis of continuous-space stochastic systems. Existing results for stochastic systems include the construction of finite Markov decision processes (MDPs) for formal verification and synthesis [1, 144, 147]. In these approaches, a finite state model with probabilistic transitions is constructed based on the continuous probabilistic evolution of the system. Approximation methods for stochastic systems with error bounds that depend on higher-orders of the discretisation resolution is proposed in [145]. The assumption of continuous probabilistic evolution for error quantification is relaxed in [146, 150] to make the abstraction approach applicable to systems that have a mixture of probabilistic and deterministic evolution. Forward and backward computations of the safety and reachability probabilities using abstract models are studied in [149, 152], with extension of such techniques to infinite time-horizon properties in [163].

Utilising compact data structures and mu-calculus for formal verification and synthesis of stochastic systems against high-level specifications is studied recently [104, 103, 10, 11, 102]. Earlier in Sec. 2.3.1, we introduced how we can formally synthesise a control through an abstraction-based controller synthesis algorithm. Unfortunately, these techniques rely on state-space discretisation. Therefore, they severely suffer from the curse of dimensionality. So, their computation cost increases exponentially with the size of the state set. *Compositional* abstraction-based techniques have been recently introduced to mitigate this scalability issue. In this way, finite abstractions of large systems are constructed by dividing the system into a smaller subsystems and then doing the abstraction and synthesis locally [91, 90, 86]. Safety

verification and synthesis of stochastic systems based on *discretisation-free* approaches are studied in [123] using control *barrier certificates*. This notion is extended and used for verification and synthesis against high-level temporal properties in [70, 71].

On the downside, there is no guarantee that a control barrier certificate exists for a stochastic system.

The research on data-driven constructions of abstract models is limited. Legat et al. [94] provide an abstraction-based controller synthesis approach for hybrid systems by computing Lyapunov functions and Bellman-like Q-functions and using a branch and bound algorithm to solve the optimal control problem. Makdesi et al. [106] studied unknown monotone dynamical systems and sampled a set of trajectories generated by the system to find a minimal map overapproximating the dynamics of any system that produces these transitions. Consequently, they calculate an abstraction of the system related to this mapping and prove that an alternating bisimulation relation exists between them. However, this approach is restricted to monotone systems and does not apply to nonlinear dynamical systems.

We present in Chapter 3 of this thesis a data-driven abstraction-based control synthesis approach, where we construct an abstraction using sampled trajectories, which then can be used for synthesising a controller against any linear temporal logic specification. In the rest of this chapter, we will focus on reinforcement learning algorithms that are used extensively in the research presented in the thesis.

2.4.2 Reinforcement learning

Reinforcement learning (RL) [160] is a promising paradigm for sequential decision-making when a system model is unavailable or hard to construct and analyse. The objective of an RL algorithm is to find suitable action policies to maximise the collected rewards that depend on the states and actions taken at those states. In general, in RL algorithms, an agent learns from its own experience. The agent has some states, actions, and rewards. There is an initial state and a goal for the agent. After learning through mistakes, given a state, it can make an optimal move that guides it toward the end goal.

Reinforcement learning differs from other machine learning algorithms, such as supervised and unsupervised learning. In supervised learning, the goal is to generalise and extrapolate from a labelled training dataset which usually comes from a knowledgeable external expert. Supervised learning is an essential category of learning algorithms; however, it is inadequate for learning by interaction. RL also is different from unsupervised learning where the goal is to learn a hidden structure or pattern from an unlabelled dataset. It is tempting to assume that RL is a type of unsupervised learning; however, finding a hidden

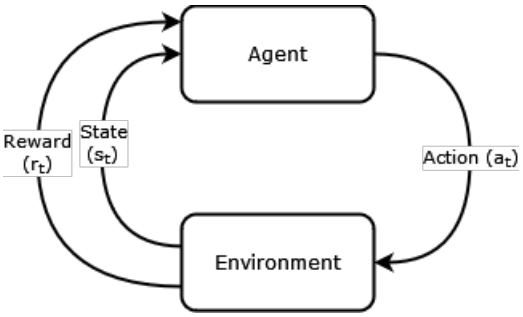


Fig. 2.2 Schematic of a reinforcement learning algorithm

structure, though it can be helpful, does not maximise the accumulated reward and does not give us the correct behaviour of the system.

RL learns through interaction and experience, which translates to exploration and exploitation. Throughout the learning, the agent needs to take a sequence of actions which maximises the reward based on her previous experience. On the other hand, the agent needs to discover good actions by trying other actions. In other words, the agent should exploit her already known knowledge about the environment while trying to explore the unknown parts of the environment. One challenge in RL research is to find a balance between exploration and exploitation.

Another key feature of RL is learning the whole task by interacting with an uncertain environment. In other disciplines, the problem of finding good behaviour can be divided into several sub-problems and solving all sub-problems leads to finding the policy. For instance, in control theory, we can solve a reachability problem by breaking it into a planning and tracking problem. Then, the combined solution of the two problems gives us the optimal controller. This is not the case at least in vanilla RL algorithms. In vanilla RL, we represent the goal as a reward structure and solve the problem without prior knowledge about the environment.

As we established, in any RL algorithm, we have an agent who wants to learn good behaviour, and this agent is interacting with an unknown environment. Beyond agent and environment, we can now introduce other vital elements of an RL algorithm: a policy, a reward function, a value function, and, optionally, an environment model [160].

A policy describes the agent's decision-making at a given time. Roughly speaking, a policy maps the perceived history or state of the environment to actions to be taken when the system behaves according to the perceived history or the visited states. A policy can be stochastic as well.

A reward function represents the goal in an RL problem. The environment sends a reward to the RL agent each time step. The agent's goal is to maximise the accumulated reward. In

this way, the agent locally can use this reward each time step and eventually steer the system towards a prefered behaviour. The environment sends an immediate reward when we take action, given the state and the action. The agent can change this reward directly by choosing a different action or indirectly by changing the state.

A value function acts as a prediction for the accumulated reward for the future. The immediate reward gives us a sense of quality for the current state. Although the immediate reward is necessary for the RL algorithm, we need to predict the accumulated reward for the future. Value function plays the role of accumulated reward approximator.

Last but not least, we need access to an environment's model. In RL algorithms, the agent needs to visit all state and action pairs infinitely often to converge. If we have access to physical systems satisfying this requirement is not easy to meet. In most cases, we can introduce a mathematical model miming the system's behaviour.

In the following, I will provide a brief account of model-free RL algorithms, which is a class of RL algorithms that do not construct or learn a model of the system based on data. This is in particular important as they allow a model-free analysis, verification and synthesis of the system. I will prove this observation in Chapters 5, 4, and 6.

2.4.3 Model-free reinforcement learning

Model-free RL [157] refers to a class of asymptotically space-efficient techniques because they do not construct a complete system model. In other words, model-free RL does not approximate the transition probabilities of the underlying system. These techniques include classical algorithms like TD(λ) [159] and Q-learning [172] as well as their extensions to deep neural networks such as deep deterministic policy gradient (DDPG) [96] and neural-fitted Q-iterations [128]. TD(λ) and Q-learning algorithms provide convergence results for finite-state MDPs with unknown transition probabilities: they compute optimal strategies as long as the number of training episodes goes to infinity. On the downside, these algorithms suffer severely if the state space of the underlying finite MDP is too large. Conversely, DDPG and neural-fitted Q-iterations can deal with large, finite MDPs with unknown transition probabilities at the cost of providing no convergence guarantee. Model-free reinforcement learning has achieved performance comparable to human experts in video and board games [162, 112, 138]. This success has motivated applications of RL to design controllers for safety-critical systems [96, 95] despite a lack of theoretical convergence guarantees of RL for general continuous state spaces [29]. There are two major ways to value a policy based on the infinite reward sequence that it generates, namely the average- and discounted-reward policy value formulations. They induce the average- and discounted-reward optimality criteria,

respectively [35]. We focus on Q-learning as a discounted RL and differential Q-learning as an example of average reawrd RL as a primary model-free RL algorithm in the following.

Q-learning

Discounted RL is the most prominent approach for designing a policy for infinite horizon tasks. The infinite-horizon discounted model takes the long-run reward of the agent into account, but rewards that are received in the future are geometrically discounted according to a discount factor. In this way the discounted RL algorithm can guarantee convergence. One of the earliest discounted RL algorithm is Q-learning. Watkins [172] proposed the first Q-learning algorithm for estimating the optimal action-value functions, Q-functions. The agent's goal is to devise a policy that maximises the accumulated reward. The algorithm has a function that calculates the quality of a state–action combination:

$$Q : S \times A \to \mathbb{R}.$$

Before learning begins, Q is initialised to a possibly arbitrary fixed value. Then, at each time t the agent selects an action a_t, observes a reward r_t, enters a new state s_{t+1} (that may depend on both the previous state s_t and the selected action), and Q is updated. The core of the algorithm is a Bellman equation as a simple value iteration update, using the weighted average of the old value and the new information:

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \Bigg)}^{\text{temporal difference}}$$

where r_t is the reward received when moving from the state s_t to the state s_{t+1} , and α is the learning rate $(0 < \alpha \leq 1)$.

Differential Q-learning

There is a rich history of studies in average reward RL [35, 100]. The lack of stopping criteria for multichain MDPs affects the generality of model-free RL algorithms. In this way, all model-free RL algorithms put some restrictions on the structure of MDP (e.g. ergodicity [2, 173] or communicating property). The work [141] proposes a model-based RL algorithm for maximising average reward objective with safety constraints for communicating MDPs.

It is worth noting that in a multichain setting, the state-of-the-art learning algorithms use model-based RL algorithms.

The state of the art model-free average reward RL algorithms assume that the underlying MDP is (weakly) communicating. One of the recent attempts is to provide off-policy RL algorithms for average reward objectives. Off-policy algorithms do not update the policy and value function at the same time. Under the communicating assumption, there exists a unique optimal reward rate $r^\star$ that is independent of start state. We define reward rate r for an arbitrary policy π and a start state s:

$$r(\pi, s) = \lim_{n\to\infty} \sum_{t=1}^{n} \mathbb{E}[R_t | S_0 = s, a_{0:t-1} \sim \pi]$$

where $r^\star = \sup_\pi r(s,\pi)$. It is worth noting $r^\star$ does not depend on the state.

In the following we explain the differential Q-learning as an example of average reward RL algorithms.

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t}_{\text{learning rate}} \cdot \underbrace{\delta_t}_{\text{temporal difference}}$$

$$Q^{new}(s, a) \leftarrow Q(s, a), \forall s, a \neq s_t, a_t$$

Where the temporal difference is defined as follows:

$$\delta_t = R_{t+1} - \underbrace{(\bar{R}_{t-1} + \eta \cdot \alpha_{t-1} \cdot \delta_{t-1})}_{\text{scalar estimate of } r^\star} + \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - Q(s_t, a_t)$$

where η is a positive constant and $\bar{R}_t$ is an estimate of $r^\star$.

2.4.4 Reward machines in reinforcement learning

Many objectives, including satisfaction of temporal properties on stochastic systems, do not admit an equivalent additive reward structure. A natural approach used in the literature (e.g., [92]) is to use heuristics for assigning additive rewards and then apply RL algorithms to obtain a policy. Unfortunately, there is no unique procedure for constructing these rewards, and the learning does not necessarily converge to the optimal policy. For instance, consider the objective that a robot should visit two regions infinitely often. We can specify a reward for the following behaviour: sequentially visit the first region m number of times and then the second region n number of times, with positive numbers m, n. The RL algorithm cannot advise on the best sequence of visiting these regions, which depends on the robot's dynamics.

Due to these limitations, there is a need to provide data-driven algorithms that do not require any heuristics and have suitable convergence guarantees to policies that are optimal for the satisfaction of temporal properties.

The development and use of formal reward structures for RL have witnessed an increased interest in recent years. This research area largely focuses on translating tasks into reward functions for reinforcement learning algorithms. The classical approach is to translate the task into an automaton with reward and then incorporate this automaton in the learning algorithm. There is a rich literature in using this technique in different scenarios for instance when we have finite-horizon or infinite-horizon tasks, or when we have discounted reward, average reward or total reward reinforcement learning.

Recently, there is a surge in using the structure of the reward in designing RL algorithms. In reward machine based RL algortihms on top of translating the task to an automaton, the algorithm also have access to the structure of the automaton as well. In this way, we can use the full potential of the information that we have i.e. the task by using for instance reward shaping or counterfactual reasoning.

In the rest of this thesis, we first present our novel data-driven framework for constructing finite abstractions of continuous systems with formal correctness guarantees. We then present our results on using RL for formal synthesis of controllers with convergence guarantees.

Chapter 3

Data-driven abstraction-based control synthesis

3.1 Chapter introduction

This chapter studies formal synthesis of controllers for continuous-space systems with unknown dynamics to satisfy requirements expressed as linear temporal logic formulas. Formal abstraction-based synthesis schemes rely on a precise mathematical model of the system to build a finite abstract model, which is then used to design a controller. The abstraction-based schemes are not applicable when the dynamics of the system are unknown. We propose a data-driven approach that computes the growth bound of the system using a finite number of trajectories. The growth bound together with the sampled trajectories are then used to construct the abstraction and synthesise a controller.

We show that our data-driven approach can be readily used as a model-free abstraction refinement scheme by modifying the formulation of the growth bound and providing similar sample complexity results. The performance of our approach is shown on three case studies.

The research presented in the chapter has been submitted for publication in Nonlinear Analysis: Hybrid Systems, and the e-print can also be viewed on arXiv [78]. This research was the result of a collaboration with the Max Planck Institute for Software Systems. My role in this research is to provide the theoretical results, help with the simulations, and write the paper.

3.2 Introduction

One of the major objectives in the design of safety-critical systems is to ensure their safe operation while satisfying high-level requirements. Examples of safety-critical systems include power grids, autonomous vehicles, traffic control, and battery-powered medical devices. The automated design of controllers for such systems that can fulfil the given requirements has received significant attention recently. These systems can be represented as control systems with continuous state spaces. Within these continuous spaces, it is challenging to leverage automated control synthesis methods that provide satisfaction guarantees for high-level specifications, such as those expressed in Linear Temporal Logic [8, 13, 161, 49].

A common approach to tackle the continuous nature of the state space is to use abstraction-based controller design (ABCD) schemes [161, 13, 105, 133]. The first step in the ABCD scheme is to compute a finite abstraction by discretising the state and action spaces. Finite abstractions are connected to the original system via an appropriate behavioural relation such as feedback refinement relations or alternating bisimulation relations [127, 161]. Under such behavioural relations, trajectories of the abstraction are related to the ones of the original system. Therefore, a controller designed for the simpler finite abstract system can be refined to a controller for the original system. The controller designed by the ABCD scheme is described as formal due to the guarantees on the satisfaction of the specification by the original system in a closed loop with the designed controller.

ABCD schemes generally rely on a precise mathematical model of the system. This stems from the fact that establishing a behavioural relation between the original system and its finite abstraction uses reachability analysis over the dynamics of the original system that require knowledge of the dynamical equations. Although such equations can, in principle, be derived, for instance, by using physics laws, the real-world control systems are a mixture of differential equations, block diagrams, and lookup tables. Therefore, extracting a clean analytical model for systems of practical interest could be infeasible. A promising approach to tackle this issue is to develop data-driven control synthesis schemes with appropriate formal (probabilistic) guarantees.

The main contribution of this chapter is to provide a data-driven approach for the formal synthesis of controllers to satisfy temporal specifications. We focus on continuous-time nonlinear dynamical systems whose dynamics are unknown, but sampled trajectories are available. Our approach constructs a *finite* abstract model of the system using only a finite number of sampled trajectories and the growth bound of the system. We formulate the computation of the growth bound as a robust convex program (RCP) with an infinite uncountable number of constraints. We then approximate the solution of the RCP with a scenario convex program (SCP) that has a finite number of constraints and can be solved

using only a finite set of sampled trajectories. We establish a sample complexity result that gives a lower bound for the required number of trajectories to guarantee the correctness of the growth bound *over the whole state space* with a given confidence. We also provide a sample complexity result for the satisfaction of the specification on the system in closed loop with the designed controller for given confidence. Our result requires estimating a bound on the Lipschitz constant of the system with respect to the initial state that we obtain using extreme value theory. As our last contribution, we show that our approach can be extended to a model-free abstraction refinement scheme by modifying the formulation of the growth bound and providing similar sample complexity results. We demonstrate the performance of our approach on three case studies.

The remainder of this chapter is organised as follows. After discussing the related work, Section 3.3 covers preliminaries on dynamical systems and finite abstractions and provides the problem statement. In Section 3.4, we present the assumptions and theoretical results needed for connecting RCPs and their corresponding SCPs. In Section 3.5, we present our approach on the data-driven computation of the growth bound and the abstraction and prove our sample complexity result. This section also discusses the estimation of the Lipschitz constant of the system for computing the number of samples. Section 3.6 discusses the extension of our approach to a data-driven abstraction refinement scheme. Several numerical examples are provided in Section 3.7 that support the theoretical findings of the chapter. Finally, Section 3.8 contains concluding remarks and future research directions.

Related Work. There is an extensive body of literature on *model-based* formal synthesis for both deterministic and probabilistic systems. We refer the reader to the books [8, 161, 13] and seminal papers [49, 1]. *Data-driven* approaches for analysis, verification, and synthesis of systems have received significant attention recently to improve efficiency and scalability of model-based approaches, and to study problems in which a model of the system is either not available or costly and time-consuming to construct.

Given a prior inaccurate knowledge about the model of the system, a research line is to use data for refining the model and then synthesise a controller. Such approaches assume a class of models and improve the estimation of the uncertainty within the model class. These approaches range from using Gaussian processes [110, 9], differential inclusions [38], rapidly-exploring random graphs [51], piecewise affine models [131], and model-based reinforcement learning algorithms [27]. A data-driven framework is proposed in [42] for verifying properties of hybrid systems when the continuous dynamics are unknown but the discrete transitions are known.

Data-driven model-free approaches compute the solution of the synthesis problem directly from data without constructing a model. In [64], authors provide a reach-avoid Q-learning

algorithm with convergence guarantees for an arbitrarily tight conservative approximation of the reach-avoid set. The paper [170] proposes a falsification-based adversarial reinforcement learning algorithm for metric temporal logic specifications. Satisfying signal temporal logic specifications is studied in [168] using counterexample-guided inductive synthesis on nonlinear systems, and using model-free reinforcement learning in [75] for Markov decision processes. A learning framework for synthesis of control-affine systems in provided in [158]. The authors of [171] study learning from demonstration while preventing the violation of safety under the learned policy.

The research on data-driven constructions of abstract models is very limited. Legat et al. [94] provide an abstraction-based controller synthesis approach for hybrid systems by computing Lyapunov functions and Bellman-like Q-functions, and using a branch and bound algorithm to solve the optimal control problem. Makdesi et al. [106] studied unknown monotone dynamical systems and sampled a set of trajectories generated by the system to find a minimal map overapproximating the dynamics of any system that produces these transitions. Consequently, they calculate an abstraction of the system related to this map and prove that an alternating bisimulation relation exists between them. In contrast, our approach is not restricted to monotone systems and is applicable to any nonlinear dynamical system.

The closest work to our problem formulation is the work by Devonport et al. [34], where a data-driven abstraction technique is provided for satisfying finite-horizon specifications. Our results are more general than the work [34] in two main aspects. First, our constructed abstraction can be used for synthesising a controller against any linear temporal logic specification. Our sample complexity result is independent of the horizon of the specification and does not limit using the approach on finite-horizon specifications. Second, the guarantee provided in [34] is based on a Probably Approximately Correct (PAC) approach. It means that the constructed abstraction is always wrong on a small subset of the state space whose size can be made smaller at the cost of high computational efforts. Our formulated guarantee ensures that the abstraction is valid on the entire state space with high confidence. The confidence is interpreted from the frequentist view of probability: if we run our algorithm multiple times, we always get a correct abstraction except a small number of times reflected in the confidence value.

In our approach, we formulate the synthesis problem as a robust convex program and approximate it with a scenario program. Such approximations have been studied for the past two decades. Calafiore and Campi [22] provide an approximately feasible solution for the associated chance constrained program by solving a scenario program, and give a sample complexity result. Relaxing the convexity assumption is studied in [156] by assuming additional properties of the underlying probability distributions. We will use the results by

Esfahani et al. [41], where the optimality of the robust program is connected directly to the scenario program. These results are also used recently in the papers [87, 132] for performing data-driven verification and synthesis. Inspired by the works [175, 174], we will use extreme value theory to estimate the Lipschitz constant needed for the sample complexity results.

3.3 Preliminaries and problem statement

3.3.1 Preliminaries

Notation. We denote the set of natural, real, positive real, and non-negative real numbers by $\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}_{>0}$, and $\mathbb{R}_{\geq 0}$, respectively. The set of natural numbers including zero is denoted by $\mathbb{N}_{\geq 0}$. We use superscript $n > 0$ with these sets to denote the Cartesian product of n copies of these sets. The power set of a set A is denoted by 2^A and includes all the subsets of A. For any $x, y \in \mathbb{R}^n$ with $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$, and a relational symbol $\triangleright \in \{\leq, <, =, >, \geq\}$, we write $x \triangleright y$ if $x_i \triangleright y_i$ for every $i \in \{1, 2, \ldots, n\}$. A matrix $M \in \mathbb{R}^{n \times n}$ is said to be non-negative if all of its entries are non-negative. We use the operators $|\cdot|$ and $\|\cdot\|$ to denote the element-wise absolute value and the infinity norm, respectively. We use the notation $\Omega_\varepsilon(c) \coloneqq \{x \in \mathbb{R}^n \mid \|x - c\| \leq \varepsilon\}$ to denote the ball with respect to infinity norm centred at $c \in \mathbb{R}^n$ with radius $\varepsilon \in \mathbb{R}^n_{>0}$. We consider a probability space $(\Omega, \mathscr{F}_\Omega, \mathbb{P}_\Omega)$, where Ω is the sample space, $\mathscr{F}_\Omega$ is a sigma-algebra on Ω comprising its subsets as events, and $\mathbb{P}_\Omega$ is a probability measure that assigns probabilities to events.

Control Systems. A continuous-time control system is a tuple $\Sigma = (X, x_{\mathsf{in}}, U, W, f)$, where $X \subset \mathbb{R}^n$ is the state space, $x_{\mathsf{in}} \in X$ is the initial state, $U \subset \mathbb{R}^m$ is the input space, and $W \subset \mathbb{R}^n$ is the disturbance space which is assumed to be a compact set containing the origin. The vector field $f : X \times U \to X$ is such that $f(\cdot, u)$ is locally Lipschitz for all $u \in U$. The evolution of the state of Σ is characterised by the differential equation

$$\dot{x}(t) = f(x(t), u(t)) + w(t), \tag{3.3.1}$$

where $w(t) \in W$ represents the additive disturbance.

We consider the class of input and disturbance signals $u : \mathbb{R}_{\geq 0} \to U$ and $w : \mathbb{R}_{\geq 0} \to W$ to be piecewise constant with respect to a *sampling time* $\tau > 0$, i.e., $u(t) = u(k\tau)$ and $w(t) = w(k\tau)$ for every $k\tau \leq t < (k+1)\tau$ and $k \in \mathbb{N}_{\geq 0}$. Given a sampling time $\tau > 0$, an initial state $x_0 \in X$, a constant input $u \in U$, and a constant disturbance $w \in W$, define the *continuous-time trajectory* $\zeta_{x_0,u,w}$ of the system on the time interval $[0, \tau]$ as an absolutely continuous function $\zeta_{x_0,u,w} : [0, \tau] \to X$ such that $\zeta_{x_0,u,w}(0) = x_0$, and $\zeta_{x_0,u,w}$ satisfies the differential equation $\dot{\zeta}_{x_0,u,w}(t) = f(\zeta_{x_0,u,w}(t), u) + w$ for almost all $t \in [0, \tau]$. The solution

of (3.3.1) from x_0 for the constant control input u with $w(t) = 0$ for all $t \geq 0$ is called the *nominal trajectory* of the system. For a fixed τ, we define the operators

$$\varphi(x,u,w) := \zeta_{x,u,w}(\tau) \text{ and}$$
$$\Phi(x,u) := \{\varphi(x,u,w) \mid w \in W\}$$

respectively for the trajectory at time τ and the set of such trajectories starting from x.

In this chapter, we consider control systems $\Sigma = (X, x_{\mathsf{in}}, U, W, f)$ whose vector field f is not known, but we can observe their *time-sampled trajectories*. A sequence $x_0, x_1, x_2, \ldots$ is a time-sampled trajectory of Σ if for each $i \geq 0$, we have $x_{i+1} \in \Phi(x_i, u_i)$ for some $u_i \in U$.

Finite-state Abstraction of Control Systems. Let $\Sigma = (X, x_{\mathsf{in}}, U, W, f)$ be a control system with a sampling time $\tau > 0$. We consider abstract models constructed by using uniformly sized rectangular partitioning of X and U. We select representative points from these partition sets to obtain $\widehat{X}$ and $\widehat{U}$. We assume that the radius of these partition sets are provided as vectors $\eta_x \in \mathbb{R}^n_{>0}$ and $\eta_u \in \mathbb{R}^m_{>0}$, respectively. Parameters η_x, η_u are inputs to the abstraction procedure. A *finite-state abstraction* of Σ is characterised by the tuple $\widehat{\Sigma} = (\widehat{X}, \widehat{U}, \hat{f})$, where $\widehat{X}$ is the set of representative points from a finite partition of X, $\widehat{U}$ is the set of representative points from a finite partition of U, and $\hat{f} : \widehat{X} \times \widehat{U} \to 2^{\widehat{X}}$ is a set-valued map. For any $\hat{x} \in \widehat{X}$ and $\hat{u} \in \widehat{U}$, $\hat{x}' \in \hat{f}(\hat{x}, \hat{u})$ if there is a pair of states $x \in \Omega_{\eta_x}(\hat{x})$ and $x' \in \Omega_{\eta_x}(\hat{x}')$ such that $x' \in \Phi(x, \hat{u})$. Note that, the larger η_x is (where comparison is made dimension-wise), the smaller is the cardinality of $\widehat{X}$ resulting in a coarser abstraction. On the other hand, the smaller η_x is, the more precise the abstraction $\widehat{\Sigma}$ will be, increasing the chance of a successful controller synthesis (see, e.g., [161] for more details on this construction).

Feedback Controller. A *feedback controller* for $\widehat{\Sigma}$ is a function $\widehat{C}\colon \widehat{X} \to \widehat{U}$. We denote by $\widehat{C} \parallel \widehat{\Sigma}$ the feedback composition of $\widehat{\Sigma}$ and $\widehat{C}$. The set of trajectories of the closed-loop system $\widehat{C} \parallel \widehat{\Sigma}$ consists of all finite trajectories $\hat{x}_0, \hat{x}_1, \hat{x}_2, \ldots$ such that for all $i \in \mathbb{N}_{\geq 0}$, we have $\hat{x}_{i+1} \in \hat{f}(\hat{x}_i, \widehat{C}(\hat{x}_i))$.

We can relate a finite abstraction $\widehat{\Sigma}$ to Σ for control synthesis purposes. Simulation relations or feedback refinement relations [161, 127] established between Σ and $\widehat{\Sigma}$ enable us to refine a controller $\widehat{C}$ designed for $\widehat{\Sigma}$ to a controller C for Σ. In its general form, such a refined controller C maps the current states $x \in \Omega_{\eta_x}(\hat{x})$ into an input $u = \widehat{C}(\hat{x})$ for Σ. The purpose of designing $\widehat{C}$ is that the closed-loop system $C \parallel \Sigma$ satisfies the given objective. Our synthesis objective is expressed as Linear Temporal Logic (LTL) specifications. We refer to [8] and references therein for detailed syntax and semantics of LTL. For the details of the controller synthesis and tool implementation using abstract models we refer to [127] and [129], respectively.

3.3.2 Problem statement

We study abstraction-based control design (ABCD) for systems with *unknown* dynamics using available data from the system such that a given specification is satisfied with high confidence on the closed-loop system.

Assumption 3.1 *The vector field f of the control system $\Sigma = (X, x_{\mathsf{in}}, U, W, f)$ in unknown, but sampled trajectories of the system can be obtained in the form of $\mathscr{S}_N := \{(x_k, u_k, x'_k) \,|\, x'_k \in \Phi(x_k, u_k),\, k = 1, 2, \ldots, N\}$.*

Problem 3.1 (Data-driven ABCD) ***Inputs:*** *Control system $\Sigma = (X, x_{\mathsf{in}}, U, W, f)$ with unknown vector field f, specification Ψ, sampled trajectories $\mathscr{S}_N$, and confidence parameter $\beta \in (0,1)$.*

Outputs: *Abstract model $\widehat{\Sigma}$, abstract controller $\widehat{C}$, and refined controller C for Σ, such that $C \parallel \Sigma$ satisfies Ψ with confidence $(1-\beta)$.*

The first step of the ABCD is to compute a finite abstraction $\widehat{\Sigma}$ for Σ. Once such an abstraction is computed, synthesis of the controller $\widehat{C}$ and refining it to C follow the model-based ABCD scheme. Therefore, the main challenge is to provide a data-driven computation of the abstraction $\widehat{\Sigma}$ that is a true overapproximation of Σ with confidence $(1-\beta)$.

Problem 3.2 (Data-driven Abstraction) ***Inputs:*** *Control system $\Sigma = (X, x_{\mathsf{in}}, U, W, f)$ with unknown vector field f, sampled trajectories $\mathscr{S}_N$, discretisation parameters η_x and η_u, and confidence parameter $\beta \in (0,1)$.*

Outputs: *Finite model $\widehat{\Sigma}$ that is an abstraction of Σ with confidence $(1-\beta)$.*

In this chapter, we tackle Problem 3.2 by showing how to construct $\widehat{\Sigma}$ from sampled trajectories $\mathscr{S}_N$, and provide a lower bound on the data size N in order to ensure correctness of the abstraction with confidence $(1-\beta)$. The required theoretical tools are presented in the next section.

3.4 Robust convex programs

In this section, we describe robust convex programs (RCPs) and data-driven approximation of their solution. In Sections 3.5 and 3.6, we show how such an approximation can be used for solving the data-driven abstraction in Problem 3.2.

Let $T \subset \mathbb{R}^q$ be a compact convex set for some $q \in \mathbb{N}$ and $c \in \mathbb{R}^q$ be a constant vector. Let $(\mathscr{D}, \mathscr{B}, \mathbb{P})$ be the probability space of the *uncertainty* and $g\colon T \times \mathscr{D} \to \mathbb{R}$ be a measurable

function, which is convex in the first argument for each $d \in \mathscr{D}$, and bounded in the second argument for each $\theta \in T$. The robust convex program (RCP) is defined as

$$\text{RCP: } \begin{cases} \min_\theta c^\top \theta \\ s.t.\ \theta \in T \text{ and } g(\theta, d) \leq 0 \quad \forall d \in \mathscr{D}. \end{cases} \tag{3.4.1}$$

Computationally tractable approximations of the optimal solution of the RCP (3.4.1) can be obtained using *scenario convex programs* (SCPs) that only require gathering finitely many samples from the uncertainty space [114]. Let $(d_i)_{i=1}^N$ be N independent and identically distributed (i.i.d.) samples drawn according to the probability measure $\mathbb{P}$. The SCP corresponding to the RCP (3.4.1) strengthened with $\gamma \geq 0$ is defined as

$$SCP_\gamma : \begin{cases} \min_\theta c^\top \theta \\ s.t.\ \theta \in T, \text{ and } g(\theta, d_i) + \gamma \leq 0 \ \forall i \in \{1, 2, \ldots, N\}. \end{cases} \tag{3.4.2}$$

We denote the optimal solution of RCP (3.4.1) as θ^*_{RCP} and the optimal solution of SCP_γ (3.4.2) as θ^*_{SCP}. Note that θ^*_{RCP} is a single deterministic quantity but θ^*_{SCP} is a random quantity that depends on the i.i.d. samples $(d_i)_{i=1}^N$ drawn according to $\mathbb{P}$. The RCP (3.4.1) is a challenging optimisation problem since the cardinality of $\mathscr{D}$ is infinite and the optimisation has infinite number of constraints. In contrast, the SCP (3.4.2) is a convex optimisation with finite number of constraints for which efficient optimisation techniques are available [18]. The following theorem provides a sample complexity result for connecting the optimal solution of the SCP_γ to that of the RCP.

Theorem 3.1 ([114]) *Assume that the mapping $d \mapsto g(\theta, d)$ in* (3.4.1) *is Lipschitz continuous uniformly in $\theta \in T$ with Lipschitz constant L_d and let $h\colon [0,1] \to \mathbb{R}_{\geq 0}$ be a strictly increasing function such that*

$$\mathbb{P}(\Omega_\varepsilon(d)) \geq h(\varepsilon), \tag{3.4.3}$$

for every $d \in \mathscr{D}$ and $\varepsilon \in [0,1]$. Let θ^*_{RCP} be the optimal solution of the RCP (3.4.1) *and θ^*_{SCP} the optimal solution of SCP_γ* (3.4.2) *with*

$$\gamma = L_d h^{-1}(\varepsilon) \tag{3.4.4}$$

computed by taking N i.i.d. samples $(d_i)_{i=1}^N$ *from* $\mathbb{P}$*. Then* θ^*_{SCP} *is a feasible solution for the RCP with confidence* $(1-\beta)$ *if the number of samples* $N \geq N(\varepsilon,\beta)$*, where*

$$N(\varepsilon,\beta) := min\left\{N \in \mathbb{N} \,\Big|\, \sum_{i=0}^{q-1} \binom{N}{i} \varepsilon^i (1-\varepsilon)^{N-i} \leq \beta\right\}, \tag{3.4.5}$$

with q being the dimension of the decision vector $\theta \in T$*.*

3.5 Data-driven abstraction

In this section, we first discuss the steps required for model-based abstraction of control systems. We then show how this can be formulated as an RCP and present its associated SCP. Finally, we use the connection between the RCPs and SCPs in Theorem 3.1 to provide a lower bound for number of required samples to certify a desired confidence. The simplifying assumption used in this section is that samples from the *nominal trajectories* of the system Σ is also available in the form of $\{(x_k,u_k,x'_k)\,|\,x'_k = \varphi(x_k,u_k,0),\, k=1,2,\ldots,N\}$. We discuss in the next section how this assumption can be relaxed by modifying the inequality of the growth bound.

3.5.1 Growth bound for reachable sets

Consider a control system $\Sigma = (X, x_{\mathsf{in}}, U, W, f)$ with the disturbance set $W = [-\bar{w}, \bar{w}]$ for some vector $\bar{w} \in \mathbb{R}^n_{\geq 0}$. Let η_x and η_u be discretisation parameters for the state and input spaces X and U used to construct $\widehat{X}$ and $\widehat{U}$ of sizes n_x and n_u, respectively. The first step of ABCD is to compute a finite abstraction $\widehat{\Sigma} = (\widehat{X}, \widehat{U}, \hat{f})$ using overapproximations of the reachable sets for every pair of abstract state and input. The reachable set for every pair $(\hat{x},\hat{u}) \in \widehat{X} \times \widehat{U}$ is defined as

$$Reach(\hat{x},\hat{u}) := \{x' \in \Phi(x,\hat{u}) \mid x \in \Omega_{\eta_x}(\hat{x})\}.$$

The set $Reach(\hat{x},\hat{u})$ is usually overapproximated using the growth bound of the system dynamics [127].

Definition 3.1 *The growth bound of a control system* Σ *with abstract state and input spaces* $\widehat{X}, \widehat{U}$ *is a function* $\kappa\colon \mathbb{R}^n_{\geq 0} \times \widehat{X} \times \widehat{U} \to \mathbb{R}^n_{\geq 0}$ *that satisfies*

$$\begin{aligned} &|\varphi(x,\hat{u},w) - \varphi(\hat{x},\hat{u},0)| \leq \kappa(|x-\hat{x}|,\hat{x},\hat{u}) \\ &\quad \forall \hat{x} \in \widehat{X},\, \forall \hat{u} \in \widehat{U},\, \forall x \in \Omega_{\eta_x}(\hat{x}),\, \forall w \in W. \end{aligned} \tag{3.5.1}$$

Note that $\varphi(\hat{x},\hat{u},0)$ is the nominal (disturbance-free) trajectory of the system. Using this definition, for every abstract state-input pair $(\hat{x},\hat{u}) \in \widehat{X} \times \widehat{U}$, the reachable set $Reach(\hat{x},\hat{u})$ is over-approximated with a ball centered at $z(\hat{x},\hat{u}) := \varphi(\hat{x},\hat{u},0)$ with radius $\lambda(\hat{x},\hat{u}) := \kappa(\eta_x,\hat{x},\hat{u})$.

When the system dynamics are known, it is shown in [127] that the growth bound can be computed as

$$\kappa(r,\hat{x},\hat{u}) = e^{L(\hat{u})\tau} r + \int_0^\tau e^{L(\hat{u})s} \bar{w} ds, \tag{3.5.2}$$

for all $r \in \mathbb{R}^n_{\geq 0}$, $\hat{x} \in \widehat{X}$, and $\hat{u} \in \widehat{U}$, where $L\colon \widehat{U} \to \mathbb{R}^{n\times n}$ is a matrix such that the entries of $L(\hat{u})$ satisfy the following inequality for all $x \in X$:

$$L_{i,j}(\hat{u}) \geq \begin{cases} D_j f_i(x,\hat{u}) & i = j \\ |D_j f_i(x,\hat{u})| & i \neq j, \end{cases} \tag{3.5.3}$$

for all $i,j \in \{1,2,\ldots,n\}$, where $f_i(x,u)$ is the i^{th} element of the vector field $f(x,u)$ and $D_j f_i$ is its partial derivative with respect to the j^{th} element of x.

3.5.2 SCP for the computation of growth bound

When the model of the system is unknown, the matrix $L(\hat{u})$ defined using (3.5.3) is not computable, thus the growth bound in (3.5.2) is not available. To tackle this bottleneck, we use the parameterisation

$$\kappa(\theta)(r,\hat{x},\hat{u}) := \theta_1(\hat{x},\hat{u})r + \theta_2(\hat{x},\hat{u}), \forall r \in \mathbb{R}^n_{\geq 0}, \hat{x} \in \widehat{X}, \hat{u} \in \widehat{U}, \tag{3.5.4}$$

where $\theta_1 \in \mathbb{R}^{n\times n}$ and $\theta_2 \in \mathbb{R}^n$. We denote by $\theta \in \mathbb{R}^{n^2+n}$ the concatenation of columns of θ_1 and θ_2.

Remark 3.1 *The parameterised growth bound in* (3.5.4) *is linear with respect to r similar to* (3.5.2)*, but is more general and less conservative by allowing θ_1, θ_2 to depend on $\hat{x}$ (i.e., they are defined locally for each abstract state).*

Theorem 3.2 *The inequality* (3.5.1) *with the parameterised growth bound* (3.5.4) *can be written as the robust convex program*

$$\begin{cases} \min_\theta c^\top \theta \\ s.t.\ 0 \leq \theta \leq \bar{\theta},\ and\ \forall x \in \Omega_{\eta_x}(\hat{x}),\ \forall w \in W, \\ \qquad |\varphi(x,\hat{u},w) - \varphi(\hat{x},\hat{u},0)| - \kappa(\theta)(|x-\hat{x}|,\hat{x},\hat{u}) \leq 0, \end{cases} \tag{3.5.5}$$

where $c = [1,1,\ldots,1] \in \mathbb{R}^{n^2+n}$ and $\bar{\theta}$ is a sufficiently large positive vector.

Proof 3.1 *We first show that the optimisation* (3.5.5) *is in fact a robust convex programme. Let* $\mathscr{D} = \Omega_{\eta_x}(\hat{x}) \times W$ *be the uncertainty space and*

$$g(\theta, x, w) := |\varphi(x, \hat{u}, w) - \varphi(\hat{x}, \hat{u}, 0)| - \kappa(\theta)(|x - \hat{x}|, \hat{x}, \hat{u})$$

for all $x \in \Omega_{\eta_x}(\hat{x})$ *and* $w \in W$ *and fixed* $(\hat{x}, \hat{u}) \in \widehat{X} \times \widehat{U}$. *We need to show that* g *is convex in* θ *for each* $(x, w) \in \mathscr{D}$ *and bounded in* (x, w) *for every* $\theta \in [0, \bar{\theta}]$. *The convexity holds due to the parameterisation of* $\kappa(\theta)$ *in* (3.5.4) *being linear with respect to the optimisation variables in* θ. *The boundedness holds due to the set* $\mathscr{D}$ *being compact and trajectories of the system being continuous.*

We note that any feasible solution for the optimisation (3.5.5) *gives a function* κ *that satisfies the inequality* (3.5.1) *for* Σ. *Such a system will also have a growth bound of the form* (3.5.2) *that is a feasible solution for* (3.5.5). *To see this, we show that* $\theta_1 = e^{L(\hat{u})\tau}$ *and* $\theta_2 = \int_0^\tau e^{L(\hat{u})s}\bar{w}ds$ *are always non-negative. By definition, all the entries of* $L(\hat{u})$ *are non-negative except the diagonal entries. We decompose this matrix as* $L(\hat{u}) = Q + D$, *where* D *is a diagonal matrix with all diagonal entries equal to the constant* $max_i \sum_j |L_{i,j}(\hat{u})|$ *and* $Q = L(\hat{u}) - D$ *is a sub-stochastic matrix as (i) its non-diagonal entries are non-negative* ($Q_{i,j} = L_{i,j}(\hat{u}) \geq o$ *for* $i \neq j$), *(ii) its diagonal entries are non-positive* ($Q_{i,i} \leq 0$), *and finally (iii) all of its row sums are non-positive. Note that* D *is a multiple of identity matrix and therefore,* $DQ = QD$ *and* $e^{(Q+D)\tau} = e^{Q\tau}e^{D\tau}$. *Further, we define the matrix*

$$\bar{Q} = \begin{bmatrix} Q & \vdots & -Q\mathbf{1} \\ \dots & \dots & \dots \\ 0^\top & \vdots & 0 \end{bmatrix},$$

where 0 *and* $\mathbf{1}$ *represent* $n-$*dimensional vectors with all entries equal to zero and one, respectively. Note that* $\bar{Q}$ *is a stochastic matrix since* $\bar{Q}_{i,i} = -\sum_{j \neq i} \bar{Q}_{i,j}$ *for every* $1 \leq i \leq n+1$ *and* $\bar{Q}_{i,j} \geq 0$ *for* $i \neq j$. *Therefore, matrix* $\bar{Q}$ *correspond to the transition probability matrix of a continuous-time Markov chain with state space* $\{1, 2, \dots, n+1\}$ *(see, e.g., [8] for more details). Therefore, the entry* (i, j) *of* $e^{\bar{Q}\tau}$ *is the probability that the Markov chain reaches the* j^{th} *state from the* i^{th} *state at time* τ, *which is a non-negative quantity. Further, we have*

$$e^{\bar{Q}\tau} = \begin{bmatrix} e^{Q\tau} & \vdots & \mathbf{1} - e^{Q\tau}\mathbf{1} \\ \dots & \dots & \dots \\ \mathbf{0}^\top & \vdots & 1 \end{bmatrix}.$$

Therefore, $e^{Q\tau}$ is non-negative, which gives $e^{L(\hat{u})\tau} = e^{Q\tau}e^{D\tau}$ since $e^{D\tau} \geq 0$. This naturally results in θ_1 and θ_2 being non-negative as the integral of non-negative functions.

To construct the SCP$_\gamma$ associated with the RCP (3.5.5), we fix $\hat{x} \in \widehat{X}$ and $\hat{u} \in \widehat{U}$, consider a uniform distribution on the space $\mathscr{D} = \Omega_{\eta_x}(\hat{x}) \times W$ and obtain N i.i.d. sample trajectories $\mathscr{S}_N = \{(x_i, \hat{u}, x_i') \,|\, x_i' \in \Phi(x_i, \hat{u}), i = 1, 2, \ldots, N\}$. Note that every x_i' corresponds to a random disturbance $w_i \in W$. The SCP$_\gamma$ is

$$\begin{cases} \min_\theta c^\top \theta \\ s.t.\ 0 \leq \theta \leq \bar{\theta} \text{ and } \forall i \in \{1, \ldots, N\}, \\ \quad |x_i' - x_{nom}'| - \theta_1(\hat{x}, \hat{u})|x_i - \hat{x}| + \theta_2(\hat{x}, \hat{u}) + \gamma \leq 0, \end{cases} \tag{3.5.6}$$

where $x_{nom}' := \varphi(\hat{x}, \hat{u}, 0)$ and $\gamma \in \mathbb{R}_{\geq 0}$.

Theorem 3.3 *For any $\hat{x} \in \widehat{X}$ constructed with discretisation size η_x, any $\hat{u} \in \widehat{U}$, and the disturbance set $W = [-\bar{w}, \bar{w}]$, the optimal solution of* (3.5.6) *gives a growth bound for the system Σ corresponding to $(\hat{x}, \hat{u})$ with confidence $(1 - \beta)$, when the number of samples $N \geq N(\varepsilon, \beta)$ and*

$$\gamma = 4L_\varphi(\hat{u}) \sqrt[2n]{\varepsilon \prod_{i=1}^{n} \eta_x(i) \prod_{i=1}^{n} \bar{w}(i)}, \tag{3.5.7}$$

where $\varepsilon \in [0, 1]$, n is the dimension of the state space and $L_\varphi(\hat{u})$ is the Lipschitz constant of the system trajectories $\varphi(x, \hat{u}, w)$ with respect to (x, w).

Proof 3.2 *We apply Theorem 3.2 to the RCP* (3.5.5) *for fixed $\hat{x} \in \widehat{X}$ and $\hat{u} \in \widehat{U}$. Define*

$$\begin{aligned} g(\theta, x, w) := \max\{&|\varphi(x, \hat{u}, w) - \varphi(\hat{x}, \hat{u}, 0)| \\ &- \theta_1(\hat{x}, \hat{u})|x - \hat{x}| - \theta_2(\hat{x}, \hat{u})\}, \end{aligned} \tag{3.5.8}$$

where the $\max\{\cdot\}$ *is applied to the elements of its argument that belongs to $\mathbb{R}^n$. Since the distribution on $\mathscr{D} = \Omega_{\eta_x}(\hat{x}) \times W$ is uniform, we choose*

$$h(\varepsilon) = \mathbb{P}(\Omega_\varepsilon(d)) = \frac{(\varepsilon/2)^{2n}}{\prod_{i=1}^{n} \eta_x(i) \prod_{i=1}^{n} \bar{w}(i)}$$

to satisfy the inequality (3.4.3). *Note that $h(\varepsilon)$ gives the probability of choosing a point within the $2n-$ball $\Omega_\varepsilon(d)$ uniformly at random. We use Equation* (3.4.4) *as $\gamma = L_d h^{-1}(\varepsilon)$ to get the value of γ in* (3.5.7). *It only remains to show that $g(\theta, x, w)$ is Lipschitz continuous*

with constant $L_d = 2L_\varphi(\hat{u})$. Note that $L_\varphi(\hat{u})$ is the Lipschitz constant of $\varphi(x,\hat{u},w)$ with respect to (x,w), and satisfies

$$\|\varphi(x,\hat{u},w) - \varphi(x',\hat{u},w')\| \leq L_\varphi(\hat{u})\|(x,w) - (x',w')\| \tag{3.5.9}$$

for all $x,x' \in \Omega_{\eta_x}(\hat{x})$ and $w,w' \in W$. Since $\|\theta_1(\hat{x},\hat{u})\|$ can be bounded by $L_\varphi(\hat{u})$, we get that

$$\begin{aligned}
&\|g(\theta,x,w) - g(\theta,x',w')\| \\
&\leq \|\varphi(x,\hat{u},w) - \varphi(x',\hat{u},w')\| + \|\theta_1(\hat{x},\hat{u})\|\|x-x'\| \\
&\leq L_\varphi(\hat{u})\|(x,w) - (x',w')\| + L_\varphi(\hat{u})\|x-x'\| \\
&\leq 2L_\varphi(\hat{u})\|(x,w) - (x',w')\|,
\end{aligned}$$

Therefore, $g(\theta,x,w)$ is Lipschitz continuous with constant $2L_\varphi(\hat{u})$. This completes the proof.

Remark 3.2 *The value of γ in* (3.5.7) *depends on the Lipschitz constant L_φ. We provide an algorithm in the next subsection for estimating this constant using sampled trajectories of the system. Note that as the above proof shows, the estimated quantity $\theta_1 = L_\varphi \mathbf{1}_{n\times n}$ can be used to construct the abstraction, but this would give conservative results without any formal guarantee. We will demonstrate this observation on a case study in Section 3.7.*

Corollary 3.1 *The abstract model constructed using the growth bounds as solutions of SCP_γ with confidence $(1-\beta)$ for state-input pairs $(\hat{x},\hat{u}) \in \widehat{X} \times \widehat{U}$ is a valid abstract model for Σ with confidence at least $(1-n_x n_u \beta)$, where n_x and n_u are respectively the cardinality of $\widehat{X}$ and $\widehat{U}$.*

Proof 3.3 *Denote the optimal solution of SCP_γ in* (3.5.6) *by θ^*. The ball centered at $z(\hat{x},\hat{u}) := x'_{nom}$ with radius $\lambda(\hat{x},\hat{u}) = \kappa(\theta^*)(\eta_x,\hat{x},\hat{u}) + \gamma$ is a valid overapproximation of the reachable set from the state-input pair $(\hat{x},\hat{u})$ with confidence at least $1-\beta$. Since the number of pairs $(\hat{x},\hat{u})$ is $n_x n_u$, the chance of getting an invalid growth bound in at least one instance of SCP_γ is bounded by $n_x n_u \beta$. Therefore, we get a sound abstraction that truly overapproximates the behaviour of the system with confidence $(1-n_x n_u \beta)$.*

Remark 3.3 *The parameter $\varepsilon \in [0,1]$ gives a tradeoff between the required number of samples and the level of conservativeness applied to the SCP. Smaller ε results in a larger number of sample trajectories, but reduces the value of γ in* (3.5.7) *(less conservative constraints in the SCP and higher chance of finding a feasible solution). In contrast, larger ε results in a smaller number of sample trajectories but increases the value of γ.*

Remark 3.4 *The quantity* $2n$ *used in* (3.5.7) *is in fact the dimension of the sample space* $\mathscr{D} = \Omega_{\eta_x}(\hat{x}) \times W$. *If the system does not have any disturbance (i.e., the system can be modeled as an ODE having deterministic trajectories), the sample space will be* $\mathscr{D} = \Omega_{\eta_x}(\hat{x})$ *and its dimension n can be used in* (3.5.7)*:* $\gamma = 4L_{\varphi}(\hat{u}) \sqrt[n]{\varepsilon \prod_{i=1}^{n} \eta_x(i)}$. *This will substantially reduce the number of required sample trajectories. Similarly, if the disturbance does not affect some of the state equations,* $2n$ *can be replaced by* $(n+q)$ *where q is the dimension of the disturbance set considered as a non-zero measure set.*

Algorithm 1 uses the result of Corollary 3.1 to provide an algorithmic solution for Problem 3.2. This algorithm receives a confidence parameters β, divides it by the cardinality of $\widehat{X} \times \widehat{U}$ (i.e., $n_x n_u$), computes the growth bounds for each pair $(\hat{x}, \hat{u}) \in \widehat{X} \times \widehat{U}$ using the SCP$_\gamma$ in (3.5.6) with confidence $1 - \beta/(n_x n_u)$, and constructs the abstraction using these growth bounds.

Algorithm 1 Data-Driven Abstraction [78].

Data: (X, U, W) of a control system Σ, confidence β, discretisation parameters η_x, η_u

Compute the finite state and input sets $\widehat{X}$ and $\widehat{U}$ using η_x, η_u Define n_x and n_u as cardinalities of $\widehat{X}$ and $\widehat{U}$ Choose $\varepsilon \in [0,1]$ Set $N = N(\varepsilon, \frac{\beta}{n_x n_u})$ using Eq. (3.4.5) Compute γ using Eq. (3.5.7)

for $\hat{x} \in \widehat{X}$ **do**

  for $\hat{u} \in \widehat{U}$ **do**

    $\hat{f}(\hat{x}, \hat{u}) = \emptyset$ Consider the uncertainty space $\mathscr{D} = \Omega_{\eta_x}(\hat{x}) \times W$ Select N i.i.d sample trajectories using uniform distribution over $\mathscr{D}$ Simulate the nominal trajectory $(\hat{x}, \hat{u}, x'_{nom})$ Solve the SCP$_\gamma$ (3.5.6) to get the optimiser $\theta^*(\hat{x}, \hat{u})$ $z \leftarrow x'_{nom}$ $\lambda \leftarrow \kappa(\theta^*)(\eta_x, \hat{x}, \hat{u}) + \gamma$ Find all states $\hat{x}' \in \widehat{X}$ for which $\Omega_{\eta_x}(\hat{x}') \cap \Omega_\lambda(z) \neq \emptyset$ and add them to $\hat{f}(\hat{x}, \hat{u})$

  end

end

Result: $\widehat{\Sigma} = (\hat{X}, \hat{U}, \hat{f})$ as a finite abstraction of Σ with confidence $(1-\beta)$, $\theta^*(\hat{x}, \hat{u})$ as a growth bound for $\hat{x} \in \widehat{X}, \hat{u} \in \widehat{U}$

The finite abstraction $\widehat{\Sigma}$ constructed by Algorithm 1 is a valid abstraction for Σ with confidence $(1-\beta)$. This means any controller $\widehat{C}$ synthesised on $\widehat{\Sigma}$ and refined to a controller C for Σ will satisfy the desired specification with confidence $(1-\beta)$ on the closed loop system $\Sigma \parallel C$. In the next section, we extend our approach to make it suitable for abstraction refinement in case there is no controller $\widehat{C}$ satisfying the specification due to the conservatism of the approach.

3.5.3 Lipschitz constant estimation

For estimating the Lipschitz constant L_φ in (3.5.9), we estimate an upper bound for the fraction

$$\Delta(\hat{u}) := \frac{\|\varphi(x,\hat{u},w) - \varphi(x',\hat{u},w')\|}{\|(x,w)-(x',w')\|}$$

that holds for all $x,x' \in X$ and $w,w' \in W$. We follow the line of reasoning in [175, 174] and use the extreme value theory for the estimation.

Let us fix a $\delta > 0$ and assign uniform distribution to the pairs (x,w) and (x',w') over the domain

$$\{x,x' \in X,\, w,w' \in W \text{ with } \|(x,w)-(x',w')\| \leq \delta\}. \tag{3.5.10}$$

Then $\Delta(\hat{u})$ is a random variable with an unknown cumulative distribution function (CDF). Based on the assumption of Lipschitz continuity of the system, the support of the distribution of $\Delta(\hat{u})$ is bounded from above, and we want to estimate an upper bound for its support. We take $\mathfrak{n}$ sample pairs (x,w) and (x',w'), and compute $\mathfrak{n}$ samples $\Delta_1,\Delta_2,\ldots,\Delta_\mathfrak{n}$ for $\Delta(\hat{u})$. The CDF of $\max\{\Delta_1,\Delta_2,\ldots,\Delta_\mathfrak{n}\}$ is called the limit distribution of $\Delta(\hat{u})$. Fisher-Tippett-Gnedenko theorem says that if the limit distribution exists, it can only be one of the three family of extreme value distributions – the Gumbel class, the Fréchet class, and the reverse Weibull class. These CDF's have the following forms:

$$\text{Gumbel class:} \quad G(s) = \exp\left[-\exp\left[\frac{s-a}{b}\right]\right], s \in \mathbb{R}$$

$$\text{Fréchet class:} \quad G(s) = \begin{cases} 0 & \text{if } s < a \\ \exp\left[-\left[\frac{s-a}{b}\right]^{-c}\right] & \text{if } s \leq a \end{cases}$$

$$\text{Reverse Weibull class: } G(s) = \begin{cases} \exp\left[-\left[\frac{a-s}{b}\right]^{c}\right] & \text{if } s < a \\ 1 & \text{if } s \leq a \end{cases}$$

where $a \in \mathbb{R}, b > 0, c > 0$ are respectively the location, scale and shape parameters of the distributions.

Among the above three distributions, only the reverse Weibull class has a support bounded from above. Therefore, the limit distribution of $\Delta(\hat{u})$ will be from this class and the location parameter a is such an upper bound. As a result, we can estimate the location parameter of the limit distribution of $\Delta(\hat{u})$ to get an estimation of the Lipschitz constant.

The approach is summarised in Algorithm 2. The most inner loop computes samples of $\Delta(\hat{u})$. The middle loop computes samples of $\max\{\Delta_1,\ldots,\Delta_\mathfrak{n}\}$. The outer loop estimates the Lipschitz constant for each $\hat{u}$ by fitting a reverse Weibull distribution.

Algorithm 2 Lipschitz Constant Estimation [78].

Data: (X,U,W) of a control system Σ, abstract input space $\widehat{U}$
Select number of samples $\mathfrak{n}$ and $\mathfrak{m}$ for the estimation
Select $\delta > 0$
for $\hat{u} \in \widehat{U}$ **do**
  for $j = 1 : \mathfrak{m}$ **do**
    for $i = 1 : \mathfrak{n}$ **do**
      Sample pairs $(x,w),(x',w')$ uniformly from the domain in (3.5.10)
      Run Σ to get trajectories $\varphi(x,\hat{u},w)$ and $\varphi(x',\hat{u},w')$
      Compute $\Delta_i := \frac{\|\varphi(x,\hat{u},w)-\varphi(x',\hat{u},w')\|}{\|(x,w)-(x',w')\|}$
    end
    $\Gamma_j := \max\{\Delta_1,\ldots,\Delta_{\mathfrak{n}}\}$
  end
  Fit a reverse Weibull distribution to the sample set $\{\Gamma_1,\Gamma_2,\ldots,\Gamma_{\mathfrak{m}}\}$
  $L_\varphi(\hat{u})$ is the location parameter of the fitted distribution
end
Result: Estimated value of $L_\varphi(\hat{u})$ for all $\hat{u} \in \widehat{U}$

3.6 Synthesis via abstraction refinement

The data-driven synthesis discussed in Section 3.5 inherits the soundness property from the ABCD approach: they both work with overapproximations of the dynamics and may not return a controller despite one may exists. Therefore, there is a need for refining the abstraction in order to check for controllers using less conservative abstractions. While the method of Section 3.5 is good for a given fixed discretisation parameter η_x, it is not suitable for reducing η_x, which requires re-computing all local parameters of the growth bounds $\theta_1(\hat{x},\hat{u}),\theta_2(\hat{x},\hat{u})$. Another shortcoming of the method is related to the data collection: the nominal trajectories of the system should be available and are used in the constraints of the SCP. In this section, we discuss an extension of the approach of Section 3.5, in order to

– enable reducing η_x without the need for re-computing the growth bound, and

– relax the assumption of having access to the nominal trajectories of the system.

Let us define a modified growth bound as a function $\kappa_e\colon \mathbb{R}^n_{\geq 0}\times\widehat{X}\times\widehat{U}\to\mathbb{R}^n_{\geq 0}$ that is strictly increasing in its first argument and satisfies

$$\begin{aligned}&|\varphi(x_1,\hat{u},w_1)-\varphi(x_2,\hat{u},w_2)| \leq \kappa_e(|x_1-x_2|,\hat{x},\hat{u})\\ &\forall\hat{x}\in\widehat{X},\,\forall\hat{u}\in\widehat{U},\,\forall x_1,x_2\in\Omega_{\eta_x}(\hat{x}),\,\forall w_1,w_2\in W.\end{aligned} \tag{3.6.1}$$

This definition is more conservative than (3.5.1) in comparing trajectories under two arbitrary disturbances, and we always have that κ_e satisfies (3.5.1). Using this new definition, for every pair of abstract state and input $(\hat{x},\hat{u})$, the corresponding overapproximation of the reach set can be computed as a ball centred at *any* $z(\hat{x},\hat{u}) \in \Phi(\hat{x},\hat{u})$ with radius $\lambda(\hat{x},\hat{u}) = \kappa_e(\eta_x,\hat{x},\hat{u})$.

we choose a parametrisation for κ_e similar to (3.5.4), i.e.,

$$\kappa_e(\theta)(r,\hat{x},\hat{u}) = \theta_1(\hat{x},\hat{u})r + \theta_2(\hat{x},\hat{u}), \tag{3.6.2}$$

where $r \in \mathbb{R}_{\geq 0}$, $\theta_1 \in \mathbb{R}^{n\times n}$, $\theta_2 \in \mathbb{R}^n$, and $\theta \in \mathbb{R}^{n^2+n}$ is constructed by concatenating columns of θ_1 and θ_2. The SCP associated with this growth bound is constructed by considering a uniform distribution over $\Omega_{\eta_x}(\hat{x}) \times W$ and obtain $2N$ i.i.d. sample trajectories $\mathscr{S}_{2N} = \{(x_i,\hat{u}_i,x_i') \,|\, x_i' \in \Phi(x_i,\hat{u}), i = 1,2,\ldots,2N\}$ so that every x_i' corresponds to a random disturbance $w_i \in W$. The modified SCP_γ is defined as

$$\begin{cases} \min c^\top \theta \\ s.t.\ 0 \leq \theta \leq \bar{\theta} \text{ and } \forall i \in \{1,\ldots,N\} \\ |x_{2i-1}' - x_{2i}'| - \theta_1(\hat{x},\hat{u})|x_{2i-1} - x_{2i}| - \theta_2(\hat{x},\hat{u}) + \gamma \leq 0 \end{cases}$$

where $c = [1,1,\ldots,1] \in \mathbb{R}^{n^2+n}$ is a constant vector, $\bar{\theta} \in \mathbb{R}^{n^2+n}_{>0}$ is sufficiently large, and $\gamma \geq 0$.

Theorem 3.4 *For any $\hat{x} \in \widehat{X}$ constructed with the discretisation size η_x, any $\hat{u} \in \widehat{U}$, and the disturbance set $W = [-\bar{w},\bar{w}]$, the optimal solution of* (3.6.3) *gives a growth bound for the system Σ corresponding to $(\hat{x},\hat{u})$ that satisfies* (3.6.1) *with confidence $(1-\beta)$, when the number of samples $2N \geq N(\varepsilon,\beta)$ and*

$$\gamma = 8L_\varphi \sqrt[4n]{\varepsilon \left[\prod_{i=1}^{n} \eta_x(i) \prod_{i=1}^{n} \bar{w}(i)\right]^2}, \tag{3.6.3}$$

where $\varepsilon \in [0,1]$, ,n is the dimension of the state space, and $L_\varphi(\hat{u})$ is the Lipschitz constant of the system trajectories $\varphi(x,\hat{u},w)$ with respect to (x,w).

Proof 3.4 *The proof of this theorem is similar to that of Theorem 3.3. Define*

$$\begin{aligned} g(\theta,x_1,w_1,x_2,w_2) := \max\{&|\varphi(x_1,\hat{u},w_1) - \varphi(x_2,\hat{u},w_2)| \\ &- \theta_1(\hat{x},\hat{u})|x_1 - x_2| - \theta_2(\hat{x},\hat{u})\}. \end{aligned}$$

To satisfy the inequality (3.4.3)*, we can choose*

$$h(\varepsilon) = \mathbb{P}(\Omega_\varepsilon(d)) = \frac{(\varepsilon/2)^{4n}}{[\prod_{i=1}^{n} \eta_x(i) \prod_{i=1}^{n} \bar{w}(i)]^2},$$

since the distribution on $(\Omega_{\eta_x}(\hat{x}) \times W)^2$ *is uniform. Using Equation* (3.4.4)*, we have* $\gamma = L_d h^{-1}(\varepsilon)$*. In order to prove that* γ *takes the value in* (3.6.3)*, we must show that g is Lipschitz continuous with constant* $L_d = 4L_\varphi(\hat{u})$*. Bounding* $\|\theta_1(\hat{x},\hat{u})\|$ *by* L_φ*, for all* (x_1, w_1, x_2, w_2) *and* (x_1', w_1', x_2', w_2') *we have*

$$\begin{aligned}
&\|g(\theta, x_1, w_1, x_2, w_2) - g(\theta, x_1', w_1', x_2', w_2')\| \\
&\leq \|\varphi(x_1, \hat{u}, w_1) - \varphi(x_1', \hat{u}, w_1')\| \\
&+ \|\varphi(x_2, \hat{u}, w_2) - \varphi(x_2', \hat{u}, w_2')\| \\
&+ \|\theta_1(\hat{x},\hat{u})\|(\|x_1 - x_1'\| + \|x_2 - x_2'\|) \\
&\leq 4L_\varphi(\hat{u})\|(x_1, w_1, x_2, w_2) - (x_1', w_1', x_2', w_2')\|.
\end{aligned}$$

Therefore, g is Lipschitz continuous with constant $4L_\varphi(\hat{u})$*. This completes the proof.*

A statement similar to Corollary 3.1 holds for the growth bound computed using (3.6.3).

3.7 Experimental evaluation

To demonstrate our approach, we apply it to a DC-DC boost converter and a path planning problem. These case studies are taken from [129, 50] and will be used as black-box models to generate sample trajectories. We also introduce a case study from power systems based on [99], that is implemented in the Power System Toolbox (PST) [26]. We will will use trajectories from the black-box reduced model of the 30 state power system model. We apply our approach to construct finite abstractions of these systems and employ SCOTS [129] to design controllers. Our algorithms are implemented in C++ on a 64-bit Linux cluster machine with two Intel Xeon E5 v2 CPUs, 1866 MHz, and 50GB RAM.

3.7.1 DC-DC boost converter

The objective in the DC-DC boost converter problem is to design a controller to enforce a reach and stay specification. The DC-DC boost converter can be modelled as a two dimensional linear switching system with two functional modes. The state vector of the system at time $t \in \mathbb{R}_{\geq 0}$ is $x(t) = (i_l(t), v_c(t))$, where i_l is the inductor current and v_c is the

Table 3.1 Results for the DC-DC boost converter [78].

Case-study	Dimension		Disturbance	Fixed Discretisation		
	X	U	W	N	time (min)	$\lvert\mathscr{V}\rvert$
DC-DC boost converter	2	1	$\{0\}$	1,807	22.2	37,783
			$[-0.01, 0.01]$	2,285	30.6	37,414

capacitor voltage. The system's evolution can be controlled by selecting the appropriate mode $u(t) \in \{1,2\}$ at every time $t \in \mathbb{R}_{\geq 0}$. The system's dynamics under the two modes can be represented as $\dot{x} = A_{u(t)}x(t) + b + cw(t)$, $u \in \{1,2\}$, with matrices A_1, A_2, b, c as reported in [50]. The state and input spaces are $X = [0.65, 1.65] \times [4.95, 5.95]$ and $U = [1,2]$. The initial state is $(i_{l_0}(t), v_{c_0}(t)) = (0.7, 5.4)$ and the target set is $[1.1, 1.6] \times [5.4, 5.9]$. The target set is shown in red colour in Figure 3.1.

Our implementation results are reported in Table 3.1 for the system without disturbance ($\bar{w} = (0,0)$) and with disturbance bound $\bar{w} = (0.01, 0)$. These results are obtained with discretisation parameters $\eta_x = (0.005, 0.005)$ and $\eta_u = 1$, confidence parameter $\beta = 0.01$, $\varepsilon = 0.01$ and estimation for $L_\varphi = 0.9935$. The resulted finite abstraction has cadinalities $n_x = 40,000$ and $n_u = 2$. The required number of sample trajectories, N, for each $(\hat{x}, \hat{u}) \in \widehat{X} \times \widehat{U}$ is computed using equation (3.4.5). Runtimes and the resulting winning region sizes, $|\mathscr{V}|$, for the DC-DC boost converter are given in Table 3.1.

We have used Algorithm 1 to compute the finite-state abstraction by collecting sample trajectories of the system. Subsequently, SCOTS is used for designing the controller. The performance of the controller is shown in Figures 3.1 and 3.2 for the system without and with the disturbance. These figures show one sample closed-loop trajectory of the system under the controllers designed by our data-driven ABCD approach. In both cases, without and with disturbance, it can be noticed from Figures 3.1 and 3.2 that our approach has been successful in finding controllers satisfying the given reach and stay specification, despite the the dynamics being unknown.

3.7.2 Path planning problem with partition refinement

We consider a path planning problem for a vehicle that is modelled as

$$\begin{aligned} \dot{x} &= v\cos(\alpha+\theta)/\cos(\alpha) + w \\ \dot{y} &= v\sin(\alpha+\theta)/\cos(\alpha) \\ \dot{\theta} &= v\tan(\omega), \end{aligned} \tag{3.7.1}$$

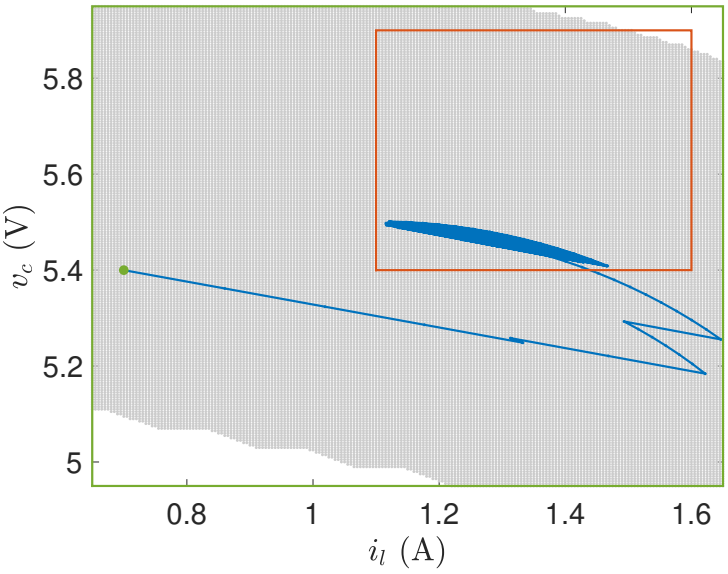


Fig. 3.1 The closed-loop trajectory of the DC-DC boost converter with $\bar{w} = (0,0)$ under the controller designed by our data-driven abstraction approach. The rectangle in red colour represents the target region and the area in grey shows the winning region of the controller [78].

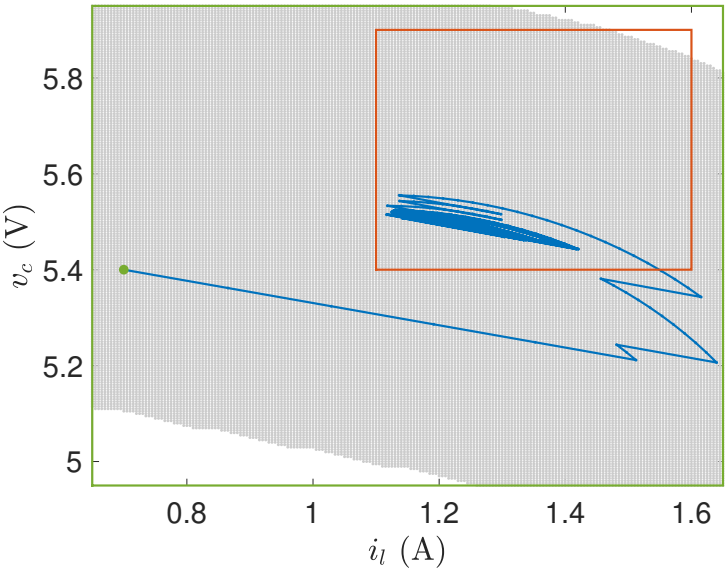


Fig. 3.2 The closed-loop trajectory of the DC-DC boost converter with $\bar{w} = (0.01,0)$ under the controller designed by our data-driven abstraction approach. The rectangle in red colour represents the target region and the area in grey shows the winning region of the controller [78].

Table 3.2 Results for the path planning case study [78].

Case-study	Dimension		Disturbance	Abstraction Refinement		
	X	U	$\bar{w}$	N	time (min)	$\lvert\mathscr{V}\rvert$
Path planning	3	2	$(0,0,0)$	$3,127$	225	$405,493$
			$(0.01,0,0)$	$4,277$	513	$447,212$

where the state variables x, y, θ represent the position of the vehicle in the 2-dimensional space and the orientation of the vehicle, respectively. Inputs are (v, ω), the disturbance is w, and $\alpha := \arctan(\tan(\omega)/2)$. The state and input spaces are $X = [0,10] \times [0,10] \times [-\pi - 0.4, \pi + 0.4]$ and $U = [-1,1]^2$, respectively. The goal is to find a controller to steer the vehicle from the initial state $(x_0, y_0, \theta_0) = (0, 1.2, 0)$ to the target set $(x, y) \in [9, 9.51] \times [0, 0.51]$ while avoiding the obstacles. These obstacles are shown in blue colour in Figures 3.3 and 3.4.

We computed the growth bounds with a coarse discretisation $\eta_x = (1.6, 1.6, 1.6)$ and reduced it iteratively with the factor of two. The algorithm successfully finds a controller for the system after five iterations. The implementation results are reported in Table 3.2. These results are obtained with $\eta_u = (0.3, 0.3)$, the confidence parameter $\beta = 0.01$, $\varepsilon = 0.01$ and estimated constant $L_\varphi = 1.46$. The resulted abstraction has cardinalities $n_x = 88,500$ and $n_u = 24$. For the case of disturbance-free model we set $\bar{w} = (0,0,0)$, and for the case of dynamics with disturbance, we set $\bar{w} = (0.01, 0, 0)$. The required number of sample trajectories for each $(\hat{x}, \hat{u})$ is computed using Equation (3.4.5) and marked with N in the table. Finally, runtimes and size of the winning regions $\lvert\mathscr{V}\rvert$ are reported.

We have used the synthesis method based on abstraction refinement presented in Section 3.6, to construct the finite-state abstraction by collecting sample trajectories of the system. We used SCOTS to design the controller fulfilling the given specification. The performance of the controller is shown in Figures 3.3 and 3.4 for the system without and with the disturbance, respectively. These figures compare the closed-loop trajectories of the system under the controllers designed by our data-driven abstraction refinement algorithm approach (black) and by the model-based approach of SCOTS (red). Our data-driven approach successfully finds a controller for the system that satisfies the specification without the need for knowing the dynamics of the system.

3.7.3 Three area three machine power system

We consider a three area three machine (3A3M) power system adapted from [99] and is shown in Figure 3.5. The system consists of three buses, which are each connected to a power

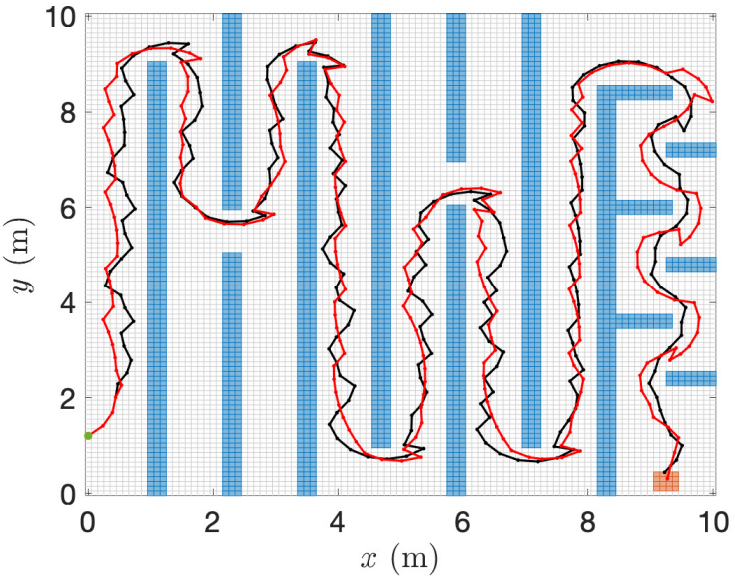


Fig. 3.3 Comparison between the closed-loop trajectories of the system (3.7.1)without disturbance under the controllers designed by our data-driven abstraction refinement approach (black) and by the model-based approach of SCOTS (red). Blue blocks represent the obstacles, the green dot represents the initial state, and the orange rectangle shows the target region [78].

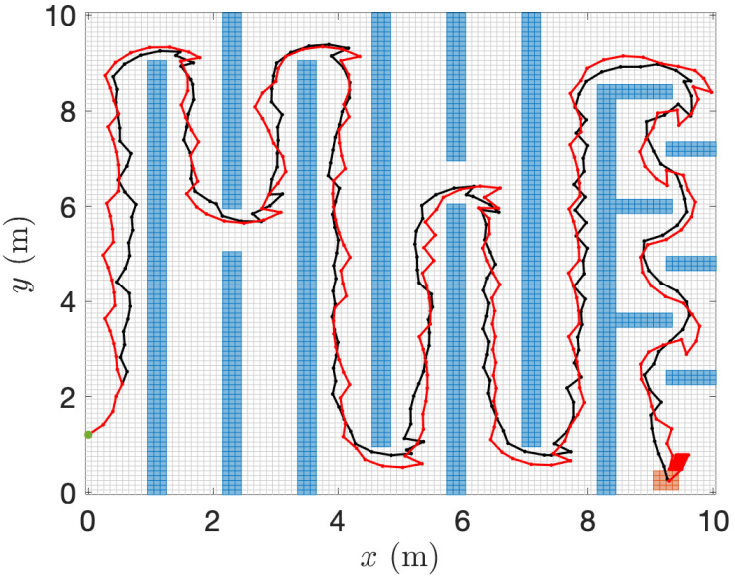


Fig. 3.4 Comparison between the closed-loop trajectories of the system (3.7.1) with disturbance bound $\bar{w} = (0.01, 0, 0)$ under the controllers designed by our data-driven abstraction refinement approach (black) and by the model-based approach of SCOTS (red) [78].

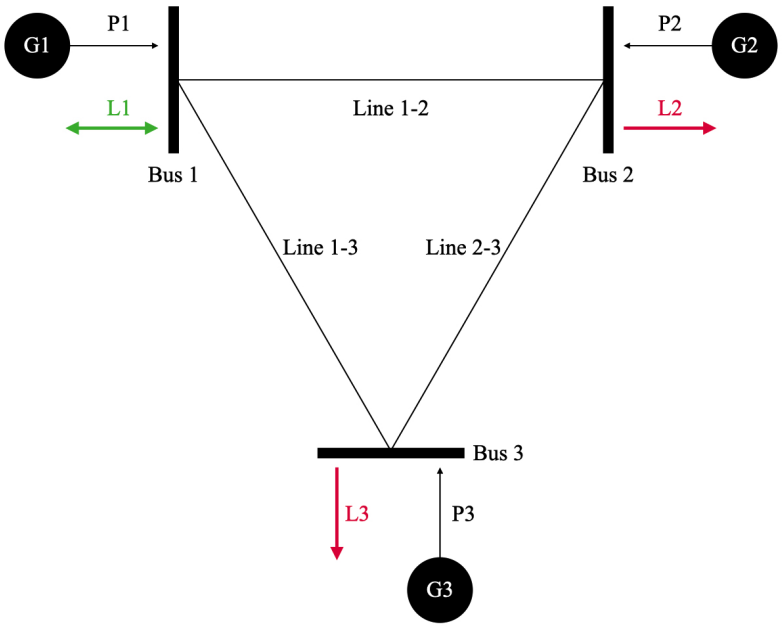


Fig. 3.5 3A3M power system with generators (G) and loads (L). L1 represents a bidirectional load such as Electric Vehicles or Energy Storage Systems [78].

source (generator) and a load. At bus 1 we consider a load which is bidirectional, meaning it can both draw power and inject power into the system. The loads at buses 2 and 3 can only draw power from the system; when these loads increase, more power will be drawn from the system, causing an imbalance between generation and consumption which may result in reduction of the network frequency. The nominal frequency of the network is set to 60 Hz.

We consider a worst case scenario when a sudden increase occurs in the loads at buses 2 and 3 by 0.2 and 0.3 per unit (pu), respectively. The control task is for the load at bus 1 to balance the load increase at buses 2 and 3 by either reducing its load or injecting power into the network. The simulation is run using PST on a 30 state model of this power system. Balanced realisation of the system reduces its dynamics to three states. To compute the data-driven finite abstraction, sample trajectories are gathered using a black-box approach of the reduced system representation for the original model. The dynamics of the reduced system are given by

$$\begin{aligned} \dot{x} &= Ax + Bu + Ew \\ y &= Cx, \end{aligned} \tag{3.7.2}$$

where

$$A = \begin{bmatrix} 0.00027563 & 0 & 0 \\ 0 & -0.3951 & 0.687 \\ 0 & -0.6869 & -0.016 \end{bmatrix}$$

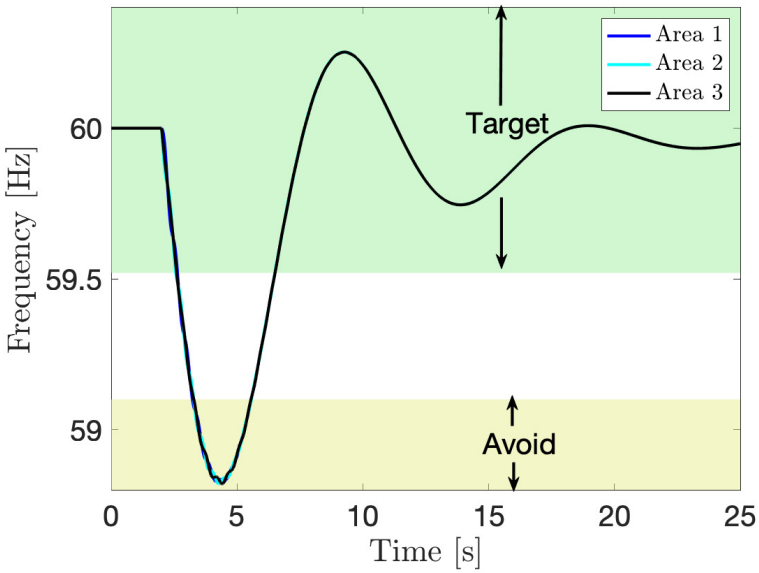


Fig. 3.6 3A3M power system frequency without applying any control input. The frequency falls below 59.1 Hz thus violates the specification [78].

$$B = \begin{bmatrix} 0.00031166 \\ 0.1359 \\ 0.0230 \end{bmatrix}$$

$$E = \begin{bmatrix} 0.00033103 & 0.00031244 \\ 0.1309 & 0.1308 \\ 0.0250 & 0.0233 \end{bmatrix}$$

$$C = \begin{bmatrix} -0.0115 & -0.2296 & 0.0412 \end{bmatrix}. \tag{3.7.3}$$

The state and input spaces are $X = [-0.02, 0.02] \times [-0.05, 0.05] \times [-0.12, 0.12]$ and $U = [0, 0.5]$. Further, we set $W = [-0.2, 0.2] \times [-0.3, 0.3]$, $\eta_u = 0.025$, $\tau = 0.4$, $\eta_x = (0.0015, 0.0015, 0.0015)$, $\beta = 0.01$ and $\varepsilon = 0.01$. The resulted abstraction has $n_x = 228,480$ and $n_u = 20$. The estimated Lipschitz constant is $L_\varphi = 1.5715$. The target set is given by $-0.008 < y < 0.008$ and the avoid set is given by $y < -0.015$. Multiplying by the nominal frequency to get the specification in Hertz, the target region is $[59.52, 60.48]$ and the avoid region is $(-\infty, 59.1)$. Figure 3.6 shows that the specification is violated when no control is applied.

We apply the data-driven approaches of Section 3.5 (fixed discretisation) and Section 3.6 (abstraction refinement). Both controllers are synthesised with disturbance $W = [-0.2, 0.2] \times$

$[-0.3, 0.3]$. A comparison of the two control approaches is shown in Table 3.3. The required number of sample trajectories for each $(\hat{x}, \hat{u})$ is computed using equation (3.4.5) and marked with N in the table. The abstraction refinement starts with $\eta_x = 0.012$ and refines the discretisation iteratively with a factor of two. The algorithm successfully finds a controller after five iterations. The runtimes and the resulting winning region sizes $|\mathscr{V}|$ are also given in Table 3.3. The abstraction refinement synthesises the controller a factor of 100 times faster than the fixed discretisation by iteratively decreasing the value of η_x.

Table 3.3 Results for the 3A3M power system [78].

Control Approach	Dimension		Disturbance			
	X	U	$\bar{w}$	N	time (min)	$\lvert\mathscr{V}\rvert$
Fixed Discretisation	3	1	$(0.2, 0.3)$	$3,290$	$5,253$	$230,760$
Adaptive Refinement			$(0.2, 0.3)$	$4,460$	50.25	$314,802$

The data-driven control approach with fixed discretisation is simulated in PST and is reported in Figures 3.7 and 3.8. The controlled system successfully keeps the frequencies of the three areas outside of the avoid set (i.e., always above 59.1 Hz) and bring them back to the target set (i.e., above 59.52 Hz). Figure 3.8 shows the load changes in the system. Load at bus 1 is able to maintain the frequencies of the three areas above the avoid region and facilitate the system returning to the target set for the maximum disturbances applied at buses $2, 3$. Figures 3.9 and 3.10 show the results of simulating the system in PST with the control obtained from the abstraction refinement approach. The controlled system has the same performance in satisfying the specification.

3.7.4 Comparison with PAC learning

In this subsection, we compare our approach with the results provided by Xue et al. [179] that is based on probably approximately correct (PAC) learning on the 3A3M power system case study. The abstraction approach of [179] has no bias term γ, but uses confidence parameter $\beta \in (0, 1)$, error level $\varepsilon \in (0, 1)$, and cardinality of the parameter vector θ denoted by $q \in \mathbb{N}$. The required number of samples is

$$N \geq \frac{2}{\varepsilon}(ln\frac{1}{\beta} + q), \tag{3.7.4}$$

which allows the constructed abstraction to hold for the entire state space except a subset measured by parameter ε.

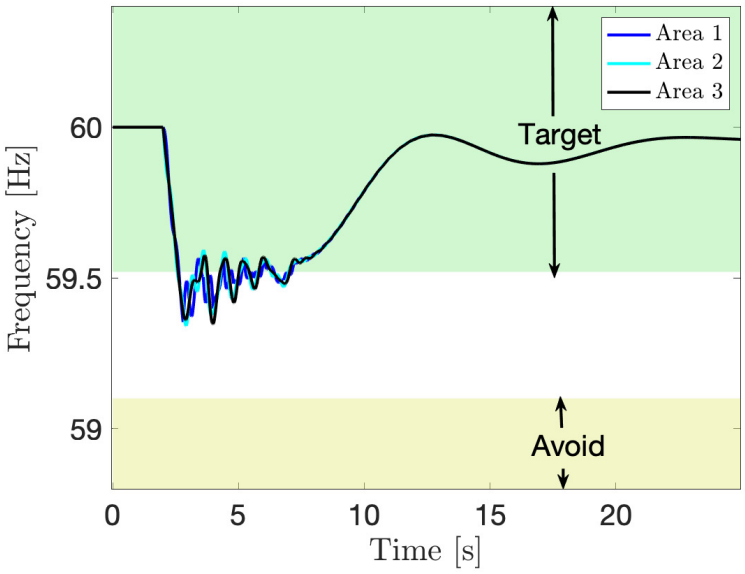


Fig. 3.7 3A3M power system frequencies for the three areas, with the frequency of an area is measured at the corresponding bus in that area. The control synthesised by the fixed discretisation approach successfully keeps the frequencies of the three areas outside of the avoid set. The frequencies leave the target set for around 4.4 seconds before staying in the target set [78].

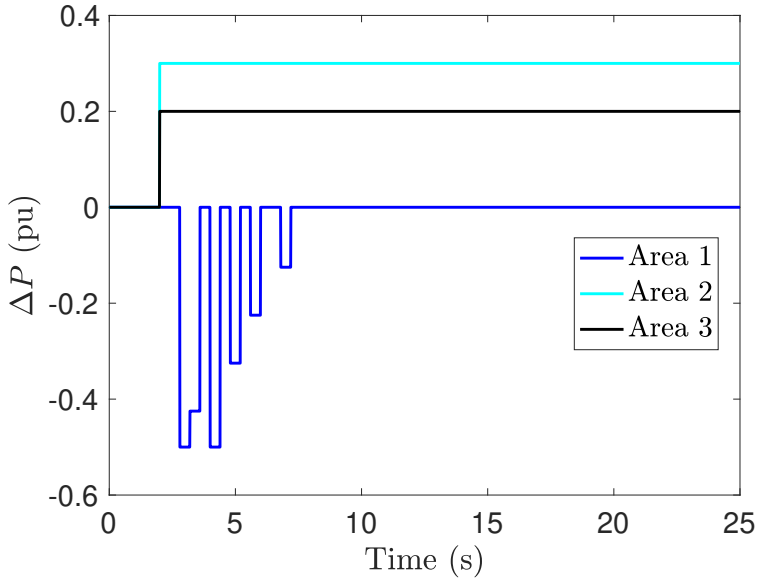


Fig. 3.8 3A3M power system load changes for the three areas. Loads at buses 2 and 3 increase by 0.3 and 0.2 pu, respectively. Load at bus 1 is used to control the frequency using our data-driven approach with fixed discretisation [78].

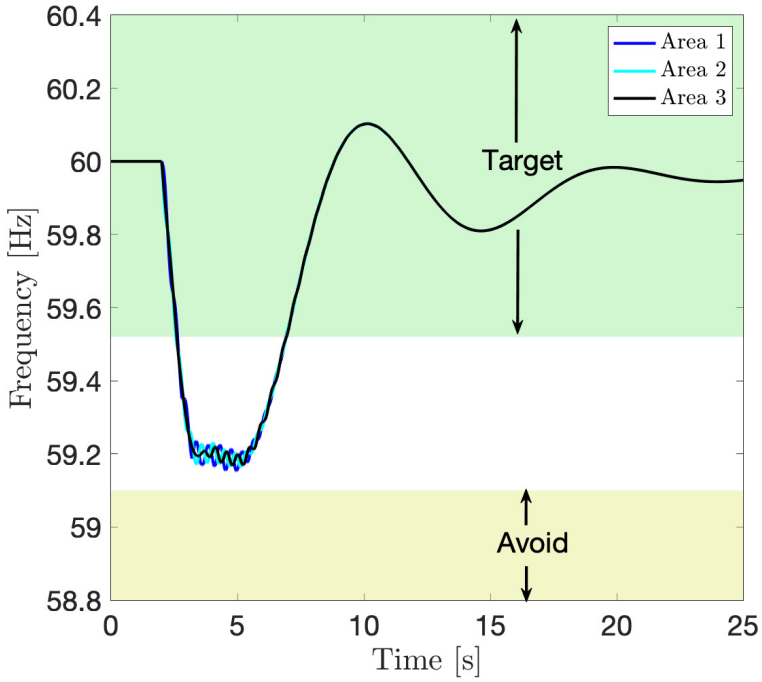


Fig. 3.9 3A3M power system frequencies for the three areas, with the frequency of an area is measured at the corresponding bus in that area. The control synthesised by the abstraction refinement approach successfully satisfies the specification. The frequencies leave the target set for around 4.2 seconds before staying in the target set [78].

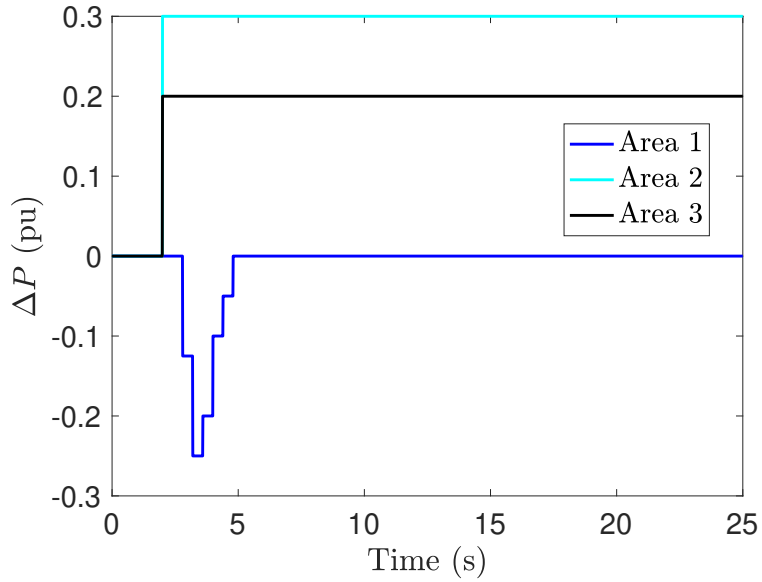


Fig. 3.10 3A3M power system load changes for the three areas. Loads at buses 2 and 3 increase by 0.3 and 0.2 pu, respectively. Load at bus 1 is used to control the frequency using our data-driven approach with abstraction refinement [78].

Table 3.4 Comparing the winning domain of controllers obtained from our RSA method, PAC method of [179], and the model-based approach of [127]. The pairwise comparison is made by computing the intersections ($\cap$) and set differences (row $\setminus$ column). The results are reported both in cardinalities and percentages [78].

Winning Domain	RSA		PAC		Model-based	
	$\cap$	$\setminus$	$\cap$	$\setminus$	$\cap$	$\setminus$
RSA	230,760	0	230,760	0	230,760	0
%	100.00%	0.00%	100.00%	0.00%	100.00%	0.00%
PAC	230,760	15,664	246,424	0	245,345	1,079
%	93.64%	6.36%	100.00%	0.00%	99.56%	0.44%
Model-based	230,760	22,216	245,345	7,631	252,976	0
%	91.22%	8.78%	96.98%	3.02%	100.00%	0.00%

We implement our data-driven robust scenario approach (RSA), the PAC approach in [179] with parameters $\beta = 0.01$ and $\varepsilon = 0.01$, and the model-based approach of [127]. Table 3.4 compares the winning domain of the controllers by reporting the intersections ($\cap$) and set differences (row $\setminus$ column). It can be seen that the winning domain obtained by our RSA method is a subset of the ones computed by PAC and the model-based approaches. This shows that our approach is more conservative than the model-based approach but correctly finds a subset of the winning domain. In contract, the PAC approach gives a winning domain that includes states not identified winning by the model-based approach. It includes 1079 states outside of the winning domain obtained by the model-based approach. Due to the nature of the PAC learning, some of these states are incorrectly identified as winning. The main reason is that the PAC method may miss to capture some of the transitions and does not always generate an overapproximation of the system behaviours. Among these 1079 states, a counter example can be found, demonstrating a lack of guarantee provided by the PAC method. At state $(0.0187, 0.0262, -0.1163)$ the PAC controller calculates $u = -0.075$ to be an input which will transition to a safe state under any disturbances. However, the system under disturbances $W_1 = 0.2$ and $W_2 = 0.3$ will lead to the state $(0.0188, 0.0131, -0.1167)$ that is outside of the winning domain of the controller. In comparison, the winning domain provided by our RSA method is a subset of the one from the model-based method and provides full guarantees on the satisfaction of the specification and correctness of the controller. This guarantee is obtained at the cost of increased number of samples and a bias term included in the growth bound calculations, which makes the controller more conservative.

As a final point on this case study, note that our sampling approach uses the Lipschitz constant estimated using sample trajectories. This Lipschitz constant can in turn be used to construct the abstraction. The direct use of the estimated Lipschitz constant does not provide a formal guarantee as it is an estimated value that converges to the true value only in the limit (i.e., the number of samples goes to infinity), and is likely to provide an overly conservative controller. On this particular case study, the direct use of the Lipschitz constant gives a controller that covers only 78.8% of the winning domain of the model-based approach.

3.7.5 Parameter optimisation

In this subsection, we discuss how selection of different parameters can affect the sample complexity and conservativeness of our method. We fix the path planning case study with the estimated Lipschitz constant 1.46. Figures 3.11 and 3.12 illustrate the effect of changing parameters ε, β on the number of samples N required for each pair $(\hat{x}, \hat{u})$ in order to compute the growth bound with confidence $(1-\beta)$. Figure 3.11 illustrates the effect of increasing the confidence parameter β on reducing the sample complexity, for a fixed $\varepsilon = 0.01$. Figure 3.12 shows that for a fixed $\beta = 0.01$, increasing ε leads to a rapid drop in N. In both Figures 3.11 and 3.12, the sample complexity increases in the presence of disturbance as the dimension of the sample space becomes larger.

Figure 3.13 demonstrate the effect of changing ε on the value of the bias term γ that makes the inequalities of the SCP more conservative. The bias term γ increases for larger values of ε. Therefore, increasing ε can decrease the sample complexity while increasing γ. Finally, it can be observed that the value of γ is larger in the presence of disturbance.

3.8 Discussion and future work

We proposed a data-driven method for computing finite abstractions of continuous systems with unknown dynamics. Our approach casts the computation of an overapproximation of reachable sets as a robust convex program (RCP). A feasible solution for the RCP is then obtained with a given confidence by solving a corresponding scenario convex program (SCP). The SCP does not need the dynamics of the system and requires only a finite set of sample trajectories. We provided a sample complexity result that gives a lower-bound on the number of trajectories to achieve a certain confidence. Our sample complexity results requires knowing a bound on the Lipschitz constant of the system, that we estimated using extreme value theory.

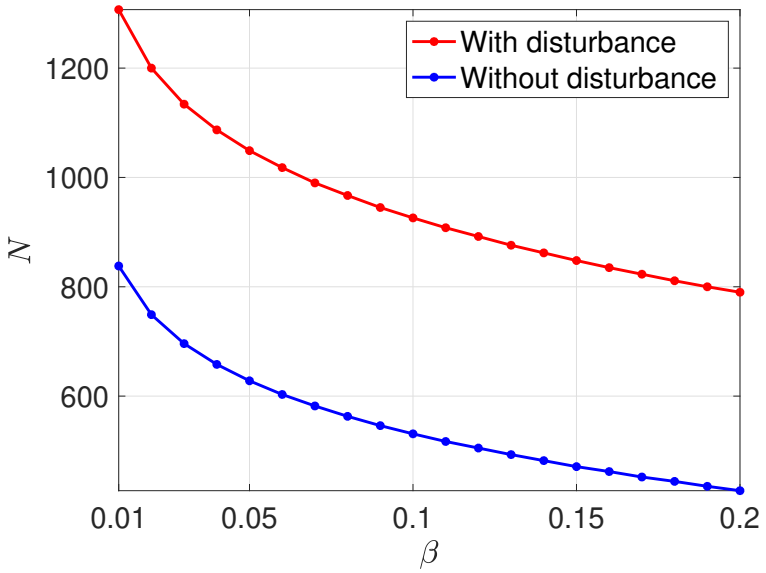


Fig. 3.11 Required number of samples for our approach as a function of β for a fixed $\varepsilon = 0.01$ [78].

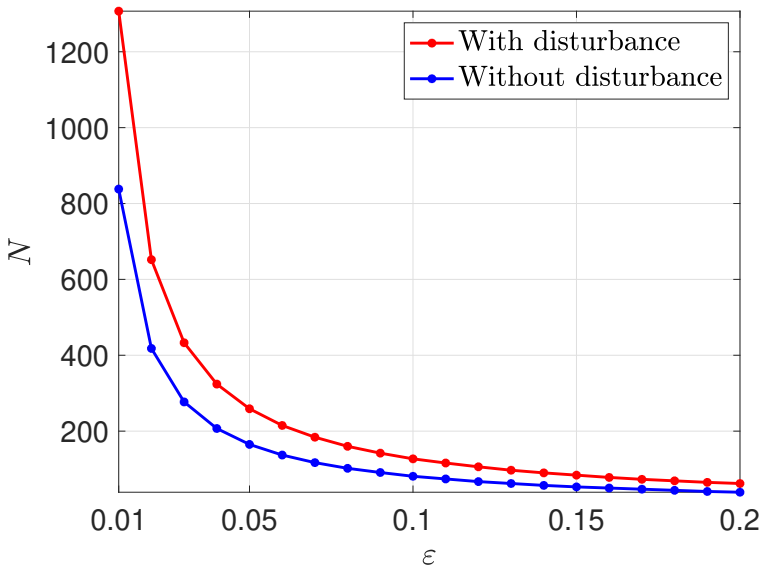


Fig. 3.12 Required number of samples for our approach as a function of ε for a fixed $\beta = 0.01$ [78].

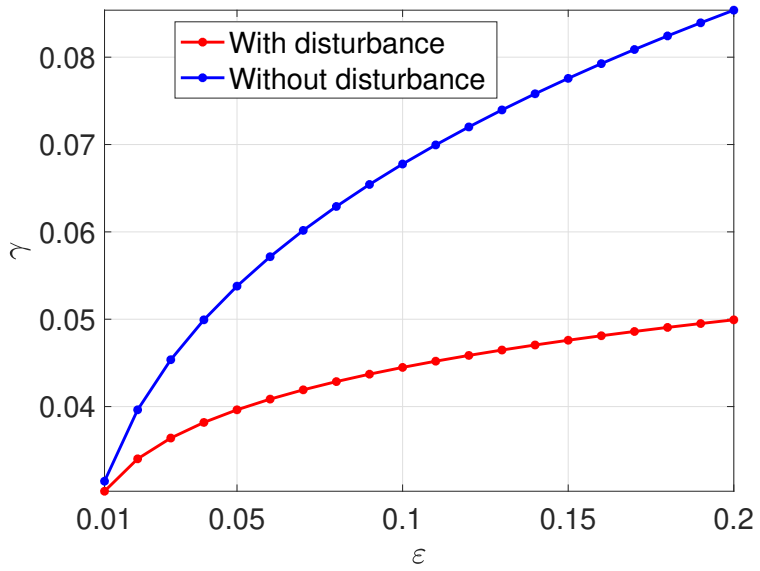


Fig. 3.13 The bias term γ as a function of ε [78].

We guaranteed that with high confidence, the computed abstraction is a valid abstraction of the system that overapproximates its behaviours on its entire state space. We showed that our data-driven approach can be embedded into abstraction refinement schemes for designing a controller and enlarging the winning region of the controller with respect to satisfaction of temporal properties. Finally, we evaluated our approach on three case studies.

In the future, we plan to extend our approach by enlarging the class of disturbances beyond piece-wise constant ones (i.e., tackling the issue of infinite dimensional sampling spaces), improve scalability of the approach by providing more efficient parallel implementation of the approach, and apply it to large case studies that are combinations of differential equations, block diagrams, and lookup tables.

Chapter 4

Model-free RL for formal control of stochastic systems

4.1 Chapter introduction

We discussed in Chapter 3 that using abstraction approaches is computationally expensive. In this chapter, the main focus is on tackling scalability issues. In this way, we introduce a game theoretic framework in which we decompose the system and restate the problem as a two-player zero-sum game. Though this approach is conservative, it solves the scalability issue of abstraction-based approaches.

This chapter introduces a novel reinforcement learning (RL) scheme to synthesise policies for continuous-space stochastic control systems with unknown dynamics. The proposed framework is based on *implicitly* abstracting the system with a finite Markov decision process (MDP) with *unknown* transition probabilities, synthesising a strategy for the abstract model, and then mapping the results back over the continuous-space system while providing *approximate optimality* guarantees for the synthesised strategy. We consider *finite-horizon* properties, expressed in the syntactically co-safe fragment of linear temporal logic (scLTL) augmented with a time bound. A key contribution is to leverage the classical convergence results for RL on finite MDPs to provide strategies that maximise the probability of satisfaction over unknown continuous-space systems. Since automata-based reward functions are often sparse, we also present a novel potential-based *reward shaping* technique to accelerate learning by producing dense rewards. We then extend our approach to *networks* of unknown stochastic control systems. The proposed *compositional* framework applies model-free *two-player* RL in an assume-guarantee fashion and *compositionally* compute strategies for continuous-space interconnected systems without explicitly constructing their finite-state abstractions. Our

approach gives a guaranteed lower bound for the probability of property satisfaction by the interconnected system based on those of individual controllers over subsystems. The effectiveness of the proposed approaches is demonstrated via three physical benchmarks, including (i) regulation of a room temperature (network), (ii) control of a road traffic (network), and (iii) control of a 7-dimensional nonlinear model of a BMW 320i car.

The research presented in the chapter has been submitted for publication in Nonlinear Analysis: Hybrid Systems, and the e-print can also be viewed on arXiv [88]. This research was the result of a collaboration with the University of Colorado Boulder. My role in this research is to provide the theoretical results and write the paper.

4.2 Introduction

Motivations. Stochastic control systems with continuous state and action sets have gained significant attention as an important modelling framework describing many real-life safety-critical applications, including traffic networks and power grids. Since the closed-form characterisation of optimal policies for continuous-space stochastic systems is not available in general, it is challenging to automatically synthesise policies [8] for such complex systems to achieve some high-level properties, *e.g.,* those expressed in linear temporal logic (LTL) [122]. To alleviate the complexity in synthesising policies for continuous-space stochastic systems, one promising approach is to discretise the state, synthesise an optimal policy for the resulting abstract finite-state model (using formal methods [8] or reinforcement learning (RL) [160]), and then translate the results back to the original system, while providing bounds on the error introduced by the discretisation process [143, 89].

Main contribution. The contribution of this chapter is twofold: First, we propose an RL approach to synthesise policies for the satisfaction of *finite-horizon* properties in unknown stochastic systems with uncountable state sets, while providing convergence guarantees. In particular, we leverage a closeness guarantee between probabilities of satisfaction by the unknown continuous-space stochastic system and its finite abstraction that can be controlled a-priori and utilise the classical convergence results for RL on finite MDPs. This approach enables us to apply model-free, off-the-shelf RL algorithms to compute ε-optimal strategies for continuous-space systems with a precision ε that is defined a-priori and without explicitly constructing finite abstractions. We also propose a novel potential-based reward shaping [118] technique to produce dense rewards that are based on the structure of the automata representing the specifications of interest.

In the second part of the chapter, we develop a scalable approach for the synthesis of controllers for *networks* of continuous-space stochastic systems with unknown dynamics so

that they satisfy finite-horizon properties. The controller for the whole network is composed from controllers synthesised for subsystems. Each of these controllers is obtained by solving a stochastic game, in which other components are treated as adversaries. Since probabilities are unknown, we use the converging multi-agent RL framework in [97, 98] to synthesise strategies. The compositional approach we propose provides a significant step towards scalability since convergent RL algorithms are computationally very expensive when applied to large finite MDPs. We demonstrate on two case studies how this approach can reduce the size of the system that needs to be reasoned over by orders of magnitude. We utilise a closeness guarantee between probabilities of satisfaction by subsystems and their implicit finite MDPs (which can be chosen a-priori), and leverage convergence results of minimax-Q learning [98] for solving stochastic games on finite MDPs. We provide, for the first time, a theoretical lower bound on the probability of satisfaction of finite-horizon properties by the original interconnected continuous-space stochastic system with unknown dynamics in terms of the bounds computed for the subsystems.

We demonstrate the effectiveness of the proposed results by applying them to three physical benchmarks including (i) regulation of a room temperature (network), (ii) control of a road traffic (network), and (iii) control of a 7-dimensional nonlinear model of a BMW 320i car.

Related work. A model-free RL framework for synthesising policies for unknown, and possibly continuous-state, stochastic systems is presented in [61, 181]. Our proposed approaches here differ from the ones in [61, 181] in two main directions. First and foremost, the proposed approaches in [61, 181] provide theoretical guarantees only if the underlying system has finitely many states, and the results for continuous-state systems are only empirically illustrated. In contrast, we utilise here a closeness guarantee between probabilities of satisfaction by the unknown continuous-space stochastic system and its finite abstraction to compute ε-optimal strategies for original systems using RL with a-priori defined precision ε. In addition, we propose in the second part of chapter a compositional RL framework for the policy synthesis of *networks* of continuous-space stochastic systems, whereas the results in [61, 181] only deal with monolithic systems.

A subset of the results in this chapter has recently appeared in [89]. Our approaches here differ from those in [89] in several directions. First and foremost, the results in [89] only apply to monolithic systems and, hence, suffer from the curse of dimensionality when confronted with large-scale interconnected systems. In contrast, we propose here a compositional RL framework for networks of continuous-space stochastic systems with unknown dynamics by breaking the main synthesis problem into simpler ones while still providing a lower bound on the probability of property satisfaction for interconnected systems based on optimal

probabilities of their subsystems. As the second main extension, we propose here a multi-level discretisation scheme for RL in which the agent learns control policies on a sequence of finer and finer discretisations of the same system. We show that this improves learning efficiency while preserving convergence results. Finally, we provide a detailed and mature description of the results announced in [89], including all proofs that were omitted.

4.3 Discrete-time stochastic control systems

Given functions $f_i : X_i \to Y_i$, for $1 \leq i \leq N$, their product $\bigtimes_{i=1}^{N} f_i : \bigtimes_{i=1}^{N} X_i \to \bigtimes_{i=1}^{N} Y_i$ is defined as $(x_1,\ldots,x_N) \mapsto [f_1(x_1);\ldots;f_N(x_N)]$. We represent a diagonal matrix with $\sigma_1,\ldots,\sigma_n$ as its entries as $\mathrm{diag}(\sigma_1,\ldots,\sigma_n)$.

We assume that random variables are measurable functions of the form $X : \Omega \to S_X$. Any random variable X induces a probability measure on its space $(S_X, \mathscr{F}_X)$ as $Prob\{A\} = \mathbb{P}_\Omega\{X^{-1}(A)\}$ for any $A \in \mathscr{F}_X$. A *discrete probability distribution*, or just distribution, over a (possibly countable) set X is a function $d : X \to [0,1]$ such that $\sum_{x \in X} d(x) = 1$ and $supp(d) = \{x \in X \mid d(x) > 0\}$ is at most countable. We say that $d : X \to [0,1]$ is a point distribution if $d(x) = 1$ for some $x \in X$.

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 metrisable space). Examples of a Borel space are the Euclidean spaces $\mathbb{R}^n$, their Borel subsets endowed with the subspace topology, as well as hybrid spaces. Any Borel space S is assumed to be endowed with a Borel σ-algebra, which is denoted by $\mathscr{B}(S)$. We say that a map $f : S \to Y$ is measurable whenever it is Borel measurable.

4.3.1 Discrete-time stochastic control systems

In this chapter, we consider networks of stochastic control systems in discrete time where each component, a discrete-time stochastic control system or dt-SCS, is defined as follows.

Definition 4.1 *A discrete-time stochastic control system (dt-SCS) is a tuple*

$$\Sigma = (X, U, W, \varsigma, f, Y, h), \tag{4.3.1}$$

where:

- *$X \subseteq \mathbb{R}^n$, a Borel space, is the state space of the system. We denote by $(X, \mathscr{B}(X))$ the measurable space with $\mathscr{B}(X)$ being the Borel sigma-algebra;*

- U *is the* external *input space;*
- $W \subseteq \mathbb{R}^p$ *is the* internal *input space;*
- ς *is a sequence of independent and identically distributed (i.i.d.) random variables from a sample space* Ω *to the set* $\mathscr{V}_\varsigma$*, namely* $\varsigma := \{\varsigma(k) : \Omega \to \mathscr{V}_\varsigma,\ k \in \mathbb{N}\}$*;*
- $f : X \times U \times W \times \mathscr{V}_\varsigma \to X$ *is a measurable function characterizing the state evolution of* Σ*;*
- $Y \subseteq \mathbb{R}^q$ *is the output space;*
- $h : X \to Y$*, a measurable function, maps states to outputs.*

We write $\mathbb{P}\{f(x,v,w,\cdot) \in B \mid x,u,w\}$ *for the probability that the next state is in* $B \in \mathscr{B}(X)$ *given current state* $x \in X$*, external input* $u \in U$*, internal input* $w \in W$*, when the remaining argument is distributed like the random variables in* ς*.*

The execution of Σ *from* $x(0) \in X$*, and inputs* $\{\upsilon(k) : \Omega \to U,\ k \in \mathbb{N}\}$ *and* $\{w(k) : \Omega \to W,\ k \in \mathbb{N}\}$ *is described by:*

$$\Sigma\colon \begin{cases} x(k+1) = f(x(k), \upsilon(k), w(k), \varsigma(k)), \\ y(k) = h(x(k)), \end{cases} \qquad k \in \mathbb{N}. \tag{4.3.2}$$

Remark 4.1 *The input space* U *of a dt-SCS* Σ *is in general a continuous Borel space,* e.g., *a subset of* $\mathbb{R}^m$*. Since any input sequence will be implemented by a digital controller, without loss of generality, we assume that the input set* U *is finite here.*

We will also consider special subclass of dt-SCS, called closed dt-SCS, where the internal inputs are absent, *i.e.,* when $p = 0$. Such systems may also result from considering an interconnection of dt-SCSs (cf. Definition 4.2), monolithically. For notational convenience, we represent closed dt-SCS as a tuple (X, U, ς, f) and its execution can be simplified to

$$\Sigma\colon \begin{cases} x(k+1) = f(x(k), \upsilon(k), \varsigma(k)), \\ y(k) = h(x(k)), \end{cases} \qquad k \in \mathbb{N}. \tag{4.3.3}$$

For a closed dt-SCS, we write $\mathbb{P}\{f(x,v,\cdot) \in B \mid x,u\}$ for the probability that the next state is in B given the current state $x \in X$ and input $u \in U$.

For emphasis, we call a non-closed dt-SCS open. When clear from context, we drop the open or closed specifier.

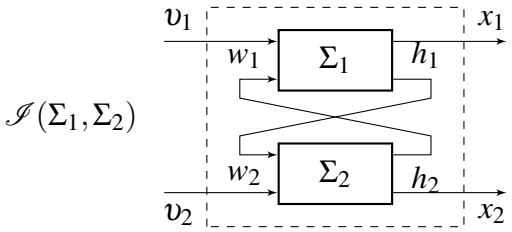


Fig. 4.1 Interconnection of stochastic control subsystems Σ_1 and Σ_2 [88].

Definition 4.2 (Network of dt-SCS) *For $1 \leq i \leq N$, let $\Sigma_i = (X_i, U_i, W_i, \varsigma_i, f_i, Y_i, h_i)$ be a family of N open dt-SCS. The network of $\langle \Sigma_i \rangle_{1 \leq i \leq N}$ is defined by the interconnection map $g : \bigtimes_{i=1}^{N} Y_i \to \bigtimes_{i=1}^{N} W_i$, and gives rise to a closed dt-SCS $\mathscr{I}_g(\Sigma_1, \ldots, \Sigma_N) = (X, U, \varsigma, f)$, where $X := \bigtimes_{i=1}^{N} X_i$, $U := \bigtimes_{i=1}^{N} U_i$, and $f := \bigtimes_{i=1}^{N} f_i$, subjected to the following interconnection constraint:*

$$[w_1; \ldots; w_N] = g(h_1(x_1), \ldots, h_N(x_N)). \tag{4.3.4}$$

An example of the interconnection of two stochastic control subsystems Σ_1 and Σ_2 is illustrated in Fig. 4.1.

4.3.2 Stochastic games and Markov decision processes

The semantics of an open dt-SCS Σ can sometimes be given as a stochastic game [125, 44] between two players—player Max (the control), who controls the external inputs U, and player Min (the adversary), who controls the internal inputs. We assume that the adversary is more powerful than the controller in that the adversary can see the choices of the controller at every step. From control synthesis perspective, this view results in a cautious controller with a pessimistic view of the environment. On the other hand, a strategy computed in this manner also works against the weaker adversary.

Definition 4.3 *A* stochastic game arena *(SGA) is a tuple $\mathscr{G} = (S, A, T, S_{\text{Max}}, S_{\text{Min}})$ where:*

- *S is (potentially uncountable) state space;*
- *A is the set of actions and we write $A(s)$ for the set of actions enabled at a state $s \in S$;*
- *$T : S \times A \times \mathscr{B}(S) \to [0,1]$ is a conditional stochastic kernel that assigns to any $s \in S$ and $a \in A$, a probability measure $P(\cdot|s,a)$ on the measurable space $(S, \mathscr{B}(S))$.*
- *$S_{\text{Max}} \subseteq S$ and $S_{\text{Min}} \subseteq S$ form a partition of S into the set of states controlled by players Max and Min, respectively.*

For the stochastic kernel T, state $s \in S$, action $a \in A$, and set $B \in \mathscr{B}(S)$, we write $T(B \mid s,a)$ for $T(s,a,B)$.

We say that a SGA is finite, if both S and A are finite. For finite SGAs, the transition function $T(s,a,\cdot)$ is a discrete probability distribution for every $s \in S$ and $a \in A$. For a finite SGA, we write $T(s' \mid s,a)$ for $T(s,a,s')$ for all $s,s' \in S$ and $a \in A$. Also, we refer to an SGA as a Markov decision process (MDP) if $S_{\text{Min}} = \emptyset$. We represent an MDP as (S,A,T). An MDP (S,A,T) is finite if both S and A are finite.

Definition 4.4 (dt-SCS: Semantics) *An open dt-SCS $\Sigma = (X,U,W,\varsigma,f,Y,h)$ can be interpreted[76, Proposition 7.6] as an SGA $\mathscr{G}_\Sigma = (S,A,T,S_{\text{Max}},S_{\text{Min}})$, where*

- *S=$X \cup (X \times U)$ such that $S_{\text{Max}} = X$ and $S_{\text{Min}} = X \times U$;*
- *A=$U \cup W$ such that for all $s \in X$ we have $A(s) = U$ and for all $s \in X \times U$ we have $A(s) = W$;*
- *$T : S \times A \times \mathscr{B}(S) \to [0,1]$ such that*

  1. *for $x \in S_{\text{Max}}$ and $u \in U$*

  $$T((x,u) \mid x,u) = 1, \quad T(S \backslash \{(x,u)\} \mid x,u) = 0,$$

  2. *for all $(x,u) \in S_{\text{Min}}$, $w \in W$, and all $B \in \mathscr{B}(X)$*

  $$T(B \mid (x,u),w) = \mathbb{P}\Big\{ f(x,v,w,\cdot) \in B \mid x,u,w \Big\}.$$

Similarly, a closed dt-SCS $\Sigma = (X,U,\varsigma,f)$ can be equivalently *represented as a Markov decision process $\mathscr{M}_\Sigma = (S = X, A = U, T)$ where $T : S \times A \times \mathscr{B}(S) \to [0,1]$ such that for all $x \in X$, $u \in U$, and $B \in \mathscr{B}(X)$, we have that*

$$T(B \mid x,u) = \mathbb{P}\Big\{ f(x,u,\cdot) \in B \,\Big|\, x,u \Big\}.$$

Abusing notation, we write Σ for its SGA $\mathscr{G}_\Sigma$ or MDP $\mathscr{M}_\Sigma$.

The objective in an SGA is to determine a policy—a decision rule for every step to choose the next action—for both players that optimise a given objective. A decision rule may be *history-dependent* (depends on all the information available at a given time step) or *memoryless* (depends only on the current state); they may be *stochastic* (defines a stochastic kernel on the actions) or *deterministic* (chooses a fixed action with probability 1). For the

objectives in this chapter, w.l.o.g. we only need to consider memoryless, deterministic policies [36]. We call these policies Markov policies as defined next.

Definition 4.5 (Markov policies) *For an SGA $\mathcal{G}$, a Markov policy ρ of player Max is a sequence $(\rho_0, \rho_1, \rho_2, \ldots)$ of decision rules where each rule $\rho_n : S_{\text{Max}} \to A$, for $n \in \mathbb{N}$, is a universally measurable function such that $\rho_n(s) \in A(s)$ for all $s \in S$. Similarly, a Markov policy ξ of player Min is a sequence $(\xi_0, \xi_1, \xi_2, \ldots)$ where $\xi_n : S_{\text{Min}} \to A$, for $n \in \mathbb{N}$, is a universally measurable function such that $\xi_n(s) \in A(s)$ for all $s_{\text{Min}} \in S$. We write $\Pi^{\mathcal{G}}_{\text{Max}}$ and $\Pi^{\mathcal{G}}_{\text{Min}}$ for the set of all Markov policies of player Max and player Min, respectively. For an MDP $\mathcal{M}$ we write $\Pi^{\mathcal{M}}$ for the set of policies. We omit the superscripts $\mathcal{M}$ and $\mathcal{G}$ when clear from the context.*

Definition 4.6 (Solution process) *Any pair of Markov policies $\rho \in \Sigma_{\text{Max}}$ and $\xi \in \Sigma_{\text{Min}}$, and initial state $s \in S$, characterise a unique stochastic process over sequences of states and actions. We write $\mathcal{G}^s_{\rho,\xi}$ for this stochastic process and write S_n and A_n for the random variables corresponding to the state and action at time step $n \in \mathbb{N}$. We write $\mathbb{E}^s_{\rho,\xi}[\cdot]$ for the expected value of a random variable for the stochastic process $\mathcal{G}^s_{\rho,\xi}$. Similarly, we write $\mathcal{M}^s_\rho$ for the stochastic process of an MDP $\mathcal{M}$ with initial state s and policy ρ, and $\mathbb{E}^s_\rho[\cdot]$ for the expected value of a random variable for $\mathcal{M}^s_\rho$.*

4.3.3 Reinforcement learning

Reinforcement learning (RL) [160] is a sample-based approach to controller synthesis in stochastic environments where a learning agent—the controller—relies on scalar reward signals to select inputs aimed at achieving a prescribed objective. RL is intimately connected to optimal control, with the main distinction being that the stochastic system is "unknown," *i.e.,* the transition probabilities as well as the reward structure of the underlying stochastic system may not be known in advance, but can be sampled. Strong convergence guarantees [160, 98] exists for learning optimal control using RL for finite stochastic game arenas. For this reason, the presentation on this subsection focuses on finite SGAs.

A stochastic game is a pair $(\mathcal{G}, \mathsf{R})$ where $\mathcal{G} = (S, A, P, S_{\text{Max}}, S_{\text{Min}})$ is a finite SGA and $\mathsf{R} : S \times A \times S \to \mathbb{R}$ is a (scalar) reward function. From an initial state $s = s_0 \in S$, the game evolves by having the player that controls s_k at time step $k \in \mathbb{N}$ select an action $a_{k+1} \in A(s_k)$. The state then evolves under probability distribution $P(\cdot \mid s_k, a_k)$ resulting in a next state s_{k+1} and reward $r_{k+1} = \mathsf{R}(s_k, a_k, s_{k+1})$. Given a discount factor $\gamma \in [0, 1)$, the payoff (from player Min to player Max) of the SGA is defined as the γ-discounted sum of rewards, *i.e.,* $\sum_{k=0}^{\infty} r_{k+1} \gamma^k$. The objective of player Max is to maximise the expected payoff, while the

objective of player Min is the opposite. Recall that Π_{Max} and Π_{Min} are the set of Markov policies for player Max and player Min in $\mathscr{G}$. We say that a policy $\rho_* \in \Pi_{\text{Max}}$ is optimal if

$$\inf_{\xi \in \Pi_{\text{Min}}} \mathbb{E}^s_{\rho_*,\xi}\Big[\sum_{k=0}^{\infty} \mathsf{R}(S_k, A_{k+1}, S_{k+1})\gamma^k\Big] \quad \geq \quad \inf_{\xi \in \Pi_{\text{Min}}} \mathbb{E}^s_{\rho,\xi}\Big[\sum_{k=0}^{\infty} \mathsf{R}(S_k, A_{k+1}, S_{k+1})\gamma^k\Big],$$

for all $\rho \in \Pi_{\text{Max}}$. The optimal policies for player Min are defined analogously. The goal of RL is to compute optimal policies for both players with samples from the game, without apriori knowledge of the transition probability and rewards.

The RL algorithms solve this problem by learning a state-action value function defined to be

$$Q_{\rho,\xi}(s,a) = \mathbb{E}^s_{\rho,\xi}\Big[\sum_{k=0}^{\infty} \mathsf{R}(S_k, A_{k+1}, S_{k+1})\gamma^k\Big],$$

where $\rho \in \Pi_{\text{Max}}$ and $\xi \in \Pi_{\text{Min}}$. These are called Q-values, or quality, of the pair $(s,a) \in S \times A$. Since the state and action space is finite, this can be represented as a look-up table called the Q-table. RL algorithms that use a look-up table are called *tabular* methods. We define

$$Q_*(s,a) = \sup_{\rho \in \Pi_{\text{Max}}} \inf_{\xi \in \Pi_{\text{Min}}} Q_{\rho,\xi}(s,a).$$

Given $Q_*(s,a)$, one can extract the policy for both players by selecting the maximum value action in states controlled by player Max and the minimum value action in states controlled by player Min. The following recurrence relation, known as Bellman optimality equations, characterises the optimal solutions and forms the basis for computing the Q-table with dynamic programming:

$$Q_*(s,a) = \sum_{s' \in S} T(s'|s,a) \cdot \Big(\mathsf{R}(s',a,s) + \gamma \cdot \operatorname*{opt}_{a' \in A(s')} Q_*(s',a')\Big),$$

where opt is max if $s' \in S_{\text{Max}}$ and min if $s' \in S_{\text{Min}}$. Minimax-Q learning [97] estimates the dynamic programming update from the stream of samples by performing the following update at each time step:

$$\begin{aligned} Q(s(k),a(k)) \leftarrow (1-\alpha_k)Q(s(k),a(k)) + \\ \alpha_k(r(k+1) + \gamma \operatorname*{opt}_{a' \in A(s(k+1))} Q(s(k+1),a')), \end{aligned}$$

where $\leftarrow$ is the assignment operator, and $\alpha_k \in (0,1)$ is the learning rate at time step k and is a parameter of the algorithm.

Theorem 4.1 (Minimax-Q[98]) *Minimax-Q learning is guaranteed to converge to the unique fixpoint $Q_*(s,a)$ if $r(k)$ is bounded, learning rate satisfies the Robbins-Monro conditions (*i.e., $\sum_{k=0}^{\infty}\alpha_k=\infty$ *and* $\sum_{k=0}^{\infty}\alpha_k^2<\infty$*), and all state-action pairs are seen infinitely often.*

Remark 4.2 *For finite horizon objectives (*i.e., *when $r(k)$=0 for all k>N for some fixed N>0), this convergence result holds even for undiscounted case,* i.e., *when γ=1.*

Minimax-Q algorithm produces the controller directly, without internally producing estimates of the unknown system dynamics, which is called *model-free*. The minimax-Q algorithm of Littman [97] reduces to standard Q-learning [172] algorithm for MDPs, *i.e.,* when $S_{\text{Min}}=\emptyset$. Our choice of minimax-Q learning and Q-learning is due to its popularity and performance.

When SGAs are not finite, one can no longer represent the Q-values in a look-up table. A popular alternative is based on neuro-dynamic programming [15] where we can represent the Q-values by a parameterised function. The study of RL when the parameterised function is an artificial neural network is called *deep reinforcement learning*, and has seen recent empirical success. Deep Q-learning [112] is a popular adaptation of the Q-learning algorithm which uses an artificial neural network to estimate Q-values. Unfortunately, for deep RL convergence to the optimal solution is not guaranteed.

4.3.4 Finite-horizon specifications

Logics like Linear Temporal Logic (LTL) and automata provide rigorous and unambiguous formalisms to express requirements for stochastic control systems [8].

Formulae of LTL and ω-automata describe sets of infinite words whose letters are drawn from a finite alphabet. In applications like ours, the alphabet is the powerset of a set of *atomic propositions*, which are measurable predicates defined over the states of a system. We are interested in *finite-horizon* properties; that is properties for which membership can be decided by examining a fixed-length prefix of a word. We briefly discuss two ways to restrict LTL to finite-horizon properties. The first way starts with a fragment of LTL known as *syntactically co-safe linear temporal logic* (scLTL) [142, 84, 13], which is defined below.

Definition 4.7 (Syntactically co-safe LTL (scLTL)) *scLTL over the atomic propositions* AP *is a fragment of LTL such that the negation operator ($\neg$) only occurs before atomic propositions, and characterised by the following grammar:*

$$\phi ::= p \mid \neg p \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \mathsf{X}\phi \mid \phi_1 \,\mathsf{U}\, \phi_2,$$

with $p \in \Sigma_a$, where $\Sigma_a = 2^{\mathsf{AP}}$, and $\vee, \wedge, \mathsf{X}, \mathsf{U}$ are the customary logical and temporal operators [8, 122].

Even though scLTL formulae are defined over infinite words (as are LTL formulae), their satisfaction only depends on a (finite) prefix of a word [84]. Any infinite word $\omega \in \Sigma_{\mathsf{a}}^{\omega}$ satisfying an scLTL formula ϕ (written $\varpi \models \phi$) has a finite prefix ω_f such that all infinite extensions of ω_f also satisfy the formula ϕ. We denote the set of all such prefixes associated with scLTL formula ϕ by $\mathscr{L}_f(\phi)$.

Given a dt-SCS with state space X and a set of atomic propositions $AP \subseteq 2^X$, a trajectory of the dt-SCS defines a word in $\Sigma_{\mathsf{a}}^{\omega}$. We write $\mathbb{P}(\Sigma_\rho^x \models \phi)$ for the probability that a trajectory of a closed dt-SCS Σ started from state x and governed by policy ρ satisfies ϕ. For an open dt-SCS, we write $\mathbb{P}(\Sigma^x_{(\rho,\xi)} \models \phi)$. This notation is extended in the natural way to MDPs, and SGAs.

A formula of scLTL describes an open set in the Cantor topology of $\Sigma_{\mathsf{a}}^{\omega}$ [4]. It is decided by a deterministic finite automaton on finite words (DFA) that accepts $\mathscr{L}_f(\phi)$ [84].

Definition 4.8 *A deterministic finite automaton (DFA) is a tuple $\mathscr{A} = (Q, \Sigma_a, \mathsf{t}, q_0, F_a)$, where Q is a finite set of states, Σ_a is a finite alphabet, $\mathsf{t} : Q \times \Sigma_a \to Q$ is a transition function, $q_0 \in Q$ is the initial state, and $F_a \subseteq Q$ is a set of accepting states. We write λ for the empty word and Σ_a^* for the set of finite strings over Σ_a. The extended transition function $\hat{\mathsf{t}} : Q \times \Sigma_a^* \to Q$ (extended from letters to words) is defined as:*

$$\hat{\mathsf{t}}(q, \bar{w}) = \begin{cases} q, & \text{if } \bar{w} = \lambda, \\ \mathsf{t}(\hat{\mathsf{t}}(q,x), a), & \text{if } \bar{w} = xa \text{ for } x \in \Sigma_a^* \text{ and } a \in \Sigma_a. \end{cases}$$

The language accepted by a DFA $\mathscr{A}$ is $\mathscr{L}(\mathscr{A}) = \{\bar{w} \in \Sigma_a^* \,|\, \hat{\mathsf{t}}(q_0, \bar{w})) \in F_a\}$.

For verification and synthesis purposes, an scLTL property can be compiled into a DFA $\mathscr{A}_\phi$ over the alphabet 2^{AP} such that $\mathscr{L}_f(\phi) = \mathscr{L}(\mathscr{A}_\phi)$. This construction is routine; we refer the interested reader to [84] for the details. The resulting DFA has a unique accepting state whose out-going transitions are all self-loops. Such a DFA is known as a *co-safety automaton*. In the following, we assume that the DFA $\mathscr{A}_\phi$ for an scLTL property ϕ is a co-safety one.

Some formulae of scLTL describe finite-horizon properties. For example, $p \vee \mathsf{X}\mathsf{X}q$ only requires checking the first three letters of a word. Other properties, like $p \,\mathsf{U}\, q$ are satisfied by finite words of arbitrary length. We can, however, adjoin a finite time horizon $\mathscr{T}$ to a formula ϕ and stipulate that an infinite word satisfies $(\phi, \mathscr{T})$ if it has a prefix of length at most $\mathscr{T} + 1$ that is in $\mathscr{L}(\mathscr{A}_\phi)$.

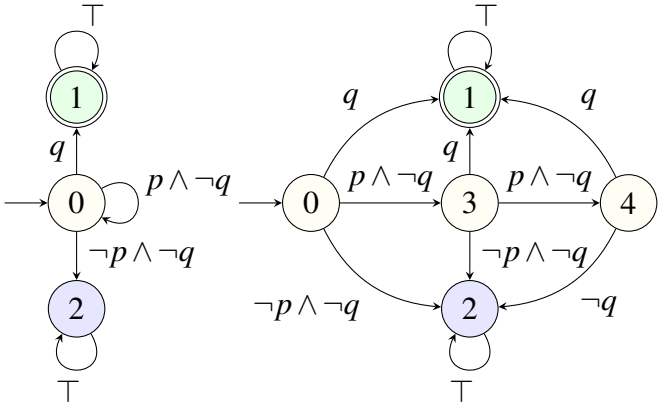


Fig. 4.2 DFA for $p \mathsf{U} q$ with no time horizon (left) and with $\mathscr{T} = 2$ (right). The finite-horizon DFA may be obtained by unrolling the co-safety DFA or by translating the finite-horizon formula $q \vee (p \wedge \mathsf{X}(q \vee (p \wedge \mathsf{X} q)))$ [88].

The sets described by scLTL formulae with finite time horizon are *clopen*. This follows from the observation [85, Lemma 2.1] that a subset of $\Sigma_{\mathsf{a}}^{\omega}$ is both open and closed if and only if it has a finite minimal basis. One then observes that there are only finitely many words of length up to $\mathscr{T}+1$ in $\mathscr{L}_f(\phi)$. Clopen sets are closed under finite union, intersection, and complementation. Accordingly, finite-horizon LTL properties are closed under finite Boolean operations.

A DFA that accepts a clopen set can be restricted to have exactly one accepting sink state and one rejecting sink state. All other states are transient: they can only be visited once. We call automata with this structure *finite-horizon automata*. The finite-horizon automaton for the bounded time-horizon satisfaction of ϕ can be computed by unrolling the co-safety automaton $\mathscr{A}_\phi$ (cf. Fig. 4.2).

The second way we consider to restrict LTL to finite-horizon properties is to remove the “until" operator from the definition of scLTL, leaving only X as temporal modality. The resulting fragment also describes clopen sets and finite-horizon automata can be obtained by direct translation of such properties. On the one hand, the formulae that forgo the until operator do not need a separate specification of the time horizon. On the other hand, they tend to be more cumbersome to write. Hence, scLTL plus time horizon is usually preferable and is the type of specification assumed in the sequel.

4.4 Problem definition

We say that a dt-SCS $\Sigma = (X, U, W, \varsigma, f, Y, h)$ is unknown if f and the distribution of ς are not known explicitly, but can be sampled. We are interested in automatically synthesising

controllers for a network of unknown dt-SCS (Definition 4.2) whose requirements are specified as finite-horizon properties. We present our solution to this problem in two parts. In the first part (Section 4.5 and 4.6) we consider the network of systems monolithically and study how to synthesise a centralised control for such monolithic system by studying the synthesis for closed and unknown dt-SCSs against scLTL specifications using RL. We extend this approach to compositionally design a decentralised controller for a network of open dt-SCSs by exploiting minimax-Q learning in the second part (Section 4.7 and 4.8). Finally, we demonstrate the effectiveness and scalability of our approaches on three physical benchmarks.

We emphasise that, even when the system is known, there is no *closed-form solution* for computing optimal policies enforcing scLTL specifications over *continuous-space* stochastic control systems. One can employ the approximation approaches, discussed in Subsection 4.5.1, to synthesise those policies which, however, suffer severely from the curse of dimensionality and, more importantly, require knowing precisely the probabilistic evolution of states in models. Instead, we propose an RL approach synthesising policies for unknown continuous-space stochastic systems while providing *quantitative probabilistic guarantees* on the satisfaction of properties.

4.5 Controller synthesis for unknown continuous-space stochastic control systems

In this section, we are concerned with automatically synthesising controllers for the unknown continuous-space stochastic control systems in (4.3.3) whose requirements are specified in finite-horizon LTL. Given a discrete-time stochastic control system $\Sigma = (X, U, \varsigma, f)$, where f and the distribution of ς are unknown and given a finite-horizon formula ϕ, we seek a Markov policy enforcing satisfaction of the property ϕ in Σ with probability within a guaranteed threshold from the unknown optimal probability.

In order to provide any formal guarantee, we need to make further assumptions on the dt-SCS. In particular, we assume that the dynamical system in (4.3.3) is Lipschitz-continuous with a constant $\mathscr{H}$. Consider the system in (4.3.3) where $\varsigma(\cdot)$ is i.i.d. with a distribution $t_\varsigma(\cdot)$. Suppose that the vector field f is continuously differentiable and the matrix $\frac{\partial f}{\partial \varsigma}$ is invertible. Then, the *implicit function theorem* guarantees the existence and uniqueness of a function $\bar{g} : X \times X \times U \to \mathscr{V}_\varsigma$ such that $\varsigma(k) = \bar{g}(x(k+1), x(k), \upsilon(k))$. In this case, the conditional

density function is:

$$t_x(x'|x,u) = \left|\det\left[\frac{\partial \bar{g}}{\partial x'}(x',x,\upsilon)\right]\right| t_\varsigma(\bar{g}(x',x,\upsilon)).$$

The Lipschitz constant $\mathscr{H}$ is specified by the dependency of the function $\bar{g}(x',x,\upsilon)$ on the variable x. As a special case, consider a nonlinear system with an additive noise

$$f(x,\upsilon,\varsigma) = f_a(x,\upsilon) + \varsigma.$$

Then the invertibility of $\frac{\partial f}{\partial \varsigma}$ is guaranteed and $\bar{g}(x',x,\upsilon) = x' - f_a(x,\upsilon)$. In this case, $\mathscr{H}$ is the product of the Lipschitz constant of $t_\varsigma(\cdot)$ and $f_a(\cdot)$.

Example 4.1 *Consider a dt-SCS Σ with linear dynamics $x(k+1) = Ax(k) + \bar{B}\upsilon(k) + \varsigma(k)$, where $A = [a_{ij}]$, and $\varsigma(k)$ is i.i.d. for $k = 0,1,2,\ldots$ with a normal distribution having zero mean and covariance matrix* $\mathrm{diag}(\sigma_1,\ldots,\sigma_n)$. *Then, one obtains $\mathscr{H} = \sum_{i,j} \frac{2|a_{ij}|}{\sigma_i\sqrt{2\pi}}$. Note that for the computation of the approximation error (cf.* (4.5.2)*), it is sufficient to know an upper bound on entries of the matrix A and a lower bound on the standard deviation of the noise.*

An alternative way of computing the Lipschitz constant $\mathscr{H}$ is to estimate it from sample trajectories of Σ. This can be done by first constructing a non-parametric estimation of the conditional density function using techniques proposed in [136] and then compute $\mathscr{H}$ numerically using the derivative of the estimated conditional density function. We refer the interested reader to [89, equation (3)] for more details.

Now we have all the required ingredients to state the main problem we address in the first part of the chapter.

Problem 4.1 *Let ϕ be a finite-horizon formula and $\Sigma = (X,U,\varsigma,f)$ a closed discrete-time stochastic control system, where f and the distribution of ς are unknown, but the Lipschitz constant $\mathscr{H}$ is known. Synthesise a Markov policy that enforces satisfaction of the property ϕ by Σ with probability within a-priori defined threshold ε of the unknown optimal probability.*

To present our solution to this problem, we first present a technical result connecting continuous-space stochastic systems with corresponding finite abstractions. We then exploit this result to provide a reinforcement learning-based solution to Problem 4.1. We emphasise that we do not explicitly construct finite abstractions of continuous-space stochastic systems.

In fact, we cannot construct them because the dynamics of continuous-space systems are unknown.

4.5.1 Abstraction of dt-SCS Σ by a finite MDP

We approximate the dt-SCS Σ in (4.3.3) with a *finite* $\widehat{\Sigma}$ using an abstraction algorithm. The algorithm first constructs a finite partition of the state space $X = \cup_i \mathsf{X}_i$. Then representative points $\hat{x}_i \in \mathsf{X}_i$ are selected as abstract states. Given a dt-SCS $\Sigma = (X, U, \varsigma, f)$, the constructed finite MDP $\widehat{\Sigma}$ is

$$\widehat{\Sigma} = (\hat{X}, \hat{U}, \varsigma, \hat{f}), \tag{4.5.1}$$

where $\hat{X} = \{\hat{x}_i, i = 1, \ldots, n_x\}$, a finite subset of X, and $\hat{U} := U$ are finite state and input sets of the MDP $\widehat{\Sigma}$. Moreover, $\hat{f} : \hat{X} \times \hat{U} \times \mathscr{V}_\varsigma \to \hat{X}$ is defined as $\hat{f}(\hat{x}, \hat{\upsilon}, \varsigma) = \mathscr{Q}_x(f(\hat{x}, \hat{\upsilon}, \varsigma))$, where $\mathscr{Q}_x : X \to \hat{X}$ is the map that assigns to any $x \in X$, the representative point $\hat{x} \in \hat{X}$ of the corresponding partition set containing x. The initial state of $\widehat{\Sigma}$ is also selected according to $\hat{x}_0 := \mathscr{Q}_x(x_0)$ with x_0 being the initial state of Σ.

Abusing notation, a policy in $\widehat{\Sigma}$ can be considered as a valid policy in Σ, *i.e.,* $\Pi^{\widehat{\Sigma}} \subseteq \Pi^{\Sigma}$. The following theorem [147] shows the closeness between a continuous-space stochastic control system Σ and its finite abstraction $\widehat{\Sigma}$ in a probabilistic setting.

Theorem 4.2 *Let $\Sigma = (X, U, \varsigma, f)$ be a continuous-space stochastic control system and $\widehat{\Sigma} = (\hat{X}, \hat{U}, \varsigma, \hat{f})$ be its finite abstraction. For finite-horizon specification ϕ, Markov policy $\rho \in \Pi^{\hat{\Sigma}}$, and initial state $x \in X$, we have:*

$$|\mathbb{P}(\Sigma^x_\rho \models \phi) - \mathbb{P}(\widehat{\Sigma}^{\hat{x}}_\rho \models \phi)| \leq \varepsilon, \quad \textit{with } \varepsilon := \mathscr{T}\delta\mathscr{H}\mathscr{L}, \tag{4.5.2}$$

where $\hat{x} = Q_x(x)$, $\mathscr{T}$ is the finite time horizon, δ is the state discretisation parameter, $\mathscr{H}$ is the Lipschitz constant of the stochastic kernel, and $\mathscr{L}$ is the Lebesgue measure of the state space. Moreover, if we take the optimal policy $\rho_* \in \Pi^{\hat{\Sigma}}$ *of satisfying the specification in $\widehat{\Sigma}$ and applying it to Σ, we have the error at most 2ε,* i.e.,

$$\Big| \max_{\rho \in \Pi^{\Sigma}} \mathbb{P}(\Sigma^x_\rho \models \phi) - \mathbb{P}(\Sigma^x_{\rho_*} \models \phi) \Big| \leq 2\varepsilon. \tag{4.5.3}$$

The error bound ε in (4.5.2) is obtained by characterising $\mathbb{P}(\Sigma^x_\rho \models \phi)$ recursively similar to dynamic programs (DP). This error is related to the approximation of the continuous kernel via a discrete one, hence, the term $\delta\mathscr{H}$ appears in ε. There is also an integration over the state space, thus $\mathscr{L}$ appears in ε. Finally, the errors contributed in every iteration of the DP are added, hence, the horizon $\mathscr{T}$ appears in ε.

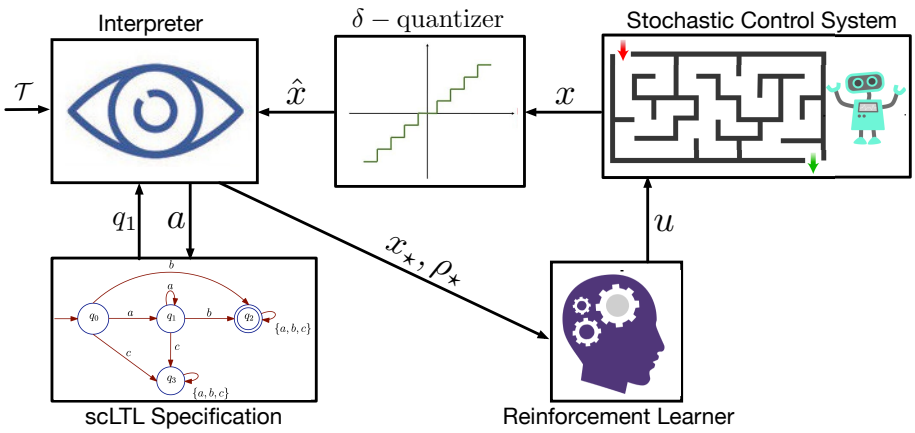


Fig. 4.3 Model-free reinforcement learning is employed by DFA $\mathscr{A}_\phi$ corresponding to scLTL objective ϕ to provide scalar rewards by combining DFA $\mathscr{A}_\phi$ and a δ-quantised observation set of the continuous-space MDP Σ. In particular, the δ-quantised observation set of the continuous-space MDP Σ is used by an *interpreter* process to compute a run of $\mathscr{A}_\phi$. When the run of $\mathscr{A}_\phi$ reaches a final state, the interpreter gives the reinforcement learner a positive reward and the training episode terminates. Any converging reinforcement learning algorithm over such δ-quantised observation set is guaranteed to maximise the probability of satisfaction of the scLTL objective ϕ and converge to a 2ε-optimal strategy over the concrete dt-SCS Σ, thanks to Theorem 4.2 [88].

Remark 4.3 *Note that in order to employ Theorem 4.2, one can first a-priori fix the desired threshold ε in* (4.5.2)*. According to the values of $\mathscr{H}$, $\mathscr{L}$, and $\mathscr{T}$, one computes the required discretisation parameter as $\delta = \frac{\varepsilon}{\mathscr{T}\mathscr{H}\mathscr{L}}$. For instance in the case of a uniform quantiser, one can divide each dimension of the set X into intervals of size $\delta/\sqrt{n}$ with n being the dimension of the set.*

4.6 Synthesis via reinforcement learning

In this section, we sketch how we apply Theorem 4.2 to solve Problem 4.1 when conditional stochastic kernels are unknown. We begin by detailing the solution of finding optimal policies for finite-horizon properties in the case of known MDPs, and then we show how to exploit that to provide an RL-based algorithm to synthesise an optimal policy.

4.6.1 Product Markov decision process

It follows from Theorem 4.2 that one can construct a finite MDP $\widehat{\Sigma}$ from a continuous-space dt-SCS Σ with known conditional stochastic kernels such that the optimal probability of satisfaction of a finite-horizon specification ϕ for $\mathscr{T}$ steps in $\widehat{\Sigma}$ is no more than 2ε-worse than the optimal policy in Σ; see the definition of ε in Theorem 4.2. Hence, given a dt-SCS Σ with known conditional stochastic kernels, a finite-horizon property ϕ with horizon $\mathscr{T}$, a

2ε-optimal policy to satisfy ϕ is computed using a suitable finite MDP with the corresponding δ as the state discretisation parameter. This problem can be solved using the finite-horizon dynamic programming over the product of $\widehat{\Sigma}$ and the DFA $\mathscr{A}_\phi$ (cf. Definition 4.7 and the paragraph afterward) by giving a scalar reward to all transitions once a final state of $\mathscr{A}_\phi$ is reached.

Definition 4.9 (Product MDP) *Given a finite MDP $\widehat{\Sigma} = (\hat{X}, \hat{U}, \hat{T})$ with initial state $\hat{x}_0 \in \hat{X}$, a labeling function $L : X \to \Sigma_a$ (cf. Subsection 4.3.4), and a DFA $\mathscr{A}_\phi = (Q, \Sigma_a, \mathsf{t}, q_0, F_a)$ capturing the finite-horizon specification ϕ, we define the* product MDP *$\mathscr{M}_\star$ as a finite MDP $(X_\star, U_\star, T_\star)$ with initial state $x_{\star 0}$, and reward function $\mathsf{R}_\star$ where:*

- *$X_\star = \hat{X} \times Q$ is the set of states;*
- *$U_\star = \hat{U}$ is the set of actions;*
- *$T_\star : X_\star \times U_\star \times X_\star \to [0,1]$ is the probabilistic transition function defined as*

$$T_\star((x,q),v,(x',q')) = \begin{cases} \hat{T}(x,v,x'), & \text{if } q' = \mathsf{t}(q, L(x)), \\ 0, & \text{otherwise,} \end{cases}$$

- *$x_{\star 0} = (x_0, q_0)$ is the initial state; and*
- *$\mathsf{R}_\star : X_\star \times U_\star \times X_\star \to \mathbb{N}$ is the reward function defined as:*

$$\mathsf{R}_\star((x,q),v,(x',q')) = \begin{cases} 1, & \text{if } q' \in F_a, \\ 0, & \text{otherwise.} \end{cases}$$

Recall that the DFA $\mathscr{A}_\phi$ corresponding to a finite-horizon specification ϕ has the property that there is a unique accepting state and all out-going transitions from that state are self-loops. It follows that total optimal expected reward in the product is equal to the optimal probability of satisfying the specification.

Proposition 4.1 (Product preserves probability [28]) *An expected reward-optimal policy in $(X_\star, U_\star, T_\star)$ from initial state $x_{\star 0} \in S_\star$ and reward function $\mathsf{R}_\star$ along with $\mathscr{A}_\phi$ characterises an optimal policy in $\widehat{\Sigma}$ to satisfy ϕ. The optimal expected total reward and an optimal policy can be computed in polynomial time [120] in the size of the MDP and the DFA.*

4.6.2 Unknown conditional stochastic kernels

When stochastic kernels are unknown, Theorem 4.2 still provides the correct probabilistic bound given a discretisation parameter δ if the Lipschitz constant $\mathscr{H}$ is known. This observation enables us to employ RL algorithms over the underlying discrete MDP without explicitly constructing the abstraction by simply restricting observations of the reinforcement learner to the closest representative point in the set of partitions.

Model-free RL can be employed under such observations by using the DFA $\mathscr{A}_\phi$ to provide scalar rewards as defined in Definition 4.9. The observations of the MDP are used by an *interpreter* process to compute a run of the DFA. When the DFA reaches a final state, the interpreter gives the reinforcement learner a positive reward and the training episode terminates. Since the product MDP $\mathscr{M}_\star$ is a finite MDP, from Proposition 4.1, it follows that any correct and convergent RL algorithm that maximises this expected reward is guaranteed to converge to a policy that maximises the probability of satisfaction of the scLTL objective. From Theorem 4.2, it then follows that any converging reinforcement learning algorithm [68, 17] over such finite observation set then converges to a 2ε-optimal policy over the concrete dt-SCS Σ, thanks to Theorem 4.2. We summarise the proposed solution in the following theorem.

Theorem 4.3 *Let ϕ be a finite-horizon formula, $\varepsilon > 0$, and $\Sigma = (X, U, \varsigma, f)$ be a continuous-space MDP, where f and the distribution of ς are unknown but the Lipschitz constant $\mathscr{H}$ as discussed before is known. For a discretisation parameter δ satisfying $\mathscr{T}\delta\mathscr{H}\mathscr{L} \leq \varepsilon$, a convergent model-free RL algorithm (*e.g., *Q-learning [17] or TD(λ) [68]) over $\widehat{\Sigma}$ with a reward function guided by the DFA $\mathscr{A}_\phi$, converges to a policy which is 2ε-optimal for Σ.*

4.6.3 Reward shaping: overcoming sparse rewards

Consider a finite MDP $\widehat{\Sigma} = (\hat{X}, \hat{U}, \hat{T})$, a co-safety automaton $\mathscr{A}_\phi = (Q, \Sigma_{\mathsf{a}}, \mathtt{t}, q_0, q_F)$, and their product MDP $\mathscr{M}_\star = (X_\star, U_\star, T_\star)$ along with the starting state $x_{\star 0}$, and the reward function $\mathsf{R}_\star$. Since the reward function $\mathsf{R}_\star$ is sparse, it may not be computationally effective in the RL. For this reason, we introduce a "shaped" reward function R_κ (parameterised by a hyper-parameter κ) such that for suitable values of κ, optimal policies for R_κ are the same as optimal policies for $\mathsf{R}_\star$, but unlike $\mathsf{R}_\star$ the function R_κ is dense.

The function R_κ is defined based on the structure of co-safety automaton $\mathscr{A}_\phi$. Let $d(q)$ be the minimum distance of the state q to the unique accepting state q_F. Let $d_{\mathtt{max}} = 1 + \max_{q \in Q}\{d(q) : d(q) < \infty\}$. If there is no path from q to q_F, let $d(q)$ be equal to $d_{\mathtt{max}}$.

We define the *potential function* $P : \mathbb{N} \to \mathbb{R}$ as the following:

$$P(d) = \begin{cases} \kappa^{\frac{d-d(q_0)}{1-d_{\mathtt{max}}}}, & \text{for } d > 0, \\ 1, & \text{for } d = 0, \end{cases}$$

where κ is a constant hyper-parameter. Note that the potential function of the initial state is $P(d(q_0)) = 0$ and the potential function of the final state is $P(d(q_F)) = 1$. Note that

$$P(1) - P(d_{\mathtt{max}}) = \kappa.$$

We define the "shaped" reward function $\mathsf{R}_\kappa : \hat{X} \times \hat{U} \times \hat{X} \to \mathbb{R}$ as the difference between potentials of the destination and of the target states of transition of the automaton, *i.e.,*

$$\mathsf{R}_\kappa((x,q),v,(x',q')) = P(d(q')) - P(d(q)).$$

Moreover, notice that for every run $r = (x_0,q_0),v_1,(x_1,q_1),\ v_2,\ldots,v_n,(x_n,q_n)$ of $\mathscr{M}_\star$, its accumulated reward is simply the potential difference between the last and the first states, *i.e.,* $P(d(q_n)) - P(d(q_0))$.

Theorem 4.4 (Correctness of reward shaping) *For every product MDP $\mathscr{M}_\star = (X_\star, U_\star, T_\star)$ with initial state $x_{\star 0}$ and reward function $\mathsf{R}_\star$, there exists $\kappa_\star > 0$ such that for all $\kappa < \kappa_\star$ we have that the set of optimal expected reward policies for $\mathscr{M}_\star$ is the same as the set of optimal expected reward policies for $\mathscr{M}_\star$ with reward function R_κ.*

Proof 4.1 *First we note that for the optimality, it is sufficient [124] to focus on positional strategies. Let η_1 and η_2 be two positional strategies such that the optimal probability of reaching the final state q_F for η_1 is greater than that for η_2. We write p_1 and p_2 for these probabilities and $p_1 > p_2$. Notice that these probabilities are equal to the optimal expected reward with the $\mathsf{R}_\star$ reward function.*

We denote the expected total reward for policies η_1 and η_2 for the shaped reward function R_κ as s_1 and s_2, respectively. These rewards satisfy the following inequalities:

$$\begin{aligned} s_1 &\geq p_1(P(0)-P(d(q_0))) + (1-p_1)(P(d_{max})-P(d(q_0)), \\ s_2 &\leq p_2(P(0)-P(d(q_0))) + (1-p_2)(P(1)-P(d(q_0))). \end{aligned}$$

Now consider:

$$s_1 - s_2$$

$$
\begin{aligned}
&\geq \big(p_1(P(0)-P(d(q_0)))+(1-p_1)(P(d_{max})-P(d(q_0))\big) \\
&\quad -\big(p_2(P(0)-P(d(q_0)))+(1-p_2)(P(1)-P(d(q_0)))\big) \\
&= \big(p_1(1-P(d(q_0)))+(1-p_1)(P(d_{max})-P(d(q_0))\big) \\
&\qquad -\big(p_2(1-P(d(q_0)))+(1-p_2)(P(1)-P(d(q_0)))\big) \\
&= \big(p_1+(1-p_1)P(d_{max})-P(d(q_0))\big) \\
&\qquad -\big(p_2+(1-p_2)P(1)-P(d(q_0)))\big) \\
&= \big(p_1+(1-p_1)P(d_{max})\big)-\big(p_2+(1-p_2)P(1)\big) \\
&= \big(p_1+(1-p_2)P(d_{max})-(p_1-p_2)P(d_{max})\big) \\
&\qquad -\big(p_2+(1-p_2)P(1)\big) \\
&= (p_1-p_2)+(1-p_2)(P(d_{max})-P(1))-(p_1-p_2)P(d_{max}) \\
&= (p_1-p_2)-(1-p_2)\kappa-(p_1-p_2)P(d_{max}) \\
&= (p_1-p_2)-\kappa\big((1-p_2)+(p_1-p_2)P(d_{max})\big) \\
&\geq (p_1-p_2)-\kappa.
\end{aligned}
$$

We used $p_2 \geq 0$, $p_1-p_2>0$, and $P(d_{max})\leq 0$ to conclude the last inequality. It can be verified that if $\kappa < p_1-p_2$ then $s_1 > s_2$. Therefore, if η_1 is an optimal positional strategy, and η_2 is one of the next best positional strategies, choosing $\kappa_* < p_1-p_2$ guarantees that an optimal strategy in $\mathscr{M}_\kappa$ is also optimal for $\mathscr{M}_\star$, which concludes the proof. ■

Theorem 4.4 demonstrates one way to shape rewards such that the optimal policy remains unaffected while making the rewards less sparse. Along similar lines, one can construct a variety of potential functions and corresponding shaped rewards with similar correctness properties. Of course, the reward shaping scheme presented here is no silver bullet: we expect the performance of different potential functions to be incomparable along a carefully chosen ensemble of MDPs. Since rewards are shaped without any knowledge of the underlying MDP, there may be MDPs where un-shaped rewards may work as well or even better than a given shaped reward. We envisage that the ability to combine several competing ways to shape reward may work better in practice. While sparse rewards may be sufficient for simpler learning tasks, we demonstrate that shaped rewards such as the one provided here are crucial for larger case studies such as the BMW case-study reported in the case study section.

4.6.4 Discussion

Before proceeding with the second part of the chapter, we elaborate on the dimension dependency in our proposed RL techniques compared to the abstraction-based ones. Assuming

a uniform quantiser, the finite MDP constructed in Subsection 4.5.1 is a matrix with a dimension of $(n_x \times n_u) \times n_x$, where n_u is the cardinality of the finite input set U. Computing this matrix is one of the bottlenecks in abstraction-based approaches since an n-dimensional integration has to be done numerically for each entries of this matrix. Moreover, n_x (*i.e.,* the cardinality of the finite state set) grows exponentially with the dimension n. Once this matrix is computed, it is employed for the dynamic programming on a vector of the size $(n_x \times n_u)$. This is a second bottleneck of the process. On the other hand, by employing the proposed RL approach, the curse of dimensionality reduces to only *learning* the vectors of size $(n_x \times n_u)$ without having to compute the full matrix. Moreover, the abstraction-based techniques need to precisely know the probabilistic evolution of states in models, whereas in this work we only need to know the Lipschitz constant $\mathscr{H}$, which can be readily estimated from sample trajectories of Σ.

Although the proposed RL framework has the above-mentioned merits compared to the abstraction-based techniques, it may not be efficient enough while dealing with large-scale stochastic systems. In particular, when a system is described as an interconnection of subsystems, the proposed approach produces a centralised controller operating on the monolithic states of the interconnected systems. This does not take advantage of the system topology and suffers from the curse of dimensionality. In the rest of the chapter, we propose synthesising a set of decentralised controllers by solving a stochastic game for each subsystem (cf. (4.3.2)) using multi-agent reinforcement learning [97]. We derive performance bounds on the composed controller from the bounds computed based on the individual controllers.

4.7 Controller synthesis for networks of unknown stochastic control systems

In this part we generalise our results by proposing a *compositional approach* for the controller synthesis of *networks* for unknown continuous-space stochastic control systems under finite-horizon specifications. We apply a model-free *two-player* RL in an assume-guarantee fashion and *compositionally* compute policies over finite horizons for continuous-space interconnected systems without explicitly constructing their finite-state abstractions. We then propose a lower bound for the optimality guarantee of synthesised controllers when applied to the interconnected system based on those of individual controllers.

Given discrete-time stochastic subsystems $\Sigma = (X, U, W, \varsigma, f, Y, h)$, where f, h, and distribution of ς are unknown, and given finite-horizon specifications φ for them[1], we

[1]We dropped index i from Σ and φ for the sake of simple presentation.

implicitly abstract each subsystem in the network with a finite MDP with *unknown* transition probabilities. We then synthesise policies over each abstract MDP in an assume-guarantee fashion using RL and map the results back over the concrete network while providing guarantees on *satisfaction probability*. In particular, we propose a lower bound on the satisfaction probability of synthesised policies when applied to the interconnected system based on those of individual policies applied to subsystems.

In order to provide any formal guarantee, we assume that the system in (4.3.2) is Lipschitz-continuous with respect to states and internal inputs with constants $\mathscr{H}_x$ and $\mathscr{H}_w$, respectively. The next example provides a systematic way of computing $\mathscr{H}_x$ and $\mathscr{H}_w$ for a class of linear continuous-space stochastic control subsystems.

Example 4.2 *Consider a dt-SCS Σ with linear dynamics $x(k+1) = Ax(k) + \bar{B}\upsilon(k) + Dw(k) + \varsigma(k)$, for some matrices $A = [a_{ij}]$, $\bar{B}$, and $D = [d_{ij}]$ of appropriate dimensions, where $\varsigma(k)$ is i.i.d. for $k = 0,1,2,\ldots$ with normal distribution, zero mean and covariance matrix* $\mathrm{diag}(\sigma_1,\ldots,\sigma_n)$. *Then, one obtains the Lipschitz constants of the stochastic kernel with respect to states and internal inputs as $\mathscr{H}_x = \sum_{i,j} \frac{2|a_{ij}|}{\sigma_i\sqrt{2\pi}}$ and $\mathscr{H}_w = \sum_{i,j} \frac{2|d_{ij}|}{\sigma_i\sqrt{2\pi}}$, respectively.*

Now, we state the main problem that we aim to solve in this section. In particular, this problem is an extension of Problem 4.1 to allow a more efficient solution and make it more scalable when the underlying system is an interconnection of many subsystems.

Problem 4.2 *Let φ_i be finite-horizon objectives and $\Sigma_i = (X_i, U_i, W_i, \varsigma_i, f_i, Y_i, h_i)$ continuous-space subsystems, where f_i, h_i and distribution of ς_i are unknown, but Lipschitz constants $\mathscr{H}_{x_i}$ and $\mathscr{H}_{w_i}$ are known, for $1 \leq i \leq N$. Synthesise Markov policies that satisfy φ_i in Σ_i with probabilities within a-priori defined thresholds ε_i from unknown optimal ones. Then, provide a lower bound on the satisfaction probability of synthesised controllers when applied to the interconnected system based on those of individual controllers applied to subsystems.*

To present our solution to this problem, we first report the following result [164, 147] to show the closeness between a continuous-space subsystem Σ and its finite abstraction $\widehat{\Sigma}$ in a probabilistic setting. We then leverage this result in Section 4.8 to provide a two-player stochastic game RL-based solution to Problem 4.2. Note that all (Markov) policies for $\widehat{\Sigma}$ are also (Markov) policies for Σ, *i.e.,* $\widehat{\Pi}_{\mathrm{Max}} \subseteq \Pi_{\mathrm{Max}}$ and $\widehat{\Pi}_{\mathrm{Min}} \subseteq \Pi_{\mathrm{Min}}$.

Corollary 4.1 *Let $\Sigma = (X, U, W, \varsigma, f, Y, h)$ be a continuous-space subsystem and $\widehat{\Sigma} = (\hat{X}, \hat{U}, \hat{W}, \varsigma, \hat{f}, \hat{Y}, \hat{h})$ be its finite abstraction. For a given finite-horizon objective φ, initial*

state $x \in X$, and Markov policy pair $(\rho,\xi) \in \widehat{\Pi}_{\text{Max}} \times \widehat{\Pi}_{\text{Min}}$ for the closed-loop $\widehat{\Sigma}$ (denoted by the solution process $\widehat{\Sigma}^x_{(\rho,\xi)}$), the closeness between the continuous-space subsystem and its abstraction in terms of satisfaction probability is given by

$$|\mathbb{P}(\Sigma^x_{(\rho,\xi)} \models \varphi) - \mathbb{P}(\widehat{\Sigma}^{\hat{x}}_{(\rho,\xi)} \models \varphi)| \leq \varepsilon, \tag{4.7.1}$$

$$\text{with } \varepsilon := \mathscr{T}\mathscr{L}(\delta\mathscr{H}_x + \mu\mathscr{H}_w),$$

where $\hat{x} = Q_x(x)$, $\mathscr{T}$ is the time horizon, δ and μ are respectively state and internal input discretisation parameters, $\mathscr{H}_x$ and $\mathscr{H}_w$ are respectively Lipschitz constants of the stochastic kernel with respect to states and internal inputs, and $\mathscr{L}$ is the Lebesgue measure of (bounded) state space.

Note that if the state space is unbounded, we assume that the specification requires the system to stay in a safe bounded subset of the state space, and this bounded subset can be used in the above corollary. Next, we consider networks of stochastic control subsystems and provide a lower bound for the *satisfaction probability* of synthesised controllers when applied to the interconnected system based on those of individual controllers applied to subsystems as in Corollary 4.1.

Theorem 4.5 *Let $\Sigma = (X,U,\varsigma,f)$ be an interconnected continuous-space stochastic control system and $\widehat{\Sigma} = (\hat{X},\hat{U},\varsigma,\hat{f})$ be its finite abstraction. For a given finite-horizon objective $\varphi = \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_N$, we have*

$$\mathbb{P}(\Sigma^x_{\rho^*} \models \varphi) \geq \prod_{i=1}^{N} \min_{\xi_i \in \Pi^{\Sigma_i}} \mathbb{P}((\widehat{\Sigma}_i)^{x_i}_{(\rho_i^*,\xi_i)} \models \varphi_i) - \frac{1}{2}[(1+\varepsilon)^N - (1-\varepsilon)^N], \tag{4.7.2}$$

where $\varepsilon := \max_i \varepsilon_i$, the state $x \in X$ is the initial state of the interconnected system, for every subsystem $1 \leq i \leq N$ the state $x_i \in X_i$ is its initial state, and $\rho^* \in \Pi^{\Sigma}$ is the policy obtained by composing the optimal policies $\rho_i^* \in \Pi^{\Sigma_i}_{\text{Max}}$ for an objective φ_i.

Proof 4.2 *We have*

$$\mathbb{P}(\Sigma^x_{\rho^*} \models \varphi) \geq \prod_{i=1}^{N} \min_{\xi_i \in \Pi^{\Sigma_i}} \mathbb{P}((\Sigma_i)^{x_i}_{(\rho_i^*,\xi_i)} \models \varphi_i).$$

This inequality holds due to the dynamic programming (DP) formulation of $\mathbb{P}(\Sigma^x_{\rho^*} \models \varphi)$ *[153, 154] and the following property of the Bellman operator used in each iteration of DP:*

$$\begin{aligned}
&\int_{X_1}\int_{X_2} V_1(\bar{x}_1)V_2(\bar{x}_2)T_1(d\bar{x}_1\,|\,x_1,x_2)T_2(d\bar{x}_2\,|\,x_1,x_2)\\
&=\int_{X_1} V_1(\bar{x}_1)T_1(d\bar{x}_1\,|\,x_1,x_2)\int_{X_2} V_2(\bar{x}_2)T_2(d\bar{x}_2\,|\,x_1,x_2)\\
&\geq \min_{w_1}\int_{X_1} V_1(\bar{x}_1)T_1(d\bar{x}_1|x_1,w_1)\min_{w_2}\int_{X_2} V_2(\bar{x}_2)T_2(d\bar{x}_2|w_2,x_2),
\end{aligned}$$

where T_i *is the stochastic kernel of* Σ_i *and* V_i *is the value function used in each iteration of DP, for* $i \in \{1,2\}$*. This inequality allows us to consider the effect of other subsystems in the worst case and get a lower bound on the solution.*

Since for $i = 1,2,\ldots,N$*,*

$$\left|\min_{\xi_i\in\Pi^{\Sigma_i}} \mathbb{P}((\Sigma_i)^{x_i}_{(\rho_i^*,\xi_i)} \models \varphi_i) - \min_{\xi_i\in\Pi^{\Sigma_i}} \mathbb{P}((\widehat{\Sigma}_i)^{x_i}_{(\rho_i^*,\xi_i)} \models \varphi_i)\right| \leq \varepsilon_i.$$

one can write

$$\begin{aligned}
\mathbb{P}(\Sigma^x_{\rho^*} \models \varphi) &\geq \prod_{i=1}^{N}(\min_{\xi_i\in\Pi^{\Sigma_i}} \mathbb{P}((\widehat{\Sigma}_i)^{x_i}_{(\rho_i^*,\xi_i)} \models \varphi_i) - \varepsilon_i)\\
&\geq \prod_{i=1}^{N}\min_{\xi_i\in\Pi^{\Sigma_i}} \mathbb{P}((\widehat{\Sigma}_i)^{x_i}_{(\rho_i^*,\xi_i)} \models \varphi_i) - \left[\binom{N}{1}\varepsilon + \binom{N}{3}\varepsilon^3 + \ldots\right]\\
&= \prod_{i=1}^{N}\min_{\xi_i\in\Pi^{\Sigma_i}} \mathbb{P}((\widehat{\Sigma}_i)^{x_i}_{(\rho_i^*,\xi_i)} \models \varphi_i) - \frac{1}{2}[(1+\varepsilon)^N - (1-\varepsilon)^N],
\end{aligned}$$

where $\varepsilon = \max_i \varepsilon_i$ *and it completes the proof.* ■

4.8 Compositional controller synthesis via reinforcement learning

According to Corollary 4.1, one can construct a finite abstraction $\widehat{\Sigma}_i$ from a given continuous-space subsystem Σ_i with known stochastic kernels such that the optimal probability of satisfaction of a finite-horizon specification φ_i with horizon $\mathscr{T}$ in $\widehat{\Sigma}_i$ is no more than $2\varepsilon_i$-worse than the optimal strategy in Σ_i. Hence, given subsystems Σ_i with known stochastic kernels, properties φ_i, and time horizon $\mathscr{T}$, $2\varepsilon_i$-optimal strategies to satisfy φ_i in $\mathscr{T}$ steps

can be computed using a suitable finite SGA with δ_i and μ_i as state and internal input discretisation parameters for subsystems.

When the stochastic kernels are unknown, Corollary 4.1 still provides the correct probabilistic bound given discretisation parameters δ_i and μ_i if Lipschitz constants $\mathscr{H}_{x_i}$ and $\mathscr{H}_{w_i}$ are known. This observation enables us to employ two-player RL on the underlying discrete SGA, denoted by $\widehat{\Sigma}_{\delta_i}$, without explicitly constructing the abstraction by restricting observations of the reinforcement learner to the closest representative point in the discrete set of states. Similarly to Section 4.6.1, we can construct the product of a SGA and a DFA. Consequently, any converging RL algorithm for two-player stochastic games over such finite observation set then converges to a $2\varepsilon_i$-optimal strategy for the concrete dt-SCS Σ_i (cf. Corollary 4.1). We summarise our proposed solution in the following theorem.

Theorem 4.6 *Let φ_i be given finite-horizon properties, $\varepsilon_i > 0$, and $\Sigma_i = (X_i, U_i, W_i, \varsigma_i, f_i, Y_i, h_i)$ be continuous-space subsystems, where f_i, h_i and distribution of ς_i are unknown, but Lipschitz constants $\mathscr{H}_{x_i}$ and $\mathscr{H}_{w_i}$ are known for $i \in \{1, \ldots, N\}$. For discretisation parameters δ_i and μ_i satisfying $\mathscr{T}_i\mathscr{L}_i(\delta_i\mathscr{H}_{x_i} + \mu_i\mathscr{H}_{w_i}) \leq \varepsilon_i$, a convergent model-free reinforcement learning algorithm (*e.g., *minimax-Q learning [98]) for two-player stochastic games over $\widehat{\Sigma}_{\delta_i}$ converges to a $2\varepsilon_i$-optimal strategy for subsystems Σ_i. Accordingly, one can compute a lower bound for the satisfaction probability of the synthesised controllers applied to the interconnected system based on* (4.7.2).

We reiterate that in our proposed setting, we do not need to compute transition probabilities $\hat{T}_X$ since we directly learn the value functions using the proposed RL.

4.8.1 Accelerating RL with multi-level discretisation

The efficiency of tabular RL algorithms is directly connected to the size of the state set of the finite control system—with larger state sets typically requiring longer training times. For instance, if two different agents are trained on a coarse discretisation and a fine discretisation of the same system, then the former will typically have shorter training times at the cost of higher discretisation error. However, since the underlying continuous system is the same, the two agents will learn roughly similar policies. We propose first training an RL agent on a coarse discretisation of the system and then using the resulting policy to initialise training for a different RL agent on a finer discretisation of the same system. By repeating this process with increasingly fine discretisation levels, we reach the final desired discretisation level. Our experimental results demonstrate that this multi-level discretisation scheme dramatically accelerates the learning process.

The proposed multi-level discretisation algorithms begin by creating a coarse discretisation of the continuous system. It then trains a minimax-Q learning agent for a fixed time. After this time has elapsed, it decreases the discretisation parameter to create a more finely discretised system by, for instance, halving the discretisation parameter. It then creates a new learning agent for the more finely discretised problem. The values for the new Q-table get initialised from the old Q-table where the value of a state inherits its values from the corresponding nearest state in the previous discretisation. This new agent is then trained for a fixed time and is used to initialise a new agent on an even finer discretisation of the system. This process gets repeated until we reach the final desired discretisation level. Note that as long as we stop refining the discretisation at some point, we retain in the limit the convergence of minimax-Q learning since the minimax-Q learning algorithm will converge from an arbitrarily initialised Q-table.

4.9 Case studies

To demonstrate the effectiveness of the proposed approaches, we apply them to three physical benchmarks, including (i) regulation of a room temperature (network), (ii) control of a road traffic (network), and (iii) control of a 7-dimensional nonlinear model of a BMW 320i car. We note that application of formal abstraction-based methods is studied previously in smart grids [176], temperature regulation of smart buildings [148, 151, 77, 43], and transportation networks [3, 155] for either small-scale models or aggregate models without coupled dynamics.

4.9.1 Room temperature (network)

Monolithic system. We first apply our results to the temperature regulation of a room equipped with a heater. The model of this case study is adapted from [109] by including stochasticity in the model in the form of an additive noise.

The goal is to synthesise a controller for Σ using Theorem 4.3, so that the controller maintains the temperature of the room in the safe set $[19,21]$ for at least 10 steps.

Network of systems. We then apply the compositional approach proposed in the second part of the chapter to a network of $N = 20$ rooms with a circular topology (cf. Fig. 4.4). We employ Theorem 4.6 and synthesise a controller for Σ via its implicit abstract subsystems $\widehat{\Sigma}_{\delta_i}$, so that the controller maintains the temperature of any room in the safe set $[17,18]$ for at least 45 minutes.

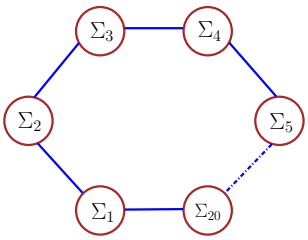


Fig. 4.4 A circular building in a network of 20 rooms [88].

4.9.2 Road traffic (network)

Monolithic System. We also apply our results to a road traffic system containing a cell with 2 entries and 1 way out, as schematically depicted in Fig. 4.5. The model of this case study is taken from [93]; stochasticity is included in the model as an additive noise. One of the

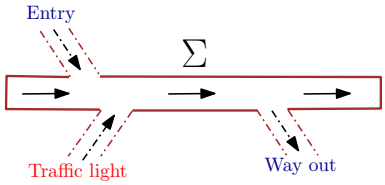


Fig. 4.5 Model of a road traffic control with the length of 500 meters, 1 way out, and 2 entries, one of which is controlled by a traffic light [88].

entries of the cell is controlled by a traffic light, denoted by $v \in \{0,1\}$, that enables (green light) or not (red light) the vehicles to pass.

We synthesise a controller for Σ using Theorem 4.3, so that traffic density is less than 20 vehicles for at least 10 steps.

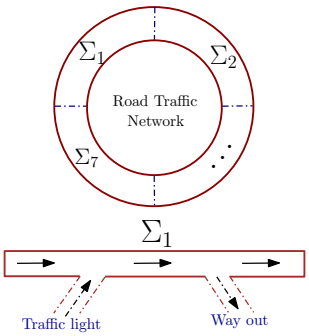


Fig. 4.6 Model of a road traffic network in a ring composed of 7 identical cells, each of which has 1 entry and 1 exit [88].

Network of systems. We apply our compositional approach to a road traffic ring network that consists of $N = 7$ identical cells, each of which has 1 entry and 1 exit, as schematically

Table 4.1 Q-Learning Results for Room Temperature and Road Traffic [88].

	Room					Traffic				
δ	p_r	p_*	ε	p_l	p_h	p_r	p_*	ε	p_l	p_h
0.01	0.969	0.975	0.246	0.728	1.0	0.985	0.999	0.016	0.983	1.0
0.02	0.974	0.975	0.493	0.481	1.0	0.997	0.999	0.031	0.967	1.0
0.05	0.954	0.975	1.233	0.000	1.0	0.999	0.999	0.079	0.919	1.0
0.1	0.977	0.975	2.467	0.000	1.0	0.999	0.999	0.159	0.839	1.0
0.2	0.973	0.974	4.935	0.000	1.0	0.999	0.999	0.319	0.680	1.0

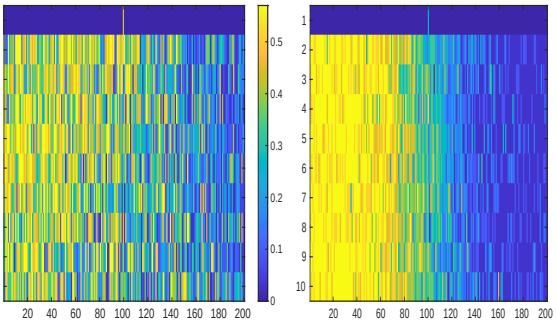


Fig. 4.7 Room temperature control: A heat-map visualisation of strategies learned via Reinforcement Learning after 10^5 episodes (left) and after $8{\cdot}10^6$ episodes (right). The x axis shows the room temperature in ℃, while the y axis shows time steps $1 \leq k \leq 10$. The action selected by the strategy is in the input set $\{0.03, 0.09, 0.15,\ 0.21, 0.27, 0.33, 0.39, 0.45, 0.51, 0.57\}$ and is color-coded according to the map shown in the middle: Bright yellow and deep blue represent maximum and minimum heat. In the first step, strategies are only defined for the initial state; this causes the blue bands at the top [88].

depicted in Fig. 4.6. By leveraging Theorem 4.6, we synthesise a controller for Σ via its implicit abstract subsystems $\widehat{\Sigma}_{\delta_i}$ so that the controller keeps traffic density below 20 vehicles per cell for at least 36 seconds.

4.9.3 Learning Controllers

Monolithic analysis. Table 4.1 shows a comparison of Q-learning to the computed optimal probabilities for the room temperature and road traffic examples. For each model, five different discretisation steps (δ) are considered and for each value of δ, probabilities of satisfaction of the finite-horizon objectives are reported in the columns labelled p_r. These probabilities are Q-values of the initial state of the finite-state MDP for the policy computed by Q-learning after 10^6 episodes. The objective is to keep the system safe for at least 10 steps. For comparison, the optimal probability p_* for a time-dependent policy is reported assuming that we know the exact dynamics for these two examples. Note that we compute p_* by dynamic programming over the constructed finite MDPs as proposed in Subsection 4.5.1 to verify our results. The optimal probability p_* reported in Table 4.1 corresponds to the

Table 4.2 Results for distributed controller learned by minimax Q-learning on the quantised subsystems [88].

	p^+	p_{low}	ε	p_{sampled}
Room	0.999943 0.999952 $\pm$ 0.000014	0.902585 0.902769 $\pm$ 0.000272	0.004807	0.998880 $\pm$ 0.000066
Traffic	0.996837 0.998878 $\pm$ 0.000066	0.932064 0.946167 $\pm$ 0.000459	0.006571	0.999999 $\pm$ 0.000002

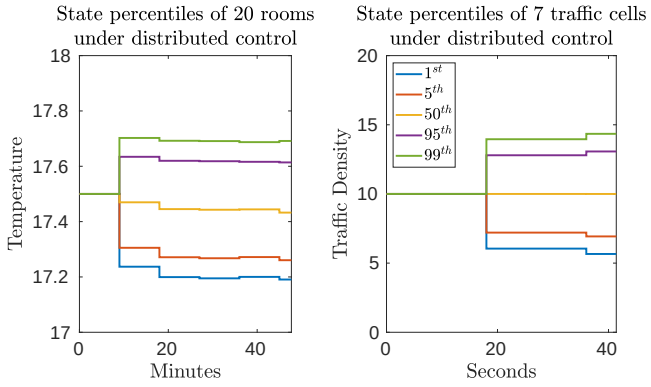


Fig. 4.8 State evolution of the learned distributed controllers visualised through percentiles from 10^6 sampled trajectories [88].

same initial condition that is utilised in the learning process. The optimal probability for the original *continuous-space* stochastic system is always within an interval $[p_l, p_h]$ centred at p_* and with a radius ε as reported in Table 4.1. One can readily see from Table 4.1 that as the discretisation parameter δ decreases, the size of this interval shrinks, which implies that the optimal probability for the original *continuous-space* stochastic system converges to p_*. While finer abstractions give better theoretical guarantees, for a fixed number of episodes it is easier to learn good strategies for coarser abstractions. This is reflected in Table 4.1, where the values of p_r do not necessarily get better with smaller values of δ. However, by increasing the number of episodes, strategies converge toward the optimal one, as illustrated in Fig. 4.7, which visualises room temperature control strategies computed by Q-learning after different numbers of episodes. Note that in Table 4.1, the error bound ε exceeds 1 for $\delta \geq 0.05$ in the room temperate control example, which is not a useful probability bound for the *continuous-space* system. However, we report the corresponding values of p_r and p_* so that they may still be compared.

Compositional analysis. Following the motivation in Sections 4.7 and 4.8, we synthesise a controller for Σ in both case studies by first producing implicitly the abstract subsystems

$\widehat{\Sigma}_{\delta_i}$. We then learn a controller for the resulting stochastic game with minimax Q-learning. To accelerate learning, we use the multi-level discretisation scheme described in Section 4.8.1. For the room temperature control and road traffic network, the final discretisation values are $\delta_i = 0.001$, $\mu_i = 0.1$, and $\delta_i = 0.05$, $\mu_i = 0.01$, respectively. For the room temperature control, we use 1.5 million episodes, a learning rate of 0.04 decayed linearly to 0.02, an exploration rate of 0.1, and a discount factor of 1. This takes approximately 5 minutes of wall-clock time. For the road traffic network, we use 2 million episodes, a learning rate of 0.1 decayed linearly to 0.02, an exploration rate of 0.2, and a discount factor of 1. This takes approximately 4 minutes of wall-clock time.

Table 4.2 shows the results for the learned controllers: p^+ is the (approximate) probability of the learned policy satisfying the finite-horizon objective over the subsystem against an optimal adversarial internal input, ε is the bound on the quantised measurement error from equation (4.7.1), p_{low} is the lower bound from equation (4.7.2) on the probability of the decentralised controller satisfying the finite-horizon objective over the interconnected system, and p_{sampled} is a 95% confidence bound on the probability of satisfying the finite-horizon objective using the decentralised controller as computed via 10^6 samples. We compute p^+ in two ways. First, we approximate p^+ without knowledge of the model by fixing the controller policy that results from learning and producing a 95% confidence bound on the probability of satisfying the objective from 10^6 samples. Second, we fix the controller policy that results from learning and compute an optimal strategy for the internal input by dynamic programming. This requires knowledge of the model and is done to validate the results of learning.

Table 4.2 shows that on these case studies, the computed bound on the probability of the decentralised controller satisfying the finite-horizon objective is within 0.1 of the estimated probability using samples for both examples. Additionally, there is a successful mitigation of the curse of dimensionality versus synthesising a centralised controller for the interconnected system monolithically. On the room temperature example, the selected quantisation parameters result in $n_x = 1000$ states and $n_w = 20$ internal inputs for each subsystem. Combined with $n_\upsilon = 6$ control inputs and a time horizon of $\mathscr{T} = 5$, there are $\mathscr{T}(n_x n_\upsilon + n_x n_\upsilon n_w) = 630{,}000$ total state-input pairs in the stochastic game which we need to solve to produce the decentralised controller. For comparison, we can select appropriate quantisation parameters which yield the same quantisation error in the compositional case, $\frac{1}{2}[(1+\varepsilon)^N - (1-\varepsilon)^N]$, as in the monolithic case, ε. The appropriate quantisation results in $\hat{n}_x = 9$ states for each individual subsystem. Even with only a few states required in each quantised subsystem, the monolithic approach still needs to reason over $\mathscr{T}(\hat{n}_x n_\upsilon)^{20} \approx 2.22 \cdot 10^{35}$ state-input pairs to produce a controller. For the road traffic example, there are

$n_x = 400$ states, $n_w = 2000$ internal inputs, $n_\upsilon = 2$ control inputs, and a time horizon of $\mathscr{T} = 2$. We require $\hat{n}_x = 21$ states per subsystem for producing a monolithic controller with the same quantisation error as in the compositional case. We get that there are $\mathscr{T}(n_x n_\upsilon + n_x n_\upsilon n_w) =$ 3,201,600 state-input pairs in the stochastic game, and $\mathscr{T}(\hat{n}_x n_\upsilon)^7 \approx 4.61 \cdot 10^{11}$ state-input pairs in the monolithic setting.

4.9.4 7-dimensional BMW 320i car

The previous case-studies are representative of what can be solved by discretisation and tabular methods like Q-learning. Relaxing those constraints, we were able to apply deep deterministic policy gradient (DDPG) [96] to a 7-dimensional *nonlinear* model of a BMW 320i car [5] to synthesise a reach-while-avoid controller. Though convergence guarantees are not available for DDPG and for most RL algorithms with nonlinear function approximations, breakthroughs in this direction will expand the applicability of our results to more complex safety-critical applications. For instance, the authors in [80] propose an approach for maximising the probability of satisfying LTL specifications over unknown stochastic processes with continuous state spaces. The results in [80] show that under some regularity assumptions, the learned policy converges to the optimal one given the convergence of the RL algorithm.

The model of the BMW 320i case study is borrowed from [5, Section 5.1] by discretising the dynamics in time and including a stochasticity inside the dynamics as additive noises. We are interested in the autonomous operation of the vehicle on a highway. Consider a situation on a two-lane highway when an accident suddenly happens on the same lane on which our vehicle is travelling. The vehicle's controller should find a safe manoeuvre to avoid the crash with the next-appearing obstacle. Details of the dynamics of the vehicle and its specification can be found in [89].

Fig. 4.9 shows the simulation from 100 samples with varying initial positions and initial heading velocities (16–18 $[m/s]$) for the learned controller. We employed potential-based reward shaping to speed-up learning in this case study from 10K episodes (no success) to under 5K episodes (for a convincing learning, see Fig. 4.9).

4.10 Conclusion

In this chapter, we proposed a policy synthesis approach for continuous-space stochastic systems with unknown dynamics. The goal of the policy was to maximise the probability that the system satisfies a complex property written in a fragment of LTL. Our approach

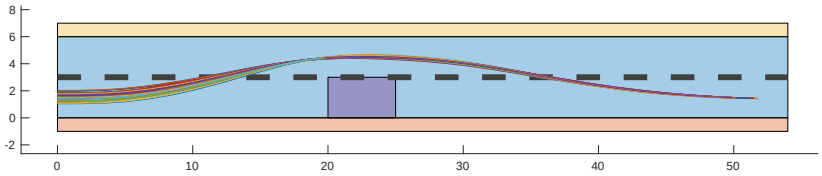


Fig. 4.9 Trajectories of 100 simulations of the RL-synthesised controller for a 7-dimensional model of a BMW 320*i* car trained using DDPG. The road segment is 6 meter wide and 50 meter long; the length of the car is 4.508 meters and its width is 1.610 meters [88].

replaced the unknown system with a finite MDP without explicitly constructing it. Since transition probabilities of the MDP are unknown, we utilised RL to find a policy and apply it to the original continuous-space system. We showed that any converging RL algorithm over such a finite MDP converges to a 2ε-optimal strategy over the concrete continuous-space system with unknown dynamics, where ε is defined a-priori and can be controlled. Since automata-based reward functions are often sparse, we also presented a potential-based *reward shaping* technique to produce dense rewards and speed up the learning procedure. We also proposed a *compositional approach* for the policy synthesis of *networks* of unknown stochastic systems using a novel two-player RL scheme. We proposed a lower bound for the probability of satisfaction of a finite-horizon property by the interconnected system based on those of subsystems. We finally applied our approaches to three physical case studies.

Chapter 5

Formal policy synthesis for continuous-state systems via reinforcement learning

5.1 Chapter introduction

In previous chapters the procedure of designing a controller includes an abstraction step and then we synthesised the controller for a finite state system. This chapter studies satisfaction of temporal properties on unknown stochastic processes that have continuous state spaces without an abstraction step.

We show how reinforcement learning (RL) can be applied for computing policies that are finite-memory and deterministic using only the paths of the stochastic process. We address properties expressed in linear temporal logic (LTL) and use their automaton representation to give a path-dependent reward function maximised via the RL algorithm. We develop the required assumptions and theories for the convergence of the learned policy to the optimal policy in the continuous state space. To improve the performance of the learning on the constructed sparse reward function, we propose a sequential learning procedure based on a sequence of labelling functions obtained from the positive normal form of the LTL specification. We use this procedure to guide the RL algorithm towards a policy that converges to an optimal policy under suitable assumptions on the process. We demonstrate the approach on a 4-dim cart-pole system and 6-dim boat driving problem.

The research presented in the chapter has been accepted and presented in the IFM Conference [80]. My role in this research was to provide the theoretic results, simulate the case studies and write the paper.

5.2 Introduction

Motivations. Omega-regular languages provide a rich formalism to unambiguously express desired properties of the system. Linear temporal logic (LTL), as a class of omega-regular languages, is widely used for task specification such as safety, liveness, and repeated reachability. Synthesising policies formally for a system to satisfy a specification requires the knowledge of a model of the system. Extensive techniques are developed in the literature for different classes of models including finite-space models [8] and continuous-state or hybrid models [53, 89, 104, 107]. Reinforcement learning (RL) is a promising paradigm for sequential decision making when a model of the system is not available or is very hard to construct and analyse. The objective of an RL algorithm is to find suitable action policies in order to maximise the collected rewards that depend on the states and actions taken at those states. The RL algorithms are in particular useful when the total collected reward has an additive structure.

Many objectives including satisfaction of omega-regular properties on stochastic systems do not admit an equivalent additive reward structure. A natural approach used in the literature (e.g., [92]), is to use heuristics for assigning additive rewards and then apply RL algorithms to obtain a policy. Unfortunately, there is no unique procedure for constructing these rewards and the learning does not necessarily converge to the optimal policy. Due to all of these limitations, there is a need to provide data-driven algorithms that do not require any heuristics and have suitable convergence guarantees to policies that are optimal for satisfaction of temporal properties.

Main contributions. We apply RL algorithms to *continuous-state* stochastic systems using only paths of the system to find optimal policies satisfying an LTL specification. We show that if a suitable assumption on the system holds, the formulated optimal average reward converges linearly to the true optimal satisfaction probability. We use negation of the specification and learn a lower bound on this satisfaction probability. To improve the performance of the learning on the constructed sparse reward function, we show how to construct a sequence of labelling functions based on the positive normal form of the LTL specification and use them for guiding the RL algorithm in learning the policy and its associated value function. This sequential learning is able to find policies for our case studies in less than 1.5 hours but direct learning does not converge in 24 hours.

Organisation. Section 5.3 recalls definition of controlled Markov processes (CMPs) as the unknown model. We also give linear temporal logic, limit-deterministic automata, and the problem statement in the same section. Section 5.4 gives construction of the augmented CMP and the product CMP. It establishes the relation between the reachability on the augmented

CMP and the LTL satisfaction on the original CMP. Section 5.5 gives the reward function for reachability on the augmented CMP that can be used by RL algorithms. It also gives a procedure for guiding the learning task via a sequence of labelling functions. Finally, Section 5.6 illustrates our approach on two case studies, a 4-dim cart-pole system and 6-dim boat driving problem.

5.3 Preliminaries and problem statement

We consider a probability space $(\Omega, \mathscr{F}_\Omega, P_\Omega)$, where Ω is the sample space, $\mathscr{F}_\Omega$ is a sigma-algebra on Ω comprising subsets of Ω as events, and P_Ω is a probability measure that assigns probabilities to events. We assume that random variables introduced in this chapter are measurable functions of the form $X : (\Omega, \mathscr{F}_\Omega) \to (S_X, \mathscr{F}_X)$ from the measurable space $(\Omega, \mathscr{F}_\Omega)$ to a measurable space $(S_X, \mathscr{F}_X)$. Any random variable X induces a probability measure on its space $(S_X, \mathscr{F}_X)$ as $Prob\{A\} = P_\Omega\{X^{-1}(A)\}$ for any $A \in \mathscr{F}_X$. We often directly discuss the probability measure on $(S_X, \mathscr{F}_X)$ without explicitly mentioning the underlying sample space and the function X itself.

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 metrisable space). Examples of a Borel space are the Euclidean spaces $\mathbb{R}^n$, its Borel subsets endowed with a subspace topology, as well as hybrid spaces of the form $Q \times \mathbb{R}^n$ with Q being a finite set. Any Borel space S is assumed to be endowed with a Borel sigma-algebra, which is denoted by $\mathscr{B}(S)$. We say that a map $f : S \to Y$ is measurable whenever it is Borel measurable. We denote the set of non-negative integers by $\mathbb{N} := \{0, 1, 2, \ldots\}$ and the empty set by $\emptyset$.

5.3.1 Controlled Markov processes

Controlled Markov processes (CMPs) are a natural choice for physical systems that have three main features: an uncountable state space that can be continuous or hybrid, control inputs to be designed, and inputs in the form of disturbance which have certain probabilistic behaviour [39].

We consider CMPs in discrete time defined over a general state space, characterised by the tuple $\mathfrak{S} = (\mathscr{S}, \mathscr{A}, \{\mathscr{A}(s) | s \in \mathscr{S}\}, T_\mathfrak{s})$, where $\mathscr{S}$ is a Borel space as the state space of the CMP. We denote by $(\mathscr{S}, \mathscr{B}(\mathscr{S}))$ the measurable space with $\mathscr{B}(\mathscr{S})$ being the Borel sigma-algebra on the state space. $\mathscr{A}$ is a Borel space as the input space of the CMP. The set $\{\mathscr{A}(s) | s \in \mathscr{S}\}$ is a family of non-empty measurable subsets of $\mathscr{A}$ with the property that $\mathscr{K} := \{(s, u) : s \in \mathscr{S}, u \in \mathscr{A}(s)\}$ is measurable in $\mathscr{S} \times \mathscr{A}$. Intuitively, $\mathscr{A}(s)$ is the set

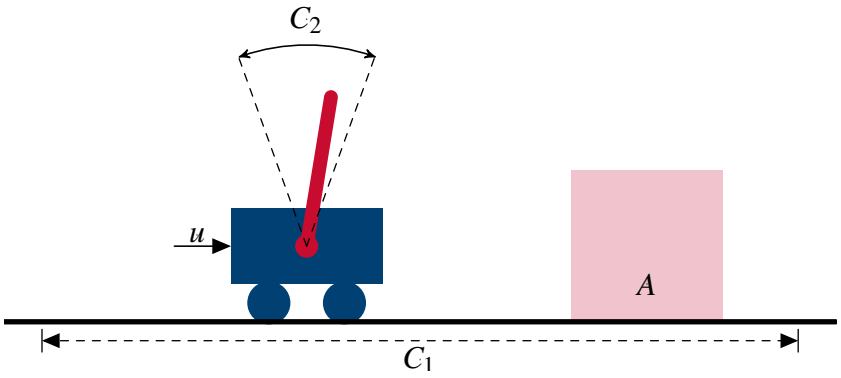


Fig. 5.1 Cart-pole system with a 4-dim state space. It should stay within the limits specified by C_1, always keep the pole upright in the range C_2, and reach the region A [80].

of inputs that are feasible at state $s \in \mathscr{S}$. $T_{\mathfrak{s}} : \mathscr{B}(\mathscr{S}) \times \mathscr{S} \times \mathscr{A} \to [0,1]$, is a conditional stochastic kernel that assigns to any $s \in \mathscr{S}$ and $u \in \mathscr{A}(s)$ a probability measure $T_{\mathfrak{s}}(\cdot|s,u)$ on the measurable space $(\mathscr{S}, \mathscr{B}(\mathscr{S}))$ so that for any set $A \in \mathscr{B}(\mathscr{S}), P_{s,u}(A) = \int_A T_s(ds|s,u)$, where $P_{s,u}$ denotes the conditional probability $P(\cdot|s,u)$.

Example 5.1 *Consider the cart-pole in Figure 5.1. The cart moves along a line in either direction. The states are position s^1, velocity s^2, pole's angle s^3, and the angular velocity s^4. The input u_n is the force applied to the cart at time step n. Its dynamics in discrete time are according to the following 4-dim difference equation:*

$$\begin{cases} s^1_{n+1} = s^1_n + \Delta s^2_n \\ s^2_{n+1} = s^2_n + \Delta a_3 \\ s^3_{n+1} = s^3_n + \Delta s^2_n \\ s^4_{n+1} = s^4_n + \Delta a_2 + \eta_n, \end{cases} \quad \textit{with} \quad \begin{cases} a_3 := a_1 - \dfrac{l a_2 \cos(s^3_n)}{(M+m)} \\ a_2 := \dfrac{g \sin(s^3_n) - \cos(s^3_n) a_1}{l(\frac{4}{3} - m(\cos(s^3_n))^2/(M+m))} \\ a_1 := \dfrac{u_n + l(s^4_n)^2 \sin(s^3_n)}{M+m}. \end{cases} \tag{5.3.1}$$

Δ is the sampling time, M is the mass of the cart, m is the mass of the pole, l is the half length of the pole, and η_n models the disturbance. The cart has discrete input and can be either pushed to the left or right with a fixed value, $\mathscr{A} = \{-F_{max}, F_{max}\}$. This input u_n appears in a_1 that affects both a_2 and a_3. Assuming that the disturbances are all independent with normal distribution $\mathscr{N}(\cdot; 0, \sigma^2)$, this system is a CMP with $\mathscr{S} = \mathbb{R}^4$, $\mathscr{A}(s) = \mathscr{A}$ for all $s \in \mathscr{S}$, and kernel

$$\begin{aligned} T_{\mathfrak{s}}(d\bar{s}\,|\,s,u) = \mathscr{N}(d\bar{s}^4; s^4_n + \ & \Delta a_2, \sigma^2)\delta(d\bar{s}^1; s^1_n + \Delta s^2_n) \\ & \times \delta(d\bar{s}^2; s^2_n + \Delta a_3)\delta(d\bar{s}^3; s^3_n + \Delta s^2_n), \end{aligned}$$

where $\delta(\cdot; a)$ is the Dirac delta measure centred at a and $\mathcal{N}(\cdot; m, \sigma^2)$ is the normal probability measure with mean m and variance σ^2.

5.3.2 Semantics of controlled Markov processes

The semantics of a CMP is characterised by its *paths* or executions, which reflect both the history of previous states of the system and of implemented control inputs. Paths are used to measure the performance of the system.

Definition 5.1 *A* finite path *of $\mathfrak{S}$ is a sequence $w_n = (s_0, u_0, \ldots, s_{n-1}, u_{n-1}, s_n)$, $n \in \mathbb{N}$, where $s_i \in \mathscr{S}$ are state coordinates and $u_i \in \mathscr{A}(s_i)$ are control input coordinates of the path. The space of all paths of length n is denoted by $\mathsf{PATH}_n := \mathscr{K}^n \times \mathscr{S}$. Further, we denote projections by $w_n[i] := s_i$ and $w_n(i) := u_i$. An* infinite path *of the CMP $\mathfrak{S}$ is the sequence $w = (s_0, u_0, s_1, u_1, \ldots)$, where $s_i \in \mathscr{S}$ and $u_i \in \mathscr{A}(s_i)$ for all $i \in \mathbb{N}$. As above, let us introduce $w[i] := s_i$ and $w(i) := u_i$. The space of all infinite paths is denoted by $\mathsf{PATH}_\infty := \mathscr{K}^\infty$.*

Given an infinite path w or a finite path w_n, we assume below that s_i and u_i are their state and control coordinates respectively, unless otherwise stated. For any infinite path $w \in \mathsf{PATH}_\infty$, its n-prefix (ending in a state) w_n is a finite path of length n, which we also call *n-history*. We are now ready to introduce the notion of control policy.

Definition 5.2 *A* policy *is a sequence $\rho = (\rho_0, \rho_1, \rho_2, \ldots)$ of universally measurable stochastic kernels ρ_n [14], each defined on the input space $\mathscr{A}$ given PATH_n and such that for all $w_n \in \mathsf{PATH}_n$ with $n \in \mathbb{N}$, $\rho_n(\mathscr{A}(s_n)|w_n) = 1$. The set of all policies is denoted by Π.*

Given a policy $\rho \in \Pi$ and a finite path $w_n \in \mathsf{PATH}_n$, the distribution of the next control input u_n is given by $\rho_n(\cdot|w_n)$ and is supported on $\mathscr{A}(s_n)$ (i.e., the chance of selecting an invalid input at s_n is zero). For a CMP $\mathfrak{S}$, any policy $\rho \in \Pi$ together with an initial probability measure $\alpha : \mathscr{B}(\mathscr{S}) \to [0,1]$ of the CMP induce a unique probability measure on the canonical sample space of paths [63] denoted by P_α^ρ with the expectation $\mathbb{E}_\alpha^\rho$. When the initial probability measure is supported on a single point, i.e., $\alpha(s) = 1$, we write P_s^ρ and $\mathbb{E}_s^\rho$ in place of P_α^ρ and $\mathbb{E}_\alpha^\rho$, respectively. We denote the set of probability measures on $(\mathscr{S}, \mathscr{B}(\mathscr{S}))$ by $\mathfrak{D}$. Implementation of a general policy requires an infinite memory. In this work, we restrict our attention to the class of policies that depend on the paths via a *finite memory*.

Definition 5.3 *A* finite-memory policy *for $\mathfrak{S}$ is a tuple $\rho_f := (\hat{\mathscr{S}}, \hat{s}_0, T_\mathfrak{p}, T_\mathfrak{o})$, where $\hat{\mathscr{S}}$ is the state space of the policy, $\hat{s}_0 \in \hat{\mathscr{S}}$ is the initial state, $T_\mathfrak{p} : \hat{\mathscr{S}} \times \mathscr{S} \times \mathscr{B}(\hat{\mathscr{S}}) \to [0,1]$ is the*

stochastic kernel for updating the state of the policy, and $T_{\mathfrak{o}} : \hat{\mathscr{S}} \times \mathscr{S} \times \mathscr{B}(\mathscr{A}) \to [0,1]$ *is the output kernel such that* $T_{\mathfrak{o}}(\mathscr{A}(s) \,|\, \hat{s}, s) = 1$ *for all* $\hat{s} \in \hat{\mathscr{S}}$ *and* $s \in \mathscr{S}$*. We denote the set of such policies by* $\Pi_f \subset \Pi$*.*

Note that the state space $\hat{\mathscr{S}}$ could in general be any continuous or hybrid space. The policy has access to the current state s_n of $\mathfrak{S}$ and updates its own state $\hat{s}_n$ according to $\hat{s}_{n+1} \sim T_{\mathfrak{p}}(\cdot \,|\, \hat{s}_n, s_n)$. As we will see later in Lemma 5.1, a finite $\hat{\mathscr{S}}$ is sufficient for optimal satisfaction of LTL specifications.

There is a special class of policies called *positional* that do not need a memory state as defined next.

Definition 5.4 *A policy* $\boldsymbol{\rho}$ *is* positional *if there is a stochastic kernel* $\mathscr{C} : \mathscr{S} \times \mathscr{B}(\mathscr{A}) \to [0,1]$ *such that at any time* $n \in \mathbb{N}$*, the input* u_n *is taken from the probability measure* $\mathscr{C}(\cdot|s_n)$*. Namely, the output kernel* $T_{\mathfrak{o}}(\cdot|\hat{s}, s)$ *in Definition 5.3 is independent of* $\hat{s}$*. We denote the class of positional policies by* $\Pi_p \subset \Pi_f$ *and a positional policy just by the kernel* $\mathscr{C} \in \Pi_p$*.*

Designing optimal finite-memory policies to satisfy a specification on $\mathfrak{S}$ can be reduced to finding an optimal positional policy for satisfying a specification on an extended model $\mathfrak{S}'$. This is formally proved in Section 5.4. Next we define the class of specifications used in this chapter.

5.3.3 Linear temporal logic

Linear temporal logic (LTL) provides a high-level language for describing the desired behaviour of a process. Formulas in this logic are constructed inductively by using a set of atomic propositions and combining them via Boolean operators. Consider a finite set of atomic propositions AP that defines the alphabet $\Sigma := 2^{\mathsf{AP}}$. Thus, each letter of this alphabet evaluates a subset of the atomic propositions as true. Composed as an infinite string, these letters form infinite words defined as $\omega = \omega_0, \omega_1, \omega_2, \ldots \in \Sigma^{\mathbb{N}}$. These words are connected to paths of CMP $\mathfrak{S}$ via a measurable labelling function $\mathsf{L} : \mathscr{S} \to \Sigma$ that assigns letters $\alpha = \mathsf{L}(s)$ to state $s \in \mathscr{S}$. That is, infinite paths $w = (s_0, u_0, s_1, u_1, \ldots)$ are mapped to the set of infinite words $\Sigma^{\mathbb{N}}$, as $\omega = \mathsf{L}(w) := (\mathsf{L}(s_0), \mathsf{L}(s_1), \mathsf{L}(s_2), \ldots)$.

Definition 5.5 *An LTL formula over a set of atomic propositions* AP *is constructed inductively as*

$$\psi ::= \mathit{true} \,|\, \mathit{false} \,|\, p \,|\, \neg p \,|\, \psi_1 \wedge \psi_2 \,|\, \psi_1 \vee \psi_2 \,|\, \mathsf{X}\,\psi \,|\, \psi_1 \,\mathsf{U}\, \psi_2 \,|\, \psi_1 \,\mathrm{sf}R\, \psi_2, \quad p \in \mathsf{AP}, \tag{5.3.2}$$

with ψ_1, ψ_2, ψ *being LTL formulas.*

Let $\omega_n = (\omega_n, \omega_{n+1}, \omega_{n+2}, \ldots)$ be a postfix of ω. The satisfaction relation is denoted by $\omega \vDash \psi$ (or equivalently $\omega_0 \vDash \psi$) and is defined recursively as follows

- $\omega_n \vDash \mathsf{true}$ always hold and $\omega_n \vDash \mathsf{false}$ does not hold.
- An atomic proposition, $\omega_n \vDash p$ for $p \in \mathsf{AP}$ holds if $p \in \omega_n$.
- A negation, $\omega_n \vDash \neg p$, holds if $\omega_n \nvDash p$.
- A logical conjunction, $\omega_n \vDash \psi_1 \wedge \psi_2$, holds if $\omega_n \vDash \psi_1$ and $\omega_n \vDash \psi_2$.
- A logical disjunction, $\omega_n \vDash \psi_1 \vee \psi_2$, holds if $\omega_n \vDash \psi_1$ or $\omega_n \vDash \psi_2$.
- A temporal next operator, $\omega_n \vDash \mathsf{X}\,\psi$, holds if $\omega_{n+1} \vDash \psi$.
- A temporal until operator, $\omega_n \vDash \psi_1 \,\mathsf{U}\, \psi_2$, holds if there exists an $i \in \mathbb{N}$ such that $\omega_{n+i} \vDash \psi_2$, and for all $j \in \mathbb{N}$, $0 \leq j < i$, we have $\omega_{n+j} \vDash \psi_1$.
- A temporal release operator is dual of the until operator and is defied as $\omega_n \vDash \psi_1 \,\mathrm{sf}R\, \psi_2$ if $\omega_n \nvDash \neg\psi_1 \,\mathsf{U}\, \neg\psi_2$.

In addition to the aforementioned operators, we can also use *eventually* $\Diamond$, and *always* $\Box$ operators as $\Diamond\psi := (\mathsf{true}\,\mathsf{U}\,\psi)$ and $\Box\psi := \mathsf{false}\,\mathrm{sf}R\,\psi$.

Remark 5.1 *The above definition is the canonical form of LTL and is called* positive normal form *(PNF), in which negations only occur adjacent to atomic propositions. If this is not the case, it is possible to construct an equivalent formula [8, Theorem 5.24] in the canonical form in polynomial time as a function of the length of the formula. We utilise the canonical form in Section 5.5.1 to construct a sequence of learning procedures that guides the optimal policy learning problem.*

Example 5.1 (Continued). The cart in Figure 5.1 should stay within the limits specified by C_1, always keep the pole upright in the range C_2, and reach the region A. We can express this requirement as the LTL specification

$$\psi = \Diamond a \wedge \Box(c_1 \wedge c_2) \tag{5.3.3}$$

with $\mathsf{AP} = \{a, c_1, c_2\}$ and the labelling function L with $a \in \mathsf{L}(s)$ if the cart is inside A, $c_1 \in \mathsf{L}(s)$ if the cart is inside C_1, and $c_2 \in \mathsf{L}(s)$ if the pole angle is inside the specified range of C_2.

5.3.4 Limit-deterministic Büchi automata

Satisfaction of LTL formulas can be checked on a class of automata called *Limit-Deterministic Büchi Automata* (LDBA) [28, 56, 137, 57]. Similar to [57], we use the translation of the specification to an LDBA that has one set of accepting transitions and is presented next. This translation is provided by [56]. An implementation of a wide range of algorithms for translating LTL to various types of automata is also available [82].

Definition 5.6 (LDBA) *an LDBA is a tuple $\mathscr{A} = (Q, \Sigma, \delta, q_0, \mathsf{Acc})$, where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ is a partial transition function, $q_0 \in Q$ is an initial state, and $\mathsf{Acc} \subset Q \times \Sigma \times Q$ is a set of accepting transitions. The transition function δ is such that it is total for all $(q, \omega) \in Q \times \Sigma$, i.e., $|\delta(q, \omega)| \leq 1$ for all $\omega \neq \varepsilon$ and $q \in Q$. Moreover, there is a partition $\{Q_N, Q_D\}$ for Q such that*

- *$\delta(q, \varepsilon) = \emptyset$ for all $q \in Q_D$, i.e., the ε-transitions can only occur in Q_N.*
- *$\delta(q, \omega) \subset Q_D$ for all $q \in Q_D$ and $\omega \in \Sigma$, i.e., the transitions starting in Q_D remain in Q_D.*
- *$\mathsf{Acc} \subset Q_D \times \Sigma \times Q_D$, the accepting transitions start only in Q_D.*

We can associate to an infinite word $\omega = (\omega_0, \omega_1, \omega_2, \ldots) \in (\Sigma \cup \{\varepsilon\})^{\mathbb{N}}$, a path $r = (q_0, \omega_0, q_1, \omega_1, q_2, \ldots)$ to $\mathscr{A}$ such that q_0 is the initial state of $\mathscr{A}$ and $q_{n+1} \in \delta(q_n, \omega_n)$ for all $n \in \mathbb{N}$. Such a path always exists when $\omega \in \Sigma^{\mathbb{N}}$. Let us denote by $inf(r)$ as the set of transitions (q, ω, q') appearing in r infinitely often. We say the word ω is accepted by $\mathscr{A}$ if it has a path r with $inf(r) \cap \mathsf{Acc} \neq \emptyset$. The *accepting language* of $\mathscr{A}$ is the set of words accepted by $\mathscr{A}$ and is denoted by $\mathscr{L}(\mathscr{A})$.

5.3.5 Problem statement

We are interested in the probability that an LTL specification ψ can be satisfied by paths of a CMP $\mathfrak{S}$ under different policies. Suppose a CMP $\mathfrak{S} = (\mathscr{S}, \mathscr{A}, \{\mathscr{A}(s) | s \in \mathscr{S}\}, T_{\mathfrak{s}})$, an LTL specification ψ over the alphabet Σ, and a labelling function $\mathsf{L} : \mathscr{S} \to \Sigma$ are given. An infinite path $w = (s_0, u_0, s_1, u_1, \ldots)$ of $\mathfrak{S}$ satisfies ψ if the infinite word $\omega = \mathsf{L}(w) \in \Sigma^{\mathbb{N}}$ satisfies ψ. We denote such an event by $\mathfrak{S} \models \psi$ and will study the probability of the event.

Remark 5.2 *In general, one should use the notation $\mathfrak{S} \models_{\mathsf{L}} \psi$ to emphasise the role of labelling function L in the satisfaction of ψ by paths of $\mathfrak{S}$. We eliminate the subscript L with the understanding that it is clear from the context. We add the labelling function in Section 5.5.1 when discussing multiple labelling functions for evaluation of $\mathfrak{S} \models \psi$.*

Given a policy $\rho \in \Pi_f$ and initial state $s \in \mathscr{S}$, we define the satisfaction probability as $f(s,\rho) := P_s^{\rho}(\mathfrak{S} \models \psi)$, and the supremum satisfaction probability $f^*(s) := \sup_{\rho\in\Pi_f} P_s^{\rho}(\mathfrak{S} \models \psi)$.

Problem 5.1 (Synthesis for LTL) *Given CMP $\mathfrak{S}$, LTL specification ψ, and labelling function L, find an optimal policy $\rho^* \in \Pi_f$ along with $f^*(s)$ s.t. $P_s^{\rho^*}(\mathfrak{S} \models \psi) = f^*(s)$.*

Measurability of the set $\{\mathfrak{S} \models \psi\}$ in the canonical sample space of paths under the probability measure P_s^{ρ} is proved in [165]. The function $f^*(s)$ is studied in [165] with an approximation procedure presented in [104]. These works are for fully known $\mathfrak{S}$ and only for *Büchi conditions* where the system should visit a set $B \subset \mathscr{S}$ infinitely often. This condition is denoted by $\psi = \square\lozenge B$.

Problem 5.2 (Synthesis for Büchi conditions) *Given $\mathfrak{S}$, a set of accepting states $B \in \mathscr{B}(\mathscr{S})$, find an optimal positional policy $\rho^* \in \Pi_p$ along with $f^*(s)$ s.t. $P_s^{\rho^*}(\mathfrak{S} \models \square\lozenge B) = f^*(s)$.*

Remark 5.3 *We have restricted our attention to finite-memory policies in Problem 5.1. This is due to the fact that proving existence of an optimal policy $\rho^* \in \Pi$ is an open problem. We note that existence of ε-optimal policies is already proved [101, 45]. We prove in Section 5.4 that Problems 5.1 and 5.2 are closely related: in order to find a solution for Problem 5.1, we can find a solution for Problems 5.2 on another CMP with an extended state space.*

5.4 Augmented CMP with reachability specification

In this section we discuss approximating solutions of Problems 5.1 and 5.2 using reachability specifications. This section contains one of the main contributions of the chapter that is formulating Assumption 5.1 and proving Theorems 5.1-5.3 and Lemma 5.1 for continuous-state CMPs.

5.4.1 The augmented CMP

Given $\mathfrak{S} = (\mathscr{S}, \mathscr{A}, \{\mathscr{A}(s)|s \in \mathscr{S}\}, T_{\mathfrak{s}})$ and a set of accepting states $B \subset \mathscr{S}$, we construct an augmented CMP $\mathfrak{S}_\zeta = \left(\mathscr{S}_\zeta, \mathscr{A}, \{\mathscr{A}_\zeta(s)|s \in \mathscr{S}_\zeta\}, T_{\mathfrak{s}}^\zeta\right)$ that has an additional dummy state ϕ, $\mathscr{S}_\zeta := \mathscr{S} \cup \{\phi\}$ and the same input space $\mathscr{A}$. The set of valid inputs $\mathscr{A}_\zeta(s)$ is the same as $\mathscr{A}(s)$ for all $s \in \mathscr{S}$ and $\mathscr{A}_\zeta(\phi) = \mathscr{A}$. The stochastic kernel of $\mathfrak{S}_\zeta$ is a modified version of $T_{\mathfrak{s}}$ as $T_{\mathfrak{s}}^\zeta(A|s,u) = [1-(1-\zeta)\mathbf{1}_B(s)]T_{\mathfrak{s}}(A|s,u)$, $T_{\mathfrak{s}}^\zeta(\phi|s,u) = (1-\zeta)\mathbf{1}_B(s)$, and $T_{\mathfrak{s}}(\phi|\phi,u) = 1$, for all $A \in \mathscr{B}(\mathscr{S})$, $s \in \mathscr{S}$ and $u \in \mathscr{A}_\zeta(s)$. In words, $T_{\mathfrak{s}}^\zeta$ takes the same $T_{\mathfrak{s}}$, adds a sink state ϕ, and for any accepting state $s \in B$, the process will jump to ϕ with probability $(1-\zeta)$. It

also normalises the outgoing transition probabilities of accepting ones with ζ. We establish a relation between $\mathfrak{S}$ and $\mathfrak{S}_\zeta$ regarding satisfaction of Büchi conditions under the following assumption.

Assumption 5.1 *For $\mathfrak{S}$ and a set B, define the random variable τ_B as the number of times the set B is visited in paths of $\mathfrak{S}$ conditioned on having it as a finite number. The quantity $\tau_B^* := \sup_\rho \mathbb{E}_s^\rho(\tau_B)$ is bounded for any $s \in \mathscr{S}$.*

Theorem 5.1 *Given $\mathfrak{S}$ satisfying Assumption 5.1 and for any positional policy ρ on $\mathfrak{S}$, there is a positional policy $\bar{\rho}$ on $\mathfrak{S}_\zeta$ such that*

$$P_s^{\bar{\rho}}(\mathfrak{S}_\zeta \models \Diamond\phi) - (1-\zeta)\mathbb{E}_s^\rho(\tau_B) \leq P_s^\rho(\mathfrak{S} \models \Box\Diamond B) \leq P_s^{\bar{\rho}}(\mathfrak{S}_\zeta \models \Diamond\phi). \tag{5.4.1}$$

For any $\bar{\rho}$ on $\mathfrak{S}_\zeta$, there is ρ on $\mathfrak{S}$ such that the same inequality holds.

The above theorem shows that the probability of satisfying a Büchi condition with accepting set $B \subset \mathscr{S}$ by $\mathfrak{S}$ is upper bounded by the probability of reaching ϕ in $\mathfrak{S}_\zeta$. It also establishes a lower bound but requires knowing $\mathbb{E}_s^\rho(\tau_B)$.

Inequalities of Theorem 5.1 can be extended to optimal satisfaction probabilities as stated in the next theorem.

Theorem 5.2 *For any $\mathfrak{S}$ satisfying Assumption 5.1, we have*

$$\sup_{\bar{\rho}} P_s^{\bar{\rho}}(\mathfrak{S}_\zeta \models \Diamond\phi) - (1-\zeta)\tau_B^* \leq \sup_\rho P_s^\rho(\mathfrak{S} \models \Box\Diamond B) \leq \sup_{\bar{\rho}} P_s^{\bar{\rho}}(\mathfrak{S}_\zeta \models \Diamond\phi). \tag{5.4.2}$$

Corollary 5.1 *Under Assumption 5.1, the optimal value $\sup_{\bar{\rho}} P_s^{\bar{\rho}}(\mathfrak{S}_\zeta \models \Diamond\phi)$ converges to $\sup_\rho P_s^\rho(\mathfrak{S} \models \Box\Diamond B)$ from above when ζ converges to one from below, and the rate of convergence is at least linear with $(1-\zeta)$.*

Next example highlights the need for Assumption 5.1 on $\mathfrak{S}$ to get the linear convergence. Such an assumption holds for all $\mathfrak{S}$ with finite state spaces as used in [57, 19] but it may not hold for $\mathfrak{S}$ with infinite state spaces.

Example 5.2 *Consider the $\mathfrak{S}$ presented in Figure 5.2, which has a countable state space $\{1,2,3,\ldots\}$ and the input space is singleton. $\mathfrak{S}$ starts at state $s=2$. The state 1 is absorbing. From any other state n, it jumps to state 1 with probability $\frac{1}{n}$ and to state $(n+1)$ with probability $\frac{n-1}{n}$. Take the set of accepting states $B = \{3,4,5,\ldots\}$. $\mathbb{E}_s^\rho(\tau_B)$ is unbounded for $\mathfrak{S}$:*

$$\mathbb{E}_s^\rho(\tau_B) = \sum_{n=1}^{\infty} n \times \frac{1}{2} \times \frac{2}{3} \times \frac{3}{4} \times \cdots \frac{n}{n+1} \times \frac{1}{n+2} = \sum_{n=1}^{\infty} \frac{n}{(n+1)(n+2)} = \infty.$$

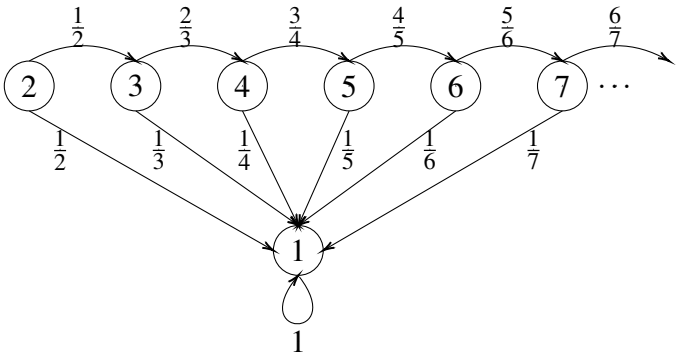


Fig. 5.2 A CMP with space $\{1,2,3,\ldots\}$, a single input and accepting states $B = \{3,4,5,\ldots\}$. Its augmented CMP $\mathfrak{S}_\zeta$ does not show convergence with a linear rate [80].

It can be easily verified that

$$P_s^{\rho}(\mathfrak{S} \models \Box\Diamond B) = \frac{1}{2}\times\frac{2}{3}\times\frac{3}{4}\times\frac{4}{5}\times\cdots = 0$$
$$P_s^{\bar{\rho}}(\mathfrak{S}_\zeta \models \Diamond\phi) = (1-\zeta)\left[1+\frac{1}{2}\zeta+\frac{1}{3}\zeta^2+\frac{1}{4}\zeta^3+\ldots\right] = \frac{-(1-\zeta)\ln(1-\zeta)}{\zeta}.$$

The left-hand side of inequality (5.4.2) *is still technically true for this* $\mathfrak{S}$ *despite* $\mathbb{E}_s^{\rho}(\tau_B) = \infty$*, but the provided lower bound is trivial and does not give linear convergence mentioned in Corollary 5.1.*

Remark 5.4 *The lower bound in* (5.4.2) *is useful for showing linear convergence when* $\zeta \to 1^-$*, but it is not beneficial for learning purposes since the computation of* τ_B^* *requires knowing the structure of the underlying unknown transition kernel* $T_{\mathfrak{s}}$*. In the next subsection, we utilise Theorem 5.2 to give a lower bound independent of* τ_B^**. We also demonstrate convergence experimentally in the case study section.*

5.4.2 The product CMP

The product of a CMP and an LDBA is used in the literature, e.g., [57, 19, 61] for finite state spaces. We provide this construction for continuous-state CMPs.

Definition 5.7 *The product CMP* $\mathfrak{S}^{\otimes} = (\mathscr{S}^{\otimes}, \mathscr{A}^{\otimes}, \{\mathscr{A}^{\otimes}(x) | x \in \mathscr{S}^{\otimes}\}, \mathscr{T}_x^{\otimes})$ *of an CMP* $(\mathscr{S}, \mathscr{A}, \{\mathscr{A}(s) | s \in \mathscr{S}\}, \mathscr{T}_s)$ *and an LDBA* $\mathscr{A} = (Q, \Sigma, \delta, q_0, \mathsf{Acc})$ *is defined as follows:* $\mathscr{S}^{\otimes} := S \times Q$ *is the set of states,* $\mathscr{A}^{\otimes} := \mathscr{A} \cup A^{\varepsilon}$ *with* $A^{\varepsilon} := \{\varepsilon_q | q \in Q\}$ *is the set of actions. The valid input sets are* $\mathscr{A}^{\otimes}(s,q) = \mathscr{A}(s)$ *if* $\delta(q,\varepsilon) = \emptyset$ *and* $\mathscr{A}^{\otimes}(s,q) = \varepsilon_{q'}$ *if* $q' \in \delta(q,\varepsilon)$*.*

The stochastic kernel is defined as

$$\mathscr{T}_x^{\otimes}(A\times\{q'\}|s,q,u) := \begin{cases} T_{\mathfrak{s}}(A|s,u) & \text{if } q' = \delta(q,\mathsf{L}(s)) \text{ and } u \in \mathscr{A}(s) \\ \mathbf{1}_A(s) & \text{if } q' = \delta(q,\varepsilon) \text{ and } u = \varepsilon_{q'} \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathbf{1}_A(s)$ is the indicator function of the set A.

Any distribution $\alpha : \mathscr{B}(\mathscr{S}) \to [0,1]$ for the initial state of $\mathfrak{S}$ induces an initial distribution $\alpha^{\otimes} : \mathscr{B}(\hat{\mathscr{S}}) \to [0,1]$ with $\alpha^{\otimes}(A\times\{q\}) := \alpha(A)$ for any $A \in \mathscr{B}(\mathscr{S})$ and $q = q_0$, and zero otherwise. The set of accepting states in the product CMP $\mathfrak{S}^{\otimes}$ is

$$\mathsf{Acc}^{\otimes} = \left\{(s,q)\,|\,(q,\mathsf{L}(s),q') \in \mathsf{Acc}, q' = \delta(q,\mathsf{L}(s))\right\}. \tag{5.4.3}$$

We say the path $w^{\otimes}$ of $\mathfrak{S}^{\otimes}$ satisfies the Büchi condition ψ_B if the number of states in $\mathsf{Acc}^{\otimes}$ visited by the path is not finite (the set is visited infinitely often).

Lemma 5.1 *Any positional policy on $\mathfrak{S}^{\otimes}$ can be translated into a finite-memory policy for $\mathfrak{S}$ that has a finite state space equal to the space of the LDBA $\mathscr{A}$. Moreover, the class of finite-memory policies are sufficient for solving Problem 5.1 if an optimal policy exists.*

Due to Lemma 5.1, we focus in the next section on finding positional policies for the product CMP using reinforcement learning. Next theorem is one of the main contributions of the chapter that formalises a lower bound on the optimal satisfaction probability.

Theorem 5.3 *For any $\mathfrak{S}$, specification ψ, labelling function L, and any $s \in \mathscr{S}$,*

$$1 - \inf_{\bar{\rho}} P^{\bar{\rho}}_{s,q_0}(\mathfrak{S}^{\otimes}_{1\zeta} \models \Diamond\phi) \le \sup_{\rho} P^{\rho}_s(\mathfrak{S} \models \psi) \le \sup_{\bar{\rho}} P^{\bar{\rho}}_{s,q_0}(\mathfrak{S}^{\otimes}_{2\zeta} \models \Diamond\phi), \tag{5.4.4}$$

where $\mathfrak{S}^{\otimes}_{1\zeta}$ and $\mathfrak{S}^{\otimes}_{2\zeta}$ are the augmented CMPs constructed for the products of $\mathfrak{S}$ with $\mathscr{A}_{\neg\psi}$ and $\mathscr{A}_{\psi}$, respectively.

In the next section, we focus on the computation of the right-hand side of (5.4.4) using RL. The left-hand side is computed similarly.

5.5 Reinforcement learning for policy synthesis

This section contains another main contribution of the chapter that is using relaxed versions of the LTL specification in learning a policy. We have shown that Problem 5.1 can be reduced to

Problem 5.2 on a product CMP, which then can be approximated using reachability objectives as shown in (5.4.4). The reachability probability is an average reward criterion

$$P_s^{\bar{\rho}}(\mathfrak{S}_\zeta \models \Diamond\phi) = \lim_{N\to\infty} \frac{1}{N+1} \mathbb{E}_s^{\bar{\rho}} \sum_{n=0}^{N} R(s_n), \tag{5.5.1}$$

with the reward function $R : \mathscr{S}_\zeta \to \mathbb{R}$ defined as $R(s) = 1$ for $s = \phi$ and $R(s) = 0$ otherwise. It can alternatively be written with a total (undiscounted) additive reward criterion by assigning reward one to the first visit of the ϕ and zero otherwise. Both cases can be computed by RL algorithms whenever the model of the CMP is not known or is hard to analyse. Any off-the-shelf RL algorithm for continuous systems can be used to learn a policy. Note that for a general LTL specification, the reward function R is state dependent on the product CMP, but it becomes path dependent when interpreted over the original CMP through the LDBA of the specification.

Advantage actor-critic RL. RL algorithms are either value based or policy based. In value-based RL, the algorithm tries to maximise a value function that is a mapping between a state-input pair and a value. Policy-based RL tries to find the optimal policy without using a value function. The policy-based RL has better convergence and effectiveness on high dimensions or continuous state spaces, while value-based RL is more sample efficient and steady. The intersection between these two categories is the actor-critic RL, where the goal is to optimise the policy and the value function together. It optimises the policy and value function as a function of state. We use in this chapter the *Advantage Actor-Critic RL* (A2C) [111] that takes the value function as a baseline. It makes the cumulative reward smaller by subtracting it with the baseline, thus have smaller gradients and more stable updates. It works better in comparison with other actor-critic RL in terms of the stability of the learning process and lower variance. An implementation of A2C is available in MATLAB. We have taken this implementation and adapted it to be applicable to the augmented CMP $\mathfrak{S}_\zeta^\otimes$. A pseudo algorithm of our approach based on the A2C is provided in the extended version [80].

5.5.1 Specification-guided learning

The reward function R used in (5.5.1) is sparse and it slows down the learning. To improve the learning performance, we give an algorithm that sequentially trains the Actor and Critic networks and guides the learning process by a sequence of labelling functions defining satisfaction of the specification with different relaxation degrees. This sequential training has a similar spirit as the approach of [92]. The novelty of our algorithm is in constructing a

sequence of labelling functions that automatically encode the satisfaction relaxation, thus requires Actor and Critic networks with fixed structures.

Relaxed labelling functions. We denote the elements of the alphabet by $\Sigma = \{\Sigma_1, \ldots, \Sigma_m\}$. The labelling function $\mathsf{L} : \mathscr{S} \to \Sigma$ induces a partition of the state space $\{S_1, S_2, \ldots, S_m\}$ such that $S_i := \mathsf{L}^{-1}(\Sigma_i)$, $\mathscr{S} = \cup_{i=1}^{n} S_i$, and $S_i \cap S_j = \emptyset$ for all $i \neq j$. Define the *r-expanded* version of a set $S \subset \mathscr{S}$ by

$$S^{+r} := \{s \in \mathscr{S} \,|\, \exists s' \in S \text{ with } \|s - s'\|_\infty \leq r\}, \tag{5.5.2}$$

for any $r \geq 0$, where $\|\cdot\|_\infty$ is the infinity norm. Define the *r-relaxed* labelling function $\mathsf{L}_r : \mathscr{S} \to 2^\Sigma$ with

$$\mathsf{L}_r(s) := \{\Sigma_i \,|\, \mathsf{L}(S_i) = \Sigma_i \text{ and } s \in S_i^{+r}\}, \quad \text{for all } s \in \mathscr{S}. \tag{5.5.3}$$

Theorem 5.4 *The relaxed labelling functions L_r are monotonic with respect to r, i.e., for any $0 \leq r \leq r'$ and L, we have $\{\mathsf{L}(s)\} = L_0(s) \subset L_r(s) \subset L_{r'}(s)$.*

Specification interpreted over Σ. We interpret the specification ψ over the letters in Σ instead of the atomic propositions in AP. For this, we take the PNF form of ψ and replace an atomic proposition p by $\vee_i\{\Sigma_i \,|\, p \in \Sigma_i\}$. We also replace $\neg p$ by $\vee_i\{\Sigma_i \,|\, p \notin \Sigma_i\}$. Let us denote this specification in PNF with the letters $\{\Sigma_1, \ldots, \Sigma_m\}$ treated as atomic propositions $\bar{\psi}$. We can construct its associated LDBA $\bar{\mathscr{A}_\psi}$ as discussed in Section 5.3.4,

$$\bar{\mathscr{A}_\psi} := (\bar{Q}, 2^\Sigma, \bar{\delta}, \bar{q}_0, \overline{\mathsf{Acc}}). \tag{5.5.4}$$

Theorem 5.5 *For any $0 \leq r \leq r'$ and L, we have*

$$\{\mathfrak{S} \models_{\mathsf{L}} \psi\} = \left\{\mathfrak{S} \models_{L_0} \bar{\psi}\right\} \subset \{\mathfrak{S} \models_{\mathsf{L}_r} \bar{\psi}\} \subset \left\{\mathfrak{S} \models_{L_{r'}} \bar{\psi}\right\}, \tag{5.5.5}$$

where L_r is the r-relaxed labelling function defined in (5.5.3)*, and $\bar{\psi}$ is the specification ψ in PNF and interpreted over Σ.*

A pseudo algorithm for the specification-guided learning is provided in Alg. 1 that is based on repeatedly applying an RL algorithm to $\mathfrak{S}$ using a sequence of r-relaxed labelling functions. The algorithm starts by applying Actor-Critic RL to the most relaxed labelling function $\mathsf{L}_{r_\mathsf{m}}$. Then it repeatedly fixes the actor network (the policy) by setting its learning rate to zero (Step 6), runs Actor-Critic RL on the next most relaxed labelling function to update the Critic network that gives the total reward (Step 7), and uses these two networks as initialisation for running Actor-Critic RL to optimise both Actor and Critic networks (Step 9).

Algorithm 1: Specification-guided learning [80].

input : CMP $\mathfrak{S}$ as a black box, specification ψ, labelling function $\mathsf{L} : \mathscr{S} \to \Sigma$
output : Actor network $\mu(s,q|\theta^{\mu})$ and Critic network $\mathscr{Q}(s,q|\theta^{\mathscr{Q}})$

1 Select **hyper-parameters** $r_{\mathsf{m}} > r_{\mathsf{m}-1} > \ldots r_1 > r_0 = 0$
2 Compute *r-relaxed* labelling functions $\mathsf{L}_{r_i} : \mathscr{S} \to 2^{\Sigma}$ according to (5.5.3)
3 Compute LDBA $\bar{\mathscr{A}_{\psi}}$ as discussed for (5.5.4)
4 Run the Actor-Critic RL with ($\mathfrak{S}$, $\bar{\mathscr{A}_{\psi}}$, $\mathsf{L}_{r_{\mathsf{m}}}$) to get Actor and Critic networks $\mu(s,q|\theta^{\mu})$ and $\mathscr{Q}(s,q|\theta^{\mathscr{Q}})$
5 **for** $i = \mathsf{m}$ ***to*** 1 **do**
6 Fix parameters θ^{μ} of the Actor network by setting its learning rate to zero
7 Run Actor-Critic RL with $\mathsf{L}_{r_{i-1}}$ to train only the Critic network
8 Change the learning rate of Actor back to normal
9 Run Actor-Critic RL with $\mathsf{L}_{r_{i-1}}$ and initial parameters obtained in Steps 6 and 7
10 **end**

Remark 5.5 *The main feature of Alg. 1 is that the structure of the LDBA $\bar{\mathscr{A}_{\psi}}$ is fixed through the entire algorithm and only the labelling function (thus the reward function) is changed in each iteration.*

We presented Alg. 1 for the computation of the right-hand side of (5.4.4). The lower bound in (5.4.4) is computed similarly. The only difference is that the LDBA is constructed using $\neg\psi$. The reward function should assign zero to ϕ and one to all other states. The *r*-relaxed labelling functions in (5.5.3) can be used for guiding the computation of the lower bound.

5.6 Case studies

To demonstrate our model-free policy synthesis method, we first apply it to the cart-pole system of Example 5.1 and then discuss the results on a 6-dim boat driving problem. Note that it is not possible to compare our approach with [89] that only handles finite-horizon specifications. Also, the approach of [60, 62] maximises the frequency of visiting a sequence of sets of accepting transitions and does not come with formal convergence or lower-bound guarantees.

Our algorithms are implemented in MATLAB R2019a on a 64-bit machine with an Intel Core(TM) i7 CPU at 3.2 GHz and 16 GB RAM.

5.6.1 Cart-pole system

We use negation of the specification (5.3.3) to learn a lower bound on the optimal satisfaction probability. We set the safe interval $C_2 = [-12^{\circ}, 12^{\circ}]$ for the angle, safe range $C_1 = [-1, 1]$

and reach set $A = [0.4, 1]$ in meters for the location. We first directly apply A2C RL to the specification (5.3.3) and set the timeout of 24 hours. The RL does not converge to a policy within this time frame. Note that it is a very challenging task to keep the pole upright and at the same time move the cart to reach the desired location.

We then apply Alg. 1 by using the expanded sets $A^{+i} = [\alpha_i, 1]$ with $\alpha_i \in \{-1, 0.01, 0.4\}$ for defining the relaxed labelling functions L_i. We select the Actor network to have 7 inputs (4 real states and 3 discrete states of the automaton) and 2 outputs. It also has two fully-connected hidden layers each with 7 nodes. The Critic network has the same number of inputs as Actor network, one output, and one fully-connected layer with 7 nodes. We also set $\zeta = 0.999$, learning rate 8×10^{-4}, and episode horizon $N = 500$.

Our sequential learning procedure successfully learns the policy within 44 minutes and gives the lower bound 0.9526 for satisfaction probability (according to Theorem 5.3). Figure 5.3 shows cart's position (left) and pole's angle (right) for 50,000 trajectories under the learned policy. The grey area is an envelop for these trajectories, their mean is indicated by the solid line and the standard deviation around mean is indicated by dashed lines. Only 515 trajectories (1.03%) go outside of the safe location $[-1, 1]$ or drop the pole outside of the angle interval $[-12^\circ, 12^\circ]$. All trajectories reach the location $[0.4, 1]$. The histogram of the first time the trajectories reach this interval is presented in Figure 5.4, which shows majority of the trajectories reach this interval within 150 time steps.

Using Hoeffding's inequality,[1] we get that the true satisfaction probability under the learned policy is in the interval $[0.975, 1]$ with confidence $1 - 4 \times 10^{-10}$. This is in line with the lower bound 0.9526 computed by the RL.

5.6.2 Boat driving problem

The objective in the boat driving problem is to design a policy for driving a boat from the left bank to the right bank quay in a river with strong nonlinear current. Variations of this problem have been used in the literature (see e.g. [121]). We use a more general version presented in [74] with the dynamics reported in [80]. The model has six continuous states including x and y coordinates for the location both in the interval $[0, 200]$. The boat starts its journey from the left bank of the river $x_0 = 0$ and $y_0 \in [60, 100]$ and should reach the right bank of the river $x_n = 200$ and $y_n \in [95, 105]$ for some n. There is an unknown nonlinear stochastic current affecting the location of the boat.

[1] Hoeffding's inequality asserts that the tail of the binomial distribution is exponentially decaying: $Prob(H \geq (p+\varepsilon)N) \leq exp(-2\varepsilon^2 N)$ for all $\varepsilon > 0$ with the number of trials N, the success probability p, and the observed number of successes H.

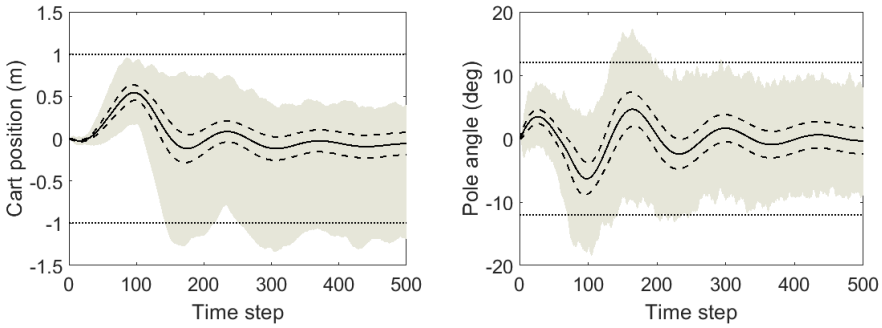


Fig. 5.3 **Cart-pole system.** Cart's position (left) and pole's angle (right) for $50,000$ trajectories under the learned policy. The grey area is an envelop for these trajectories, their mean is indicated by the solid line and the standard deviation around mean is indicated by dashed lines. Only 515 trajectories (1.03%) go outside of the safe location $[-1,1]$ or drop the pole outside of the angle interval $[-12^\circ, 12^\circ]$ [80].

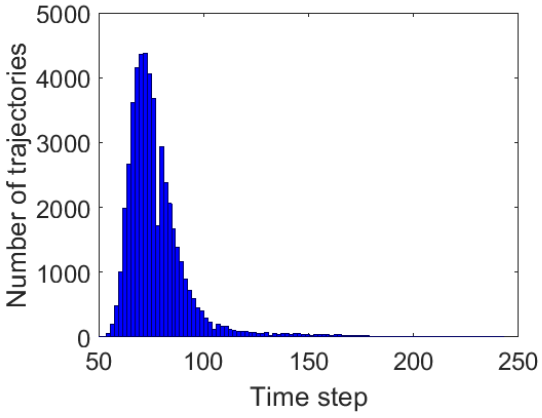


Fig. 5.4 **Cart-pole system.** Histogram of the first time the trajectories reach the interval $[0.4,1]$. A majority of the trajectories reach this interval within 150 time steps [80].

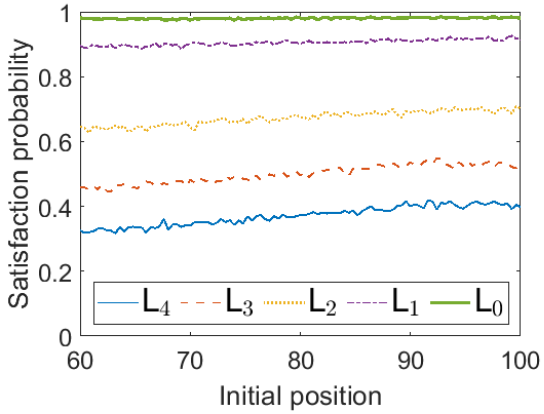


Fig. 5.5 **Boat driving problem.** The satisfaction probability as a function of the initial position y_0 for the policies learned with labelling functions L_i, $i \in \{0,1,2,3,4\}$ [80].

Direct application of A2C RL does not converge to a policy within 24 hours. We then apply Alg. 1 with labelling functions $\mathsf{L}_4, \mathsf{L}_3, \mathsf{L}_2, \mathsf{L}_1, \mathsf{L}_0$ respectively with the target range $[50, 150]$, $[80, 120]$, $[85, 115]$, $[90, 110]$, and $[95, 105]$. We also adaptively increase the value of ζ to get better lower bounds on the satisfaction probability: $\zeta_4 = 0.9950$, $\zeta_3 = 0.9965$, $\zeta_2 = 0.9980$, $\zeta_1 = 0.9995$, and $\zeta_0 = 0.9999$. The results of this sequential learning procedure are presented in Fig. 5.5 as a function of the initial position of the boat. The learning rate is set to 8×10^{-4} and the computational time is 70 minutes. The results show that the lower bound on satisfaction probability is monotonically increasing for all initial positions of the boat when ζ increases, which shows also convergence as a function of ζ.

In order to validate the computed bound, we took the initial position $(x_0, y_0) = (0, 80)$ and obtained 50,000 trajectories. All trajectories reach the target location. Based on Hoeffding's inequality, the true probability is in $[0.99, 1]$ with confidence 5×10^{-5}, which confirms the lower bound 0.9810 computed by RL.

5.7 Future work

We presented an approach for applying reinforcement learning (RL) to unknown continuous-state stochastic systems with the goal of satisfying a linear temporal logic specification. We formulated an optimal average reward criterion that converges linearly to the true optimal satisfaction probability under suitable assumptions. We used RL to learn a lower bound on this optimal value and improved the performance of the learning by a sequential algorithm using relaxed versions of the specification. In future, we plan to study the relation with discounting reward functions [19], formal connections with maximising frequency of visits, and providing guidance in adapting the network architecture in the RL to the structure of the specification.

Chapter 6

Translating omega-regular specifications to average objectives for model-free reinforcement learning

6.1 Chapter introduction

In all previous chapters, we learn an omega-regular task in an episodic manner whereby the environment is periodically reset to an initial state during learning. In some settings, this assumption is challenging or impossible to satisfy. Instead, in the continuing setting, the agent explores the environment without resets over a lifetime. This is a more natural setting for reasoning about omega-regular specifications defined over infinite traces of agent behaviour. Optimising the average reward instead of the usual discounted reward is more natural in this case due to the infinite-horizon objective that poses challenges to the convergence of discounted RL solutions.

We restrict our attention to the omega-regular languages which correspond to *absolute liveness* specifications. Any finite prefix of agent behaviour cannot invalidate these specifications in accordance with the spirit of a continuing problem. We propose a translation from absolute liveness omega-regular languages to an average reward objective for RL. Our reduction can be done on-the-fly, without full knowledge of the environment, thereby enabling the use of model-free RL algorithms. Additionally, we propose a reward structure that enables RL without episodic resetting in communicating MDPs, unlike previous approaches. We demonstrate empirically with various benchmarks that our proposed method of using average reward RL for continuing tasks defined by omega-regular specifications is more effective than competing approaches that leverage discounted RL.

The research presented in the chapter has been presented at the 21st International Conference on Autonomous Agents and Multiagent Systems [79]. An extended version of this research is invited for potential publication in the Journal of Autonomous Agents and Multiagent Systems. This research was the result of a collaboration with the University of Colorado Boulder. My role in this research is to provide the theoretical results and write the paper.

6.2 Introduction

The area of reinforcement learning (RL) for sequential decision-making has witnessed tremendous success in recent years. This is evidenced by RL architectures with superhuman performance in games of perception and precision such as Go [138, 140], general board games [139], and Atari [113, 112, 135], among others. In these settings, the reward signal by which agent experience is labelled for positive or negative reinforcement needs only account for the current state observed by the agent and the action chosen by the same. However, it is often necessary or useful to account for the history of the agent when arbitrating the credit assignment computed by the reward function of the underlying decision process. Examples of this include learning in decision processes where rewards are sparse [116], where states are partially observable [166], or where the objective is temporally extended [24]. Moreover, it is often more natural to express the goal of the agent as the language of desirable and undesirable outcomes, with the reward signal reflecting the pursuit and avoidance, respectively, of such behaviours. The use of formal language structures to define such behavioural specifications has been well-studied in the area of formal verification and is gaining traction in specifying reward signals for RL. These specifications take the form of automata with various accepting conditions that define the language they capture. It is worth noting that there exist techniques to translate natural language objectives to their corresponding automata representations in some settings [21].

The recent development of reward machines provides a similar structured representation of the underlying reward signal and can capture non-Markovian, or history-dependent, behaviour [65, 66]. These reward machines are automata whose transitions denote the reward observed by an agent for traversing from the initial node in the reward machine to some other node via a sequence of transitions that capture semantically meaningful events in the decision process. This naturally enables the definition of temporally extended objectives in RL as well as the augmentation of the underlying decision process to include observed transitions in the reward machine, thereby transforming some non-Markovian objectives into

Markovian tasks over the augmented decision process [47, 178]. Traditional off-the-shelf RL solutions can be employed for these Markovian tasks.

The field of RL [160] studies sampling-based approaches to derive decision-making policies that rely on scalar reward signals to optimise for the underlying learning objective. Samples of behaviour and their associated rewards are used in a data-driven fashion to refine state or action value functions and compute policies that maximise expected cumulative reward. In *episodic* RL, the environment is periodically reset to an initial state over the course of learning. In *continuing* RL, the environment is not reset, and the agent seeks to maximise its performance over its lifetime. Additionally, the environment in this setting should permit the agent to visit any state from all other states in order to allow the agent to correct early mistakes. Such an environment is called *communicating*.

The foregoing notions of formal languages and RL have been used to great effect in the formal synthesis of control policies, which has garnered much interest in recent years [12, 71, 103]. This paradigm enables developers to focus on defining the behavioural specification of interest in some formal language as opposed to translating and implementing said specification as a reward signal or learning objective manually, which is known to be error-prone and lacking guarantees of behaviour [81]. Formal synthesis algorithms leverage the underlying specification and compute a correct-by-construction policy yielding the desired behaviour. In this chapter, we explore such formal synthesis of policies through the use of average reward model-free reinforcement learning (RL) [100, 169] for a class of formal specifications expressed in omega-regular languages [8]. These languages provide a rich formalism to express desired properties of the system unambiguously. These languages are accepted by automata on infinite words, where a word denotes a sequence of semantically meaningful observations observed by the agent. We introduce the notion of nondeterministic reward machines to capture reward inherent in ω-regular automata. Then, by computing a product Markov decision process (MDP) between the reward machine of an ω-regular specification and the MDP that models the agent-environment dynamics, existing continuing RL algorithms can be readily adopted to search for an optimal policy.

We focus our attention to the problem of translating omega-regular objectives to average reward for model-free RL. This is justified by challenges facing the adoption of discounted RL for continuing tasks, as discussed in the sequel. Consider the cumulative reward that is often expressed as a discounted sum of the individual rewards received by the agent at each step. The use of a discount factor ensures that the cumulative reward is bounded even for an infinite sequence of actions and rewards, thereby facilitating convergence. While mathematically convenient, discounting results in short-term rewards being valued higher than the long-run performance of the system. Thus, obtaining a suitable policy for long-run

behaviour depends on choosing the right discount factor, which may have to approach 1 as the size of the environment increases. However, choosing a discount factor close to 1 results in a weak contraction in RL algorithms, causing slow convergence and instability. This is exacerbated in continuing task settings, where one has to choose a very high discount factor to approximate the maximisation of long-run performance. Moreover, despite the success of discounted RL for episodic tasks [112], the solution of discounted RL depends on initial state distributions, which makes it an optimization that is not compatible with function approximations in continuing settings [115]. Such function approximation is critical for learning policies on large-scale models, as evidenced by the adoption of large learning models in state-of-the-art RL solutions. Thus, a natural alternative to discounting is optimising the agent's average reward in these settings.

However, the adoption of average reward RL faces its own set of challenges. While establishing the existence of an optimal policy for discounted RL is relatively straightforward, analysing MDPs with the average reward objective is more difficult and requires some assumptions over the structure of the underlying MDP. Unlike discounted RL approaches, where the discount factor plays the role of the contraction parameter and enables convergence, in average reward RL algorithms, the contraction factor depends on communicating assumptions of the MDP. When the communicating assumption is satisfied, there are model-free convergent average reward RL algorithms. Satisfying the communicating assumption presents a challenge to the adoption of average reward RL for the formal synthesis of policies satisfying omega-regular specifications. Indeed, the product MDP resulting from the property and the underlying MDP may not be communicating. When episodic resetting is unavailable, communication is a natural assumption. The challenge is to ensure that this property is preserved in the MDP product. By leveraging the proposed reward machines, we demonstrate that this communicating property is preserved in the product MDP for an important class of omega-regular specifications.

The main contribution of this chapter is to provide an average-reward model-free RL algorithm for the design of policies that satisfy a given *absolute liveness* omega-regular specification. Our approach ensures that the communicating property is preserved in the product, enabling the learning of optimal policies, while not requiring episodic resetting. Despite the assumption of communicating MDPs, the naive synchronisation of the MDP with the automaton is not generally communicating. We propose a reward machine and an augmented specification to preserve the communicating property of the synchronised MDP. Our work is the first to provide a translation from omega-regular objectives to average-reward RL with formal guarantees. We validate our approach with an implementation of the proposed construction and demonstrate its effectiveness on several benchmarks.

The chapter is organised as follows. Section 6.3 includes the preliminaries and states the problem definition. Section 6.4 presents the main results of the chapter, which establish a novel algorithm for producing optimal policies for an absolute liveness property with average reward RL. In Section 6.5, we test the performance of our approach on different case studies against prior techniques. Section 6.6 discusses related work in formal synthesis, average reward RL, and related areas. We conclude with a summary in Section 6.7.

6.3 Problem definition

Markov Decision Processes. Let $\mathscr{D}(S)$ be the set of distributions over a given set S. A Markov decision process (MDP) $\mathscr{M}$ is a tuple (S, s_0, A, T, AP, L) where S is a finite set of states, $s_0 \in S$ is the initial state, A is a finite set of *actions*, $T\colon S \times A \rightharpoonup \mathscr{D}(S)$ is the probabilistic transition function, AP is the set of *atomic propositions*, and $L\colon S \to 2^{AP}$ is the *labeling function*.

For any state $s \in S$, we let $A(s)$ denote the set of actions that can be selected in state s. An MDP is a Markov chain if $A(s)$ is singleton for all $s \in S$. For states $s, s' \in S$ and $a \in A(s)$, $T(s,a)(s')$ equals $p(s'|s,a)$, probability of next state, s' given current state s and taking the action a. A *run* of $\mathscr{M}$ is an ω-word $\langle s_0, a_1, s_1, \ldots\rangle \in S \times (A \times S)^\omega$ such that $p(s_{i+1}|s_i, a_{i+1})>0$ for all $i \geq 0$. A finite run is a finite such sequence. For a *run* $r = \langle s_0, a_1, s_1, \ldots\rangle$ we define the corresponding labelled run as $L(r) = \langle L(s_0), L(s_1), \ldots\rangle \in (2^{AP})^\omega$. We write $Runs^{\mathscr{M}}(FRuns^{\mathscr{M}})$ for the set of runs (finite runs) of the MDP $\mathscr{M}$ and $Runs^{\mathscr{M}}(s)(FRuns^{\mathscr{M}}(s))$ for the set of runs (finite runs) of the MDP $\mathscr{M}$ starting from the state s. We write $last(r)$ for the last state of a finite run r.

A strategy in $\mathscr{M}$ is a function $\sigma\colon FRuns \to \mathscr{D}(A)$ such that $supp(\sigma(r)) \subseteq A(last(r))$, where $supp(d)$ denotes the support of the distribution d. A memory skeleton is a tuple $M = (M, m_0, \alpha_u)$ where M is a finite set of memory states, m_0 is the initial state, and $\alpha_u : M \times \Sigma \to M$ is the memory update function. We define the extended memory update function $\hat{\alpha}_u : M \times \Sigma^* \to M$ in a straightforward way. A finite memory strategy for $\mathscr{M}$ over a memory skeleton M is a Mealy machine (M, α_x) where $\alpha_x : S \times M \to \mathscr{D}(A)$ is the *next action function* that suggests the next action based on the MDP and memory state. The semantics of a finite memory strategy (M, α_x) is given as a strategy $\sigma : FRuns \to \mathscr{D}(A)$ such that for every $r \in FRuns$ we have that $\sigma(r) = \alpha_x(last(r), \hat{\alpha}_u(m_0, L(r)))$.

A strategy σ is *pure* if $\sigma(r)$ is a point distribution for all runs $r \in FRuns^{\mathscr{M}}$ and is *mixed* (short for strictly mixed) if $supp(\sigma(r)) = A(last(r))$ for all runs $r \in FRuns^{\mathscr{M}}$. Let $Runs^{\mathscr{M}}_{\sigma}(s)$ denote the subset of runs $Runs^{\mathscr{M}}(s)$ that correspond to strategy σ with initial state s. Let $\Pi_{\mathscr{M}}$ be the set of all strategies. We say that σ is *stationary* if $last(r) = last(r')$ implies

$\sigma(r) = \sigma(r')$ for all runs $r, r' \in FRuns^{\mathscr{M}}$. A stationary strategy can be given as a function $\sigma : S \to \mathscr{D}(A)$. A strategy is *positional* if it is both pure and stationary.

An MDP $\mathscr{M}$ under a strategy σ results in a Markov chain $\mathscr{M}_\sigma$. If σ is a finite memory strategy, then $\mathscr{M}_\sigma$ is finite-state Markov chain. The behaviour of an MDP $\mathscr{M}$ under a strategy σ and starting state $s \in S$ is defined on a probability space $(Runs_\sigma^{\mathscr{M}}(s), \mathscr{F}_{Runs_\sigma^{\mathscr{M}}(s)}, \Pr_\sigma^{\mathscr{M}}(s))$ over the set of infinite runs of σ with starting state s. Given a random variable $f : Runs^{\mathscr{M}} \to \mathbb{R}$, we denote by $\mathbb{E}_\sigma^{\mathscr{M}}(s)\{f\}$ the expectation of f over the runs of $\mathscr{M}$ originating at s that follow σ.

A sub-MDP of $\mathscr{M}$ is an MDP $\mathscr{M}' = (S', A', T', AP, L')$, where $S' \subset S$, $A' \subseteq A$ is such that $A'(s) \subseteq A(s)$ for every $s \in S'$, and T' and L' are analogous to T and L when restricted to S' and A'. Moreover, $\mathscr{M}'$ is closed under probabilistic transitions. An *end-component* [30] of an MDP $\mathscr{M}$ is a sub-MDP $\mathscr{M}'$ such that for every state pair $s, s' \in S'$ there is a strategy that can reach s' from s with positive probability. A maximal end-component is an end-component that is maximal under set-inclusion. Every state s of an MDP $\mathscr{M}$ belongs to at most one maximal end-component. An MDP $\mathscr{M}$ is *communicating* if it is equal to its maximal end-component. A *bottom strongly connected component* (BSCC) of a Markov chain is any of its end-components.

Reward machines. In the classical RL literature, the learning objective is specified using Markovian reward functions, i.e. a function $\rho : S \times A \to \mathbb{R}$ assigning utility to state-action pairs. A *rewardful* MDP is a tuple $\mathscr{M} = (S, s_0, A, T, \rho)$ where S, s_0, A, and T are defined in a similar way as for MDPs, and ρ is a Markovian reward function. A rewardful MDP $\mathscr{M}$ under a strategy σ determines a sequence of random rewards $\rho(X_{i-1}, Y_i)_{i \geq 1}$, where X_i and Y_i are the random variables denoting the i-th state and action, respectively. For $\lambda \in [0, 1[$, the *discounted reward* $Disct(\lambda)_\sigma^{\mathscr{M}}(s)$ is defined as

$$\lim_{N \to \infty} \mathbb{E}_\sigma^{\mathscr{M}}(s) \left\{ \sum_{1 \leq i \leq N} \lambda^{i-1} \rho(X_{i-1}, Y_i) \right\},$$

while the *average reward* $Avg_\sigma^{\mathscr{M}}(s)$ is defined as

$$\limsup_{N \to \infty} \frac{1}{N} \mathbb{E}_\sigma^{\mathscr{M}}(s) \left\{ \sum_{1 \leq i \leq N} \rho(X_{i-1}, Y_i) \right\}.$$

For an objective $Reward^{\mathscr{M}} \in \{Disct(\lambda)^{\mathscr{M}}, Avg^{\mathscr{M}}\}$ and state s, we define the optimal reward $Reward_*^{\mathscr{M}}(s)$ as $\sup_{\sigma \in \Pi_{\mathscr{M}}} Reward_\sigma^{\mathscr{M}}(s)$. A strategy σ is optimal for $Reward^{\mathscr{M}}$ if

$Reward_{\sigma}^{\mathscr{M}}(s)$=$Reward_{*}^{\mathscr{M}}(s)$ for all $s\in S$. The optimal cost and strategies for these objectives can be computed in polynomial time [124].

Often, complex learning objectives cannot be expressed using Markovian reward signals. A recent trend is to express learning objectives using finite-state reward machines [65]. We require a more expressive variant of reward machine capable of ε transitions and nondeterminism. We call them nondeterministic reward machines. A (nondeterministic) reward machine is a tuple $\mathscr{R} = (\Sigma_\varepsilon, U, u_0, \delta_r, \rho)$ where U is a finite set of states, $u_0 \in U$ is the starting state, $\delta_r : U \times \Sigma_\varepsilon \to 2^U$ is the transition relation, and $\rho : U \times \Sigma_\varepsilon \times U \to \mathbb{R}$ is the reward function, where $\Sigma_\varepsilon = (\Sigma \cup \{\varepsilon\})$ and ε is a special silent transition.

Given an MDP $\mathscr{M} = (S, s_0, A, T, AP, L)$ and a reward machine $\mathscr{R} = (\Sigma_\varepsilon, U, u_0, \delta_r, \rho)$ over the alphabet $\Sigma = 2^{AP}$, their product

$$\mathscr{M} \times \mathscr{R} = (S \times U, s_0 \times u_0, (A \times U) \cup \{\varepsilon\}, T^\times, \rho^\times)$$

is a rewardful MDP where $T^\times : (S\times U) \times ((A\times U)\cup\{\varepsilon\}) \to \mathscr{D}(S\times U)$ is such that $T^\times((s,u),\alpha)((s',u'))$ equals

$$\begin{cases} T(s,a)(s') & \text{if } \alpha = (a,u') \text{ and } (u,L(s),u') \in \delta_r \\ 1 & \text{if } \alpha = \varepsilon \text{ and } s = s' \text{ and } \delta(u,\varepsilon,u') \in \delta_r \\ 0 & \text{otherwise.} \end{cases}$$

and $\rho^\times : (S\times U) \times ((A\times U)\cup\{\varepsilon\}) \times (S\times U) \to \mathbb{R}$ is defined such that $\rho^\times((s,u),\alpha,(s',u'))$ equals

$$\begin{cases} \rho(u,L(s),u') & \text{if } \alpha = (a,u') \text{ and } (u,L(s),u') \in \delta_r \\ \rho(u,\varepsilon,u') & \text{if } \alpha = \varepsilon. \end{cases}$$

For technical convenience, we assume that $\mathscr{M}\times\mathscr{R}$ contains only reachable states from (s_0,u_0). For both discounted and average objectives, the optimal strategies of $\mathscr{M}\times\mathscr{R}$ are positional on $\mathscr{M}\times\mathscr{R}$. Moreover, these positional strategies characterise a finite memory strategy (with a memory skeleton based on the states of $\mathscr{R}$ and the next-action function based on the positional strategy) over $\mathscr{M}$ maximising the learning objective given by $\mathscr{R}$.

Omega-Regular Specifications. Formal specification languages, such as ω-automata and logical based objectives, provide a rigorous and unambiguous mechanism to express learning objective over continuing tasks. There is a growing trend [57, 130, 59, 19, 80] in expressing

learning objectives in RL using linear temporal logic (LTL) and ω-regular languages (that strictly generalise LTL). We will express ω-regular languages as *good-for-MDP* Büchi automata [58].

LTL [8] is a temporal logic whose formulae describe a subset of the ω-regular languages, which is often used to specify objectives in human-readable form. Given a set of atomic propositions AP, the LTL formulae over AP can be defined via the following grammar:

$$\varphi := a \in AP \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi. \tag{6.3.1}$$

Additional operators are defined as abbreviations: $\top \stackrel{\text{def}}{=} a \vee \neg a$; $\bot \stackrel{\text{def}}{=} \neg\top$; $\varphi \wedge \psi \stackrel{\text{def}}{=} \neg(\neg\varphi \vee \neg\psi)$; $\varphi \rightarrow \psi \stackrel{\text{def}}{=} \neg\varphi \vee \psi$; $\mathsf{F}\varphi \stackrel{\text{def}}{=} \top \mathsf{U} \varphi$; and $\mathsf{G}\varphi \stackrel{\text{def}}{=} \neg\mathsf{F}\neg\varphi$. We write $w \models \varphi$ if ω-word w over 2^{AP} satisfies LTL formula φ. The satisfaction relation is defined inductively [8]. Every LTL formula can be converted [137, 82] into a Good-for-MDP Büchi automaton, defined later.

Nondeterministic Büchi automata are finite state machines capable of expressing all ω-regular languages. Formally, a (nondeterministic) *Büchi automaton* is a tuple $\mathscr{A} = (\Sigma, Q, q_0, \delta, F)$, where Σ is a finite *alphabet*, Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $\delta \colon Q \times \Sigma \to 2^Q$ is the *transition function*, and $F \subset Q \times \Sigma \times Q$ is the set of *accepting transitions*.

A *run* r of $\mathscr{A}$ on $w \in \Sigma^\omega$ is an ω-word $r_0, w_0, r_1, w_1, \ldots$ in $(Q \times \Sigma)^\omega$ such that $r_0 = q_0$ and, for $i > 0$, $r_i \in \delta(r_{i-1}, w_{i-1})$. Each triple (r_{i-1}, w_{i-1}, r_i) is a *transition* of $\mathscr{A}$. We write $\mathsf{inf}(r)$ for the set of transitions that appear infinitely often in the run r. A run r of $\mathscr{A}$ is *accepting* if $\mathsf{inf}(r) \cap F \neq \emptyset$. The *language* $\mathscr{L}(\mathscr{A})$ of $\mathscr{A}$ is the subset of words in Σ^ω that have accepting runs in $\mathscr{A}$. A language is *ω-regular* if it is accepted by a Büchi automaton.

Given an MDP $\mathscr{M}$ and an ω-regular objective φ given as an ω-automaton $\mathscr{A}_\varphi = (\Sigma, Q, q_0, \delta, F)$, we want to compute an optimal strategy satisfying the objective. We define the satisfaction probability of σ from starting state s as:

$$\mathsf{PSem}^{\mathscr{M}}_{\mathscr{A}}(s, \sigma) = \Pr^{\mathscr{M}}_{\sigma}(s)\left\{ r \in \mathit{Runs}^{\mathscr{M}}_{\sigma}(s) \colon L(r) \in \mathscr{L}(\mathscr{A}) \right\}.$$

The optimal satisfaction probability $\mathsf{PSem}^{\mathscr{M}}_{\mathscr{A}}(s)$ for specification $\mathscr{A}$ is defined as $\sup_{\sigma \in \Pi_{\mathscr{M}}} \Pr^{\mathscr{M}}_{\sigma}(s, \sigma)$ and we say that $\sigma \in \Pi_{\mathscr{M}}$ is an optimal strategy for $\mathscr{A}$ if $\mathsf{PSem}^{\mathscr{M}}_{\mathscr{A}}(s, \sigma)(s) = \mathsf{PSem}^{\mathscr{M}}_{\mathscr{A}}(s)$.

Given an MDP $\mathscr{M} = (S, s_0, A, T, AP, L)$ and automaton $\mathscr{A} = (2^{AP}, Q, q_0, \delta, F)$, the *product* $\mathscr{M} \times \mathscr{A} = (S \times Q, (s_0, q_0), A \times Q, T^\times, F^\times)$ is an MDP with initial state (s_0, q_0) and

accepting transitions $F^\times$ where $T^\times : (S\times Q)\times(A\times Q) \rightharpoonup \mathscr{D}(S\times Q)$ is defined by

$$T^\times((s,q),(a,q'))((s',q')) = \begin{cases} T(s,a)(s') & \text{if } (q,L(s,a,s'),q')\in\delta \\ 0 & \text{otherwise.} \end{cases}$$

The accepting transitions $F^\times \subseteq (S\times Q)\times(A\times Q)\times(S\times Q)$ is defined by $((s,q),(a,q'),(s',q')) \in F^\times$ if, and only if, $(q,L(s,a,s'),q') \in F$ and $T(s,a)(s') > 0$. A strategy $\sigma^\times$ on the product defines a strategy σ on the MDP with the same value, and vice versa. Note that for a stationary $\sigma^\times$, the strategy σ may need memory. End-components and runs of the product MDP are defined just like for MDPs.

A run of $\mathscr{M}\times\mathscr{A}$ is accepting if $\inf(r)\cap F^\times \neq \emptyset$. We define the *syntactic satisfaction* probabilities $PSat^{\mathscr{M}}_{\mathscr{A}}((s,q),\sigma^\times)$ as the probability of accepting runs, i.e.

$$\Pr^{\mathscr{M}\times\mathscr{A}}_{\sigma^\times}(s,q)\{r \in Runs^{\mathscr{M}\times\mathscr{A}}_{\sigma^\times}(s,q) : \inf(r)\cap F^\times \neq \emptyset\}$$

Similarly, we define $PSat^{\mathscr{M}}_{\mathscr{A}}(s)$ as the optimal probability over the product, i.e. $\sup_{\sigma^\times}\left(PSat^{\mathscr{M}}_{\mathscr{A}}((s,q_0),\sigma^\times)\right)$. For a deterministic $\mathscr{A}$ the equality $PSat^{\mathscr{M}}_{\mathscr{A}}(s) = \mathsf{PSem}^{\mathscr{M}}_{\mathscr{A}}(s)$ holds; however it is not guaranteed for nondeterministic Büchi automata as the optimal resolution of nondeterministic choices may require access to future events. This motivates for the definition of a good-for-MDP nondeterminisitc Büchi automata. A Büchi automaton $\mathscr{A}$ is *good for MDPs* (GFM), if $PSat^{\mathscr{M}}_{\mathscr{A}}(s_0) = \mathsf{PSem}^{\mathscr{M}}_{\mathscr{A}}(s_0)$ holds for all MDPs $\mathscr{M}$ and starting states s_0 [58]. Note that every ω-regular objective can be expressed as a GFM automaton [58]. A popular class of GFM automata is suitable limit-deterministic Büchi automata [56, 137]. This chapter considers only GFM Büchi automata.

The satisfaction of an ω-regular objective given as a GFM automaton $\mathscr{A}$ by an MDP $\mathscr{M}$ can be formulated in terms of the accepting maximal end-components of the product $\mathscr{M}\times\mathscr{A}$, i.e. the maximal end-component that contains an accepting transition from $\mathbf{F}^\times$. The optimal satisfaction probabilities and strategies can be computed by computing the accepting maximal end-component of $\mathscr{M}\times\mathscr{A}$ and then maximizing the probability to reach states in such components. The optimal strategies are positional on $\mathscr{M}\times\mathscr{A}$ and characterise a finite memory strategy over $\mathscr{M}$ maximising the satisfaction probability of the learning objective given by $\mathscr{A}$.

Reinforcement learning. Given an MDP $\mathscr{M}$, reward machine $\mathscr{R}$, and an optimisation objective (discounted or average reward), an optimal strategy can be computed in polynomial time using linear programming [124]. Similarly, graph-theoretic techniques to find maximal end-components can be combined with linear programming to compute optimal strategies

for ω-regular objectives [56]. However, when the transition/reward structure of the MDP is unknown, such techniques are not applicable.

Reinforcement learning [160] (RL) is a sampling-based optimisation approach where an agent learns to optimise its strategy by repeatedly interacting with the environment relying on the reinforcements (numerical reward signals) it receives for its actions. We focus on model-free approach to RL where the learner computes optimal strategies without explicitly estimating the transition probabilities and rewards. These approaches are asymptotically space-efficient [157] than model-based RL and have been shown to scale well [112, 138]. Some prominent model-free RL algorithms for discounted and average reward objectives include Q-learning and TD [160] and differential Q-learning [169].

In some applications, such as running a maze or playing tic-tac-toe—the interaction between the agent and the environment naturally breaks into finite length learning sequences, called episodes. Thus the agent optimises its strategy by combining its experience over different episodes. We call such tasks *episodic*. On the other hand, for some applications—such as process control and reactive systems—this interaction continues ad-infinitum and the agent lives and learns over a single lifetime. We call such tasks *continuing*.

6.4 Construction and correctness

Let us fix a communicating MDP $\mathscr{M} = (S, s_0, A, T, AP, L)$ and an absolute liveness GFM property $\mathscr{A} = (\Sigma, Q, q_0, \delta, F)$ for the rest of this section. Our goal is to learn a reward machine $\mathscr{R}$ such that we can use an off-the-shelf average reward RL on $\mathscr{M} \times \mathscr{R}$ to compute an optimal strategy of $\mathscr{M}$ against $\mathscr{A}$.

Since the optimal strategies are not positional on $\mathscr{M}$ but rather positional on $\mathscr{M} \times \mathscr{A}$, it is natural to assume that the reward machine $\mathscr{R}$ takes the structure of $\mathscr{A}$ with a reward function providing positive reinforcement with every accepting transition. Unfortunately, even for absolute liveness GFM automata $\mathscr{A}$, the product $\mathscr{M} \times \mathscr{A}$ with a communicating MDP $\mathscr{M}$ may not be communicating.

Example 6.1 *Assume a communicating MDP M with at least one state labelled a or b, and the absolute liveness property $\varphi = \mathsf{F}(\mathsf{G}a \vee \mathsf{G}\mathsf{F}b)$ and its automaton shown in Fig. 6.1. Observe that any run that visits one of the two accepting states cannot visit the other one. Hence, the product does not satisfy the communicating property.*

Reward machine construction Let $\mathscr{A} = (\Sigma, Q, q_0, \delta, F)$ be an absolute liveness GFM automaton. Consider $\mathscr{R}_{\mathscr{A}} = (\Sigma_\varepsilon, Q, q_0, \delta', \rho)$ where $\delta'(q,a) = \delta(q,a)$ for all $a \in \Sigma$ and ε

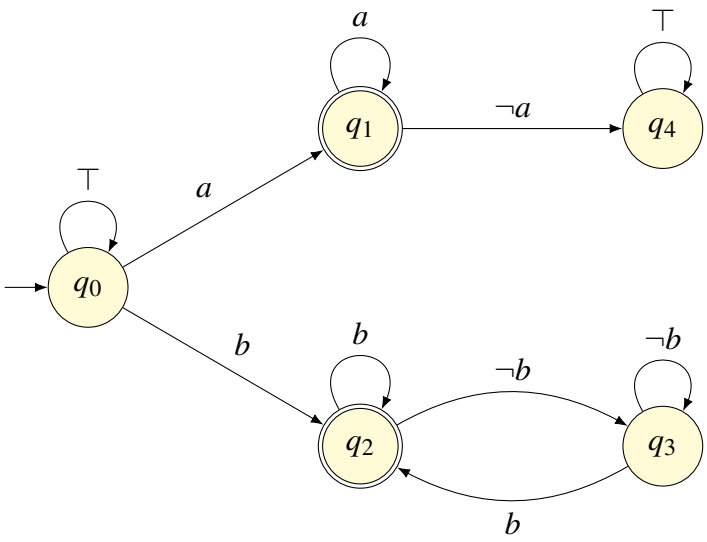


Fig. 6.1 A Büchi automaton for $\varphi = \mathsf{F}(\mathsf{G}a \vee \mathsf{GF}b)$ [79].

transitions reset to the starting state, i.e. $\delta'(q,\varepsilon) = q_0$. Note that by adding the reset (ε) action from every state of $\mathscr{R}$ to its initial state, the graph structure of $\mathscr{M}$ is strongly connected. The reward function $\rho : Q \times \Sigma \cup \{\varepsilon\} \times Q \to \mathbb{R}$ is such that

$$\rho(q,a,q') = \begin{cases} c & \text{if } a = \varepsilon \\ 1 & \text{if } (q,a,q') \in F \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 6.1 (Preservation of communication) *For a communicating MDP $\mathscr{M}$ and reward machine $\mathscr{R}_{\mathscr{A}}$ for an absolute liveness GFM automaton $\mathscr{A}$, we have that the product $\mathscr{M} \times \mathscr{R}_{\mathscr{A}}$ is communicating.*

Proof 6.1 *To show that $\mathscr{M} \times \mathscr{R}_{\mathscr{A}}$ is communicating, we need to show that for arbitrary states $(s,q),(s',q') \in S \times Q$ reachable from the initial state (s_0,q_0), we have that there is a strategy that can reach (s',q') from (s,q) with positive probability. Note that since $\mathscr{M}$ is communicating, it is possible to reach (s_0,q') from (s,q) for some q' of $\mathscr{R}_{\mathscr{A}}$ using a strategy to reach s_0 from s in $\mathscr{M}$. We can then use a reset (ε) action in $\mathscr{R}_{\mathscr{A}}$ to reach the state (s_0,q_0). Since (s',q') is reachable from the initial state (s_0,q_0), we have a strategy to reach (s',q') from (s,q) with positive probability.*

Lemma 6.2 (Average and probability) *There exists a $c^* < 0$ such that for all $c < c^*$, positional strategies that maximise the average reward on $\mathscr{M} \times \mathscr{R}_{\mathscr{A}}$ will maximise the satisfaction probability of $\mathscr{A}$.*

Proof 6.2 *The proof is in three parts.*

1. *First observe that if $c < 0$, then for any average-reward optimal strategy in $\mathscr{M} \times \mathscr{R}_{\mathscr{A}}$, the expected average reward is non-negative. This is so because all other actions except ε actions provide non-negative rewards. Hence, any strategy that takes ε actions only finitely often, results in a non-negative average reward.*

2. *Let Π^* be the set of positional strategies in $\mathscr{M} \times \mathscr{R}_{\mathscr{A}}$ such that the ε actions are taken only finitely often, i.e. no BSCC of the corresponding Markov chain contains an ε transition. Let Π^{ε} be the set of remaining positional strategies, i.e., the set of positional strategies that visit an ε transition infinitely often. Let $0 < p_{\mathtt{min}} < 1$ be a lower bound on the expected long-run frequency of the ε transitions among all strategies in Π^{ε}. Let $c_* = -1/p_{\mathtt{min}}$. Observe that for every policy $\sigma' \in \Pi^{\varepsilon}$, the expected average reward is negative and cannot be an optimal strategy in $\mathscr{M} \times \mathscr{R}_{\mathscr{A}}$. To see that, let $0 < p \leq 1$ be the the long-run frequency of the ε transitions for σ and let $0 \leq q < 1$ be the long-run frequency of visiting accepting transitions for σ. The average reward for σ is*

$$\begin{aligned} Avg_{\sigma}^{\mathscr{M}\times\mathscr{R}_{\mathscr{A}}}(s_0,q_0) &= p\cdot c + q\cdot 1 + (1-p-q)\cdot 0 \\ &\leq p\cdot c + q\cdot 1 + (1-p-q)\cdot 1 \\ &= p\cdot c + (1-p) \\ &\leq p\cdot c_* + (1-p) \\ &= -p/p_{\mathtt{min}} + (1-p) \\ &\leq -1 + (1-p) \leq -p. \end{aligned}$$

   Since every optimal policy must have a non-negative average reward, no policy in Π^{ε} is optimal for $c < c_*$.

3. *Now consider an optimal policy σ^* in Π^*. We show that this policy also optimises the probability of satisfaction of $\mathscr{A}$. There are two cases to consider.*

   (a) *If the expected average reward of σ^* is 0, then under no strategy it is possible to reach an accepting transition (positive reward transition) in $\mathscr{M} \times \mathscr{R}_{\mathscr{A}}$. Hence, every policy is optimal in $\mathscr{M}$ against $\mathscr{A}$, and so is σ^*.*

   (b) *If the expected average reward of σ^* is positive, then notice that for every BSCC of the Markov chain of $\mathscr{M} \times \mathscr{R}_{\mathscr{A}}$ under σ_*, the average reward is the same. This is so because otherwise, there is a positional policy that reaches the BSCC with the optimal average from all the other BSCCs with lower averages, contradicting the optimality of σ_*. Since for an optimal policy σ_*, every BSCC provides*

the same positive average, every BSCC must contain an accepting transition. Hence, every run of the MDP $\mathscr{M}$ under σ_* will eventually dwell in an accepting component and in the process will see a finitely many ε (reset) transitions. For any such given run r, consider the the suffix r' of the run after the last ε transition is taken and let $r = wr'$ for some finite run w. Since $L(r')$ is an accepting word in $\mathscr{A}$, and since $\mathscr{A}$ is an absolute liveness property any arbitrary prefix w' to this run r' is also accepting. This implies that the original run r is also accepting for $\mathscr{A}$. It follows that for such a strategy σ_*, the probability of satisfaction of $\mathscr{A}$ is 1, making σ_* an optimal policy for $\mathscr{M}$ against $\mathscr{A}$.

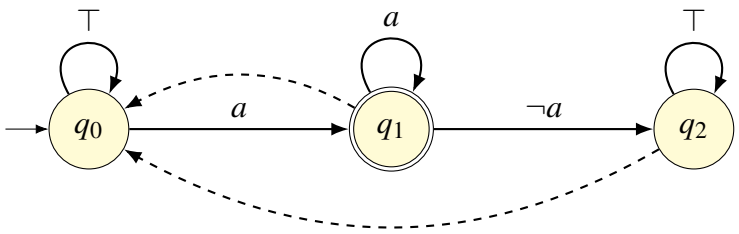
The proof is now complete.

Since our translation from ω-regular objective to reward machines is model-free, the following theorem is immediate.

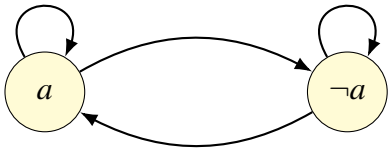
Theorem 6.1 (Convergence of model-free RL) *Differential Q-learning algorithm for maximising average reward objective on $\mathscr{M} \times \mathscr{R}_{\mathscr{A}}$ will converge to a strategy maximising the probability of satisfaction of $\mathscr{A}$ for a suitable value of c. Moreover, the product construction $\mathscr{M} \times \mathscr{R}_{\mathscr{A}}$ can be done on-the-fly and it is model-free.*

As an example, consider the property $\mathsf{F}\,\mathsf{G}\,a$ and an MDP with two states and all transitions between states are available as deterministic actions (Fig. 6.2). Only one of the states is labeled a. An infinite memory strategy could see a for one step, reset, then see two as, reset, then see three as and so forth. This strategy will produce the same average value as the positional strategy which sees a forever without resetting. However, the infinite memory strategy will fail the property while the positional one will not.

Shaping rewards via hard resets. For a Büchi automaton $\mathscr{A}$, we say that its state $q \in Q$ is *coaccessible* if there exists a path starting from that state to an accepting transition. If a state is not coaccessible then any run of the product $\mathscr{M} \times \mathscr{A}$ that ends in such a state will never be accepting, and hence one can safely redirect all of its outgoing transitions to the initial state with reward c (a hard reset). Such hard resets will promote speedy learning by reducing the time spent in such states during unsuccessful explorations, and at the same time adding these resets does not make a non-accepting run accepting or vice versa. Lemma 6.1, Lemma 6.2, and Theorem 6.1 continue to hold with such hard resets. Introducing hard resets is a reward shaping procedure in that it is a reward transformation [117] under which optimal strategies remain invariant.



(a) Automaton of $\mathsf{F}\,\mathsf{G}\,a$, dashed lines represent reset transitions [79].



(b) MDP, each transition represents an action

Fig. 6.2 The two state MDP and a persistence property [79].

6.5 Experimental results

We implemented the reduction[1] with hard resets presented in Section 6.4. As described, we do not build the product MDP explicitly, and instead, compose it on-the-fly by keeping track of the MDP and automaton states independently. We use Differential Q-learning [169] to learn optimal, positional average reward strategies. For our experiments, we have collected a set of communicating MDPs with absolute liveness properties[2].

We compare with two previous approaches for translating omega-regular languages to rewards: the method of [57] with Q-learning and the method of [19] with Q-learning. The method of [57] translates a GFM Büchi automaton into a reachability problem through a suitable parameter ζ. This reachability problem can be solved with discounted RL by rewarding reaching the target state and using a large enough discount factor. The method of [19] uses a state dependent discount factor γ_B and a GFM Büchi automaton. By using a suitable γ_B and large enough discount factor, one can learn optimal strategies for the omega-regular objective.

RQ1. How do previous approaches perform in the continuing setting? The methods of [57] and [19] may produce product MDPs that are not communicating (see Example 6.1). This means that a single continuing run of the MDP may not explore all relevant states and actions. Thus, previous methods are not guaranteed to converge in this setting.

[1]The implementation is available at https://plv.colorado.edu/mungojerrie/.
[2]Case studies are available at https://plv.colorado.edu/mungojerrie/aamas22.

We studied if this behaviour affects these prior methods in practice. As a baseline, we include our proposed approach. Instead of tuning hyperparameters for each method, where hyperparameters that lead to convergence may not exist, we instead take a sampling approach. We select a wide distribution over hyperparameters for each method and sample 200 hyperparameter combinations for each method and example. We then train for 10 million training steps on each combination. The hyperparameter distribution we selected is $\alpha \sim \mathscr{D}(0.01, 0.5)$, $\varepsilon \sim \mathscr{D}(0.01, 1.0)$, $c \sim \mathscr{D}(1, 200)$, $\eta \sim \mathscr{D}(0.01, 0.5)$, $\zeta \sim \mathscr{D}(0.5, 0.995)$, $\gamma_B \sim \mathscr{D}(0.5, 0.995)$, and discount factor $\gamma \sim \mathscr{D}(0.99, 0.99999)$ where $\mathscr{D}(a, b)$ is a log-uniform distribution from a to b. The end points of these distributions and the training amount was selected by finding hyperparameters which led to convergence in the episodic setting for these methods.

Figure 6.3 shows the resulting distribution over runs. A distribution entirely at 0 indicates that all sampled runs produced strategies that satisfy the property with probability 0. A

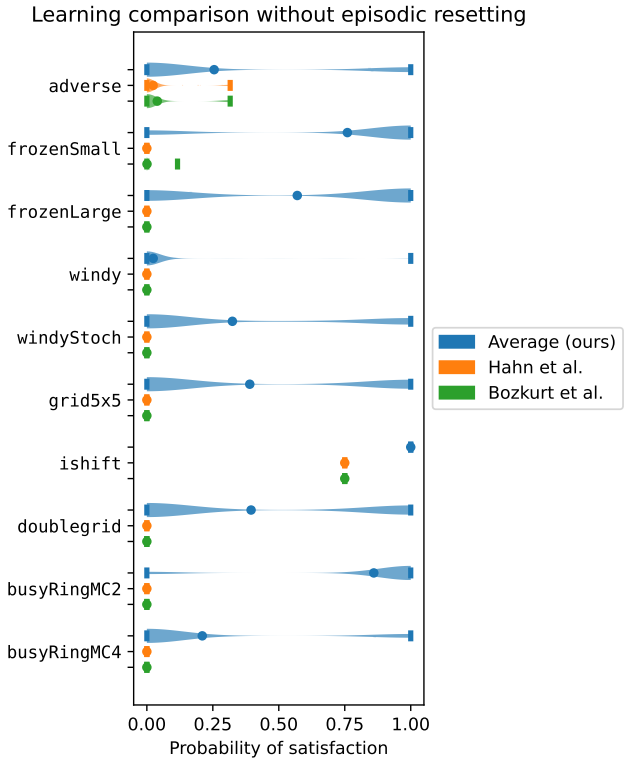


Fig. 6.3 Comparison of the distributions of probability of satisfaction of learned policies across sampled hyperparameters in the continuing setting. For each distribution, the mean is shown as a circle, and the maximum and minimum are shown as vertical bars. We compare our proposed reduction, the reduction of [57] with Q-learning, and the reduction of [19] with Q-learning. Episodic resetting was not used [79].

Name	states	prod.	time	time†	time‡	c	ε	α	η	train-steps
`adverse`	202	507	8.51	7.09	12.56	-150		0.2		10M
`frozenSmall`	16	64	0.99	20.23	9.88					500k
`frozenLarge`	64	256	4.07	3.88	8.79			0.02	0.02	3M
`windy`	123	366	1.40	1.81	2.61		0.95	0.5	0.05	1M
`windyStoch`	130	390	2.97	3.91	2.53			0.5		2M
`grid5x5`	25	100	0.62	1.12	1.02			0.5		200k
`ishift`	4	29	0.03	0.01	0.02					10k
`doublegrid`	1296	5183	16.43	3.45	3.09	-2	0.5	0.05	0.01	12M
`busyRingMC2`	72	288	0.03	0.03	0.03				0.01	10k
`busyRingMC4`	2592	15426	6.08	3.94	2.33				0.01	1.5M

Table 6.1 Learning results and comparison. Hyperparameters used for our reduction are shown. Blank entries indicate that default values were used. The default parameters are $c = -1$, $\varepsilon = 0.1$, $\alpha = 0.1$, and $\eta = 0.1$. Times are in seconds. Superscript † indicates results from Q-learning with reduction from [57], while superscript ‡ indicates Q-learning with reduction from [19]. Results for † and ‡ required episodic resetting. All hyperparameters were tuned by hand [79].

distribution entirely at 1 indicates that all sampled runs produced strategies that satisfy the property with probability 1. *For many examples, prior approaches had no successful hyperparameter combinations, with distributions centered entirely at* 0. However, our proposed approach always had some hyperparameters that led to optimal, probability 1, strategies, as indicated by the tails of the distributions touching the probability 1 region of the plot.

RQ2. How does our method compare to previous approaches when we allow episodic setting? By allowing episodic resetting, we can now find hyperparameters for previous methods that lead to convergence. We tuned all hyperparameters by hand to minimise training time while verifying with a model checker that the produced strategies are optimal. Table 6.1 shows learning times, as well as hyperparameters for our reduction. We report the number of states reachable in the MDP and the product, learning times averaged over 5 runs, the reset penalty c, the ε-greedy exploration rate ε, the Differential Q-learning learning rates α and η, as well as the number of training steps. Note that we do not do any episodic resetting when training with our reduction. This means that the RL agent must learn to recover from mistakes during training, while previous approaches are periodically reset to a good initial state. *Our reduction using Differential Q-learning is competitive with previous approaches while not being reliant on episodic resetting.*

6.6 Related work

The development and use of formal reward structures for RL have witnessed an increased interest in recent years. For episodic RL, logics have been developed over finite traces of agent behavior, including LTL_f and Linear Dynamic Logic (LDL_f) [32, 23]. These logics have equivalent automaton and reward machine representations. These representations have catalysed a series of efforts on defining novel reward shaping functions to accelerate the convergence of RL algorithms subject to formal specifications [31, 65, 24]. These methods leverage the graph structure of the automaton to provide an artificial reward signal to the agent. More recently, dynamic reward shaping using LTL_f has been introduced as a means to both learn the transition values of a given reward machine and leverage these values for reward shaping and transfer learning [167]. There has also been work on learning or synthesising the entire structure of such reward machines from agent interactions with the environment by leveraging techniques from satisfiability and active grammatical inference [116, 177, 47, 178, 166].

For the infinite-trace settings, LTL has been extensively used to verify properties and synthesise policies formally using the mathematical model of a system [8, 16, 104, 80, 89, 180]. The notion of coupling between stochastic systems is developed in [54, 53, 55] with extensions to multi-objective setting [52] to check infinite-horizon properties.

Considering the generality of the results in terms of the structure of the underlying MDP, most of the research focuses on discounted reward structures. Despite the simplicity of discounted Markov decision problems, the discounted reward structure (unlike average reward) prioritises the transient response of the system. However, the application of the average reward objective because of the restriction over the structure of the MDP is limited. The work [37] proposes a policy iteration algorithm for satisfying LTL properties of the form $\mathsf{G}\mathsf{F}\,\phi \wedge \psi$ for a communicating MDP almost surely. In [6], the authors propose a value iteration algorithm for solving the average reward problem for multichain MDPs. In this way, the algorithm first computes the optimal value for each of the strongly connected components of the MDP and then weighted reachability to find the optimal policy. The work [7] provides a linear programming formulation for policy synthesis of multichain MDPs with steady-state constraints.

In the last few years, researchers have started developing data-driven policy synthesis techniques in order to satisfy temporal properties. There is a large body of literature in safe reinforcement learning (RL) (see e.g. [48, 126, 40]). The problem of learning a policy to maximise the satisfaction probability of a temporal property using discounted RL is studied recently [20, 46, 130, 19, 59, 61, 119]. The work [57] by using a parameterised augmented MDP provides an RL-based policy synthesis for finite MDPs with unknown

transition probabilities. It shows that the optimal policy obtained by RL for the reachability probability on the augmented MDP gives a policy for the MDP with a suitable convergence guarantee. In [19] authors provide a path-dependent discounting mechanism for the RL algorithm based on a limit-deterministic Buchi automaton (LDBA) representation of the underlying omega-regular property and prove convergence of their approach on finite MDPs when the discounting factor goes to one. An LDBA is also leveraged in [59, 61, 119] for discounted-reward model-free RL in both continuous- and discrete-state MDPs. The LDBA is used to define a reward function that incentivises the agent to visit all accepting components of the automaton. These works use episodic discounted RL with a discount factor close to one to solve the policy synthesis problem. There are two issues with the foregoing approaches. First, because of the episodic nature of the algorithms, they are not applicable in continuing settings. Second, because of high discount factors in practice, these algorithms are difficult to converge. On the other hand, recent work on reward shaping for average reward RL has been explored based on safety properties to be satisfied by the synthesised policy [72]. In contrast to the solution proposed in this chapter, the preceding approach requires knowledge of the graph structure of the underlying MDP and does not account for absolute liveness properties.

There is a rich history of studies in average reward RL [35, 100]. The lack of stopping criteria for multichain MDPs affects the generality of model-free RL algorithms. In this way, all model-free RL algorithms put some restrictions on the structure of MDP (e.g. ergodicity [2, 173] or communicating property). The closest line of work to this work is to use the average reward objective for safe RL. The work [141] proposes a model-based RL algorithm for maximising average reward objective with safety constraints for communicating MDPs. It is worth noting that in multichain settings, the state-of-the-art learning algorithms use model-based RL algorithms. The work [83] studies satisfaction of ω-regular properties using data-driven approaches. The authors introduce an algorithm where the optimality of the policy is conditioned to not leaving the corresponding maximal end component which leads to a sub-optimal solution. The authors provide PAC analysis for the algorithm as well. Despite all the efforts to use data-driven approaches for satisfying the ω-regular properties, there is a gap in using average reward model-free RL algorithms for satisfying temporal properties.

This chapter is an attempt to close this gap by proposing a model-free average reward RL algorithm for a subclass of LTL properties called absolute liveness properties. We claim this subclass captures a large class of interesting properties and is suitable for average reward RL. Furthermore, the eventual satisfaction semantics of an arbitrary omega-regular or LTL specification ϕ can be captured by an absolute liveness property $\mathsf{F}\,\phi$.

6.7 Conclusion

This work addressed the problem of synthesising policies that satisfy a given absolute liveness omega-regular property in the continuing setting. Continuing tasks are concerned with the eventual satisfaction of properties, which is naturally captured by the notion of absolute liveness. Our key contribution is a model-free translation from the omega-regular specification to an average reward objective, enabling the use of off-the-shelf average reward RL. This is in contrast to existing methods in the literature that use discounted, episodic learning, which requires the ability to reset the underlying environment. Such a requirement is restrictive in some settings, and our approach avoids this episodic learning, instead learning the optimal policy in one life-long episode without resetting. Furthermore, the proposed solution does not require access to a model of the environment nor to its graph structure, thereby avoiding a common assumption made in the literature where the computation of end components is required for verification and synthesis of policies subject to some omega-regular specification.

As a result, the proposed approach can be integrated with a wide range of model-free average-reward RL algorithms. For our experiments, we applied Differential Q-learning to a range of case studies and showed that the proposed approach is successful in converging to optimal strategies under the raised assumptions. In particular, our experiments showed that the proposed approach is superior to previous methods in the continuing setting. This lends credence to the important and understudied idea that average reward RL is better-suited for continuing task settings than the more popular discounted RL. For future work, we will explore the use of function approximation in the hopes that the average reward RL can experience the same success for continuing tasks that its discounted RL counterpart has witnessed in episodic settings.

Chapter 7

Conclusion

7.1 Summary of the research and contributions

In this thesis we developed data-driven approaches to formally synthesis controller for cyber-physical systems. This approaches can be divided into two categories: abstraction based methods that rely on a simplified model of the system, and model-free methods that aim for synthesising the controller directly without constructing any (simplified) model for the system.

In Chapter 3, We proposed a method for computing finite abstractions of continuous systems with unknown dynamics that is based on the use of data. We proposed a way to compute an overapproximation of reachable sets by implementing it as a robust convex program (RCP). It is then possible to determine a feasible solution to the RCP with preset confidence by solving a scenario convex program (SCP) that corresponds to the RCP. There is no need to include the system's dynamics as part of the SCP, and all that is needed is a finite set of sample trajectories. Using the sample complexity result that we have provided, we can give a lower bound on the number of trajectories needed to achieve a certain level of confidence. Based on our analysis, we can confidently state that the computed abstraction is a valid abstraction of the system. In other words, this abstract representation accurately describes the system's behaviour over the entire state space. Using our data-driven approach, we could design a controller and enlarge its winning region to satisfy temporal properties using abstraction refinement schemes.

In Chapter 4, we proposed a policy synthesis approach for continuous-space stochastic systems with unknown dynamics. Essentially, the policy's goal was to maximise the probability that the system would satisfy a complex property written in a fragment of Linear Temporal Logic (LTL). This approach replaces the unknown system with a finite Markov decision process (MDP) without having to construct it explicitly. To find a policy and apply it

to the original continuous-space system, we applied RL to the MDP with unknown transition probabilities. We proposed a compositional method to synthesise policies for unknown stochastic networks using a novel two-player RL scheme.

In Chapter 5, We applied reinforcement learning (RL) to unknown continuous-state stochastic systems to satisfy a linear temporal logic specification. Taking into account appropriate assumptions, we formulated an optimal average reward criterion which converges linearly to the true optimal satisfaction probability. By applying a sequential algorithm and relaxing the specification, we were able to learn a lower bound on this optimal value.

In Chapter 6, we sought to synthesise policies that satisfy a set of absolute liveness omega-regular properties in a continuing setting. The notion of absolute liveness naturally captures the idea of continuing tasks, which involves the eventual satisfaction of properties. Furthermore, the proposed solution requires no access to a model of the environment or its graph. Thus, we avoid a commonly held claim in the literature, where end components must be computed for policies to be verified and synthesised.

7.2 Limitations of the research

In this thesis, we investigated the potential of data-driven control synthesis techniques with formal guarantees. The algorithms presented in this thesis have a lot of potential in real world applications but at the same time they come with their limitations. In the following I will point out some of these limitations.

In Chapter 3 we proposed a data-driven abstraction-based approach. As a limitation of this approach, we limit disturbance signals whose value does not change during the fixed sampling period. We can relax this restriction by knowing a set, possibly an overestimated set, including all the possible accumulated disturbance signals during the sampling period. Additionally, we considered a uniform distribution over the disturbance set. We can extend our method to those distributions whose probability density function is known. The sample complexity of the approach is exponential with respect to the dimension of the system. It would be interesting to study existence of such sampling approaches that require computational complexity below exponential (e.g., polynomial with respect to the space dimension).

In Chapter 4, instead of analysing the whole network monolithically, and hence facing the scalability barrier, we investigated a compositional approach that solves the optimisation problem for each subsystem in the network separately, while considering the other subsystems as adversaries in a two-player game. In this approach, we limit the class of properties to just invariant properties. Future work include considering richer class of properties beyond safety

and allow some form of communication between the subsystems (a preliminary work on this direction is under submission).

In Chapter 5 we presented an algorithm for control synthesis of continuous state systems and provided the correctness guarantees of the reward machine for satisfying temporal properties. The convergence guarantee provided in this approach is valid under a technical assumption on the underlying dynamics of the system. It is known that this assumption does hold for finite-state Markov decision processes. It would be interesting to study which class of continuous-space models satisfy the required assumption. In other words, what structural properties are needed in the underlying dynamical system such that the convergence of the algorithm to the true value is guaranteed?

As an alternative for discounted reward, in Chapter 6 we study average reward objectives and in particular how we can translate a temporal property to an average reward objective for RL algorithms. In this work we had two limitations: First, all of the state-of-the-art model-free RL algorithms have (weakly) communicating assumption over the structure of underlying MDP. Second, we need to restrict the class of properties to absolute liveness properties which is a subclass of ω-regular properties. The intuition behind these assumptions is to make sure the system can recover from the mistakes that can happen early on in the learning procedure. It would be interesting to relax these assumptions as much as possible, for instance by finding the largest class of properties where the translation to average reward would lead to a correct optimal policy.

7.3 Suggestions for future work

This thesis was an attempt to study data-driven control synthesis approaches. We developed algorithms with correctness guarantees in both episodic and continual tasks. This area of research is relatively new and there are a lot of directions we are considering for future work. In the following I mention a few of these directions regarding the contribution of each chapter.

The data-driven abstraction-based approach presented in Chapter 3 can be extended to handle a larger class of disturbances beyond piecewise constant ones. We can also parallelise the method and apply it to large-scale case studies more efficiently. The formal RL algorithm which was the focus of Chapter 5 can be extended to more general classes of properties such as hyper-properties.

The average reward objectives were presented in Chapter 6. Our future efforts will explore function approximation for the average reward objective with the hope of seeing the same success as the discounted reward RL in episodic settings. A second research direction

is to study the connection between transfer learning and lifelong learning and to incorporate their success into our approach. The third research direction is to study lexicographic and more general multi-objective scenarios in average reward framework.

The compositional approach for control synthesis was presented in Chapter 4. The class of properties we consider for this research was restricted to invariant properties. A future direction for research is to extend the approach to finite-horizon properties. A second direction is to explore using collaborative frameworks in control synthesis in decentralised multi-agent scenarios. Security and privacy in CPS has received attention recently. It would be interesting to extend the data-driven ideas of this thesis to handle challenging problems of security of CPS.

References

[1] Abate, A., Prandini, M., Lygeros, J., and Sastry, S. (2008). Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734.

[2] Abbasi-Yadkori, Y., Bartlett, P., Bhatia, K., Lazic, N., Szepesvari, C., and Weisz, G. (2019). POLITEX: Regret bounds for policy iteration using expert prediction. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3692–3702. PMLR.

[3] Adzkiya, D., Soudjani, S., and Abate, A. (2014). Finite abstractions of stochastic max-plus-linear systems. In Norman, G. and Sanders, W., editors, *Proceedings of the International Conference on Quantitative Evaluation of Systems*, volume 8657 of *LNCS*, pages 74–89. Springer Verlag.

[4] Alpern, B. and Schneider, F. B. (1985). Defining liveness. *Information Processing Letters*, 21:181–185.

[5] Althof, M. (2019). Commonroad: Vehicle models (version 2018a). Tech. rep. https://commonroad.in.tum.de, Technical University of Munich, 85748 Garching, Germany (October 2018).

[6] Ashok, P., Chatterjee, K., Daca, P., Křetínskỳ, J., and Meggendorfer, T. (2017). Value iteration for long-run average reward in Markov decision processes. In *International Conference on Computer Aided Verification*, pages 201–221. Springer.

[7] Atia, G. K., Beckus, A., Alkhouri, I., and Velasquez, A. (2020). Verifiable planning in expected reward multichain MDPs. *arXiv preprint arXiv:2012.02178*.

[8] Baier, C. and Katoen, J.-P. (2008). *Principles of model checking*. MIT press.

[9] Bajcsy, A., Bansal, S., Bronstein, E., Tolani, V., and Tomlin, C. J. (2019). An efficient reachability-based framework for provably safe autonomous navigation in unknown environments. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1758–1765. IEEE.

[10] Banerjee, T., Majumdar, R., Mallik, K., Schmuck, A.-K., and Soudjani, S. (2022a). A direct symbolic algorithm for solving stochastic rabin games. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 81–98. Springer.

[11] Banerjee, T., Majumdar, R., Mallik, K., Schmuck, A.-K., and Soudjani, S. (2022b). Fast symbolic algorithms for omega-regular games under strong transition fairness. *arXiv preprint arXiv:2202.07480*.

[12] Belta, C. and Sadraddini, S. (2019). Formal methods for control synthesis: An optimization perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:115–140.

[13] Belta, C., Yordanov, B., and Gol, E. A. (2017). *Formal methods for discrete-time dynamical systems*, volume 15. Springer.

[14] Bertsekas, D. and Shreve, S. (1996). *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific.

[15] Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

[16] Blom, H. and J. Lygeros (2006). *Stochastic Hybrid Systems: Theory and Safety Critical Applications*. Number 337 in Lecture Notes in Control and Information Sciences. Springer Verlag, Berlin Heidelberg.

[17] Borkar, V. S. and Meyn, S. P. (2000). The ODE method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, 38(2):447–469.

[18] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[19] Bozkurt, A. K., Wang, Y., Zavlanos, M. M., and Pajic, M. (2020). Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10349–10355. IEEE.

[20] Brázdil, T., Chatterjee, K., Chmelik, M., Forejt, V., Křetínskỳ, J., Kwiatkowska, M., Parker, D., and Ujma, M. (2014). Verification of Markov decision processes using learning algorithms. In *Automated Technology for Verification and Analysis (ATVA)*, pages 98–114. Springer.

[21] Brunello, A., Montanari, A., and Reynolds, M. (2019). Synthesis of LTL formulas from natural language texts: State of the art and research directions. In *26th International Symposium on Temporal Representation and Reasoning (TIME 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[22] Calafiore, G. C. and Campi, M. C. (2006). The scenario approach to robust control design. *IEEE Transactions on automatic control*, 51(5):742–753.

[23] Camacho, A., Baier, J. A., Muise, C., and McIlraith, S. A. (2018). Finite LTL synthesis as planning. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*.

[24] Camacho, A., Icarte, R. T., Klassen, T. Q., Valenzano, R. A., and McIlraith, S. A. (2019). LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *IJCAI*, volume 19, pages 6065–6073.

[25] Cheng, R., Orosz, G., Murray, R. M., and Burdick, J. W. (2019). 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.

[26] Chow, J. and Cheung, K. (1992). A toolbox for power system dynamics and control engineering education and research. *IEEE Transactions on Power Systems*, 7(4):1559–1564.

[27] Cohen, M. H. and Belta, C. (2021). Model-based reinforcement learning for approximate optimal control with temporal logic specifications. In *HSCC '21: 24th ACM International Conference on Hybrid Systems: Computation and Control, Nashville, Tennessee, May 19-21, 2021*, pages 12:1–12:11. ACM.

[28] Courcoubetis, C. and Yannakakis, M. (1995). The complexity of probabilistic verification. *Journal of the ACM (JACM)*, 42(4):857–907.

[29] Dai, B., Shaw, A., Li, L., Xiao, L., He, N., Liu, Z., Chen, J., and Song, L. (2017). Sbeed: Convergent reinforcement learning with nonlinear function approximation. *arXiv:1712.10285*.

[30] De Alfaro, L. (1998). *Formal verification of probabilistic systems*. PhD thesis, Stanford University.

[31] De Giacomo, G., Iocchi, L., Favorito, M., and Patrizi, F. (2019). Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29(1), pages 128–136.

[32] De Giacomo, G. and Vardi, M. (2015). Synthesis for LTL and LDL on finite traces. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[33] Dean, S., Mania, H., Matni, N., Recht, B., and Tu, S. (2020). On the sample complexity of the linear quadratic regulator. *Foundations of Computational Mathematics*, 20(4):633–679.

[34] Devonport, A., Saoud, A., and Arcak, M. (2021). Symbolic abstractions from data: A pac learning approach. *arXiv preprint arXiv:2104.13901*.

[35] Dewanto, V., Dunn, G., Eshragh, A., Gallagher, M., and Roosta, F. (2020). Average-reward model-free reinforcement learning: a systematic review and literature mapping. *arXiv preprint arXiv:2010.08920*.

[36] Ding, J., Kamgarpour, M., Summers, S., Abate, A., Lygeros, J., and Tomlin, C. (2013). A stochastic games framework for verification and control of discrete time stochastic hybrid systems. *Automatica*, 49(9):2665–2674.

[37] Ding, X., Smith, S. L., Belta, C., and Rus, D. (2014). Optimal control of Markov decision processes with linear temporal logic constraints. *IEEE Transactions on Automatic Control*, 59(5):1244–1257.

[38] Djeumou, F., Vinod, A. P., Goubault, E., Putot, S., and Topcu, U. (2020). On-the-fly control of unknown systems: From side information to performance guarantees through reachability. *arXiv preprint arXiv:2011.05524*.

[39] Dynkin, E. B. and Yushkevich, A. A. (1979). *Controlled Markov processes*, volume 235. Springer.

[40] Efroni, Y., Mannor, S., and Pirotta, M. (2020). Exploration-exploitation in constrained MDPs. *arXiv preprint arXiv:2003.02189*.

[41] Esfahani, P. M., Sutter, T., and Lygeros, J. (2014). Performance bounds for the scenario approach and an extension to a class of non-convex programs. *IEEE Transactions on Automatic Control*, 60(1):46–58.

[42] Fan, C., Qi, B., Mitra, S., and Viswanathan, M. (2017). Dryvr: Data-driven verification and compositional reasoning for automotive systems. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 441–461. Springer.

[43] Farahani, S. S., Majumdar, R., Prabhu, V. S., and Soudjani, S. (2018). Shrinking horizon model predictive control with signal temporal logic constraints under stochastic disturbances. *IEEE Transactions on Automatic Control*.

[44] Filar, J. and Vrieze, K. (1997). *Competitive Markov Decision Processes*. Springer.

[45] Flesch, J., Predtetchinski, A., and Sudderth, W. (2018). Simplifying optimal strategies in limsup and liminf stochastic games. *Discrete Applied Mathematics*, 251:40 – 56.

[46] Fu, J. and Topcu, U. (2014). Probably approximately correct MDP learning and control with temporal logic constraints. In *Proceedings of Robotics: Science and Systems*.

[47] Gaon, M. and Brafman, R. (2020). Reinforcement learning with non-Markovian rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34(04), pages 3980–3987.

[48] Garcıa, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480.

[49] Girard, A. and Pappas, G. J. (2007). Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5):782–798.

[50] Girard, A., Pola, G., and Tabuada, P. (2009). Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Transactions on Automatic Control*, 55(1):116–126.

[51] Grover, K., dos Santos Barbosa, F., Tumova, J., and Kretınsky, J. (2021). Semantic abstraction-guided motion planningfor scltl missions in unknown environments. In *Robotics: Science and Systems*.

[52] Haesaert, S., Nilsson, P., and Soudjani, S. (2021). Formal multi-objective synthesis of continuous-state MDPs. *IEEE Control Systems Letters*, 5(5):1765–1770.

[53] Haesaert, S. and Soudjani, S. (2020). Robust dynamic programming for temporal logic control of stochastic systems. *IEEE Transactions on Automatic Control*, 66(6):2496–2511.

[54] Haesaert, S., Soudjani, S., and Abate, A. (2017). Verification of general Markov decision processes by approximate similarity relations and policy refinement. *SIAM Journal on Control and Optimization*, 55(4):2333–2367.

[55] Haesaert, S., Soudjani, S., and Abate, A. (2018). Temporal logic control of general Markov decision processes by approximate policy refinement. *IFAC-PapersOnLine*, 51(16):73–78.

[56] Hahn, E. M., Li, G., Schewe, S., Turrini, A., and Zhang, L. (2015). Lazy probabilistic model checking without determinisation. In *International Conference on Concurrency Theory (CONCUR)*, pages 354–367.

[57] Hahn, E. M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., and Wojtczak, D. (2019). Omega-regular objectives in model-free reinforcement learning. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 395–412. Springer.

[58] Hahn, E. M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., and Wojtczak, D. (2020). Good-for-mdps automata for probabilistic analysis and reinforcement learning. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 306–323. Springer.

[59] Hasanbeig, M., Abate, A., and Kroening, D. (2019a). Certified reinforcement learning with logic guidance. *arXiv preprint arXiv:1902.00778*.

[60] Hasanbeig, M., Abate, A., and Kröning, D. (2019b). Logically-constrained neural fitted Q-iteration. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMS)*, pages 2012–2014.

[61] Hasanbeig, M., Kantaros, Y., Abate, A., Kroening, D., Pappas, G. J., and Lee, I. (2019c). Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. In *IEEE Conference on Decision and Control (CDC)*, pages 5338–5343. IEEE.

[62] Hasanbeig, M., Kroening, D., and Abate, A. (2020). Deep reinforcement learning with temporal logics. In *Formal Modeling and Analysis of Timed Systems*, pages 1–22.

[63] Hernández-Lerma, O. and Lasserre, J. B. (1996). *Discrete-time Markov control processes*, volume 30 of *Applications of Mathematics*. Springer.

[64] Hsu, K.-C., Rubies-Royo, V., Tomlin, C. J., and Fisac, J. F. (2021). Safety and liveness guarantees through reach-avoid reinforcement learning. In *Robotics: Science and Systems*.

[65] Icarte, R. T., Klassen, T., Valenzano, R., and McIlraith, S. (2018). Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116. PMLR.

[66] Icarte, R. T., Klassen, T. Q., Valenzano, R. A., and McIlraith, S. A. (2020). Reward machines: Exploiting reward function structure in reinforcement learning. *CoRR*, abs/2010.03950.

[67] Ikonen, E. and Najim, K. (2001). *Advanced process identification and control*. CRC Press.

[68] Jaakkola, T., Jordan, M. I., and Singh, S. P. (1994). Convergence of stochastic iterative dynamic programming algorithms. In *Advances in neural information processing systems*, pages 703–710.

[69] Jagtap, P., Pappas, G. J., and Zamani, M. (2020a). Control barrier functions for unknown nonlinear systems using Gaussian processes. In *Proceedings of the 59th IEEE Conference on Decision and Control (CDC)*, pages 3699–3704.

[70] Jagtap, P., Soudjani, S., and Zamani, M. (2018). Temporal logic verification of stochastic systems using barrier certificates. In *International Symposium on Automated Technology for Verification and Analysis*, pages 177–193. Springer.

[71] Jagtap, P., Soudjani, S., and Zamani, M. (2020b). Formal synthesis of stochastic systems via control barrier certificates. *IEEE Transactions on Automatic Control*.

[72] Jiang, Y., Bharadwaj, S., Wu, B., Shah, R., Topcu, U., and Stone, P. (2021). Temporal-logic-based reward shaping for continuing reinforcement learning tasks. *Good Systems-Published Research*.

[73] Jothimurugan, K., Bansal, S., Bastani, O., and Alur, R. (2021). Compositional reinforcement learning from logical specifications. *Advances in Neural Information Processing Systems*, 34:10026–10039.

[74] Jouffe, L. (1998). Fuzzy inference system learning by reinforcement methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(3):338–355.

[75] Kalagarla, K. C., Jain, R., and Nuzzo, P. (2021). Model-free reinforcement learning for optimal control of markovdecision processes under signal temporal logic specifications. *arXiv preprint arXiv:2109.13377*.

[76] Kallenberg, O. (1997). *Foundations of modern probability*. Springer-Verlag, New York.

[77] Kamgarpour, M., Ellen, C., Soudjani, S., Gerwinn, S., Mathieu, J., Mullner, N., Abate, A., Callaway, D., Fränzle, M., and Lygeros, J. (2013). Modeling options for demand side participation of thermostatically controlled loads. In *International Conference on Bulk Power System Dynamics and Control (IREP)*, pages 1–15.

[78] Kazemi, M., Majumdar, R., Salamati, M., Soudjani, S., and Wooding, B. (2022a). Data-driven abstraction-based control synthesis. *arXiv preprint arXiv:2206.08069*.

[79] Kazemi, M., Perez, M., Somenzi, F., Soudjani, S., Trivedi, A., and Velasquez, A. (2022b). Translating omega-regular specifications to average objectives for model-free reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 732–741.

[80] Kazemi, M. and Soudjani, S. (2020). Formal policy synthesis for continuous-state systems via reinforcement learning. In *International Conference on Integrated Formal Methods*, pages 3–21. Springer.

[81] Kress-Gazit, H., Lahijanian, M., and Raman, V. (2018). Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:211–236.

[82] Kretínský, J., Meggendorfer, T., and Sickert, S. (2018). Owl: A library for ω-words, automata, and LTL. In *International Symposium on Automated Technology for Verification and Analysis (ATVA)*, volume 11138 of *LNCS*, pages 543–550. Springer.

[83] Kretínský, J., Michel, F., Michel, L., and Perez, G. (2020). Finite-memory near-optimal learning for Markov decision processes with long-run average reward. In *Conference on Uncertainty in Artificial Intelligence*, pages 1149–1158. PMLR.

[84] Kupferman, O. and Vardi, M. Y. (2001). Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314.

[85] Landweber, L. H. (1969). Decision problems for ω-automata. *Mathematical Systems Theory*, 3(4):376–384.

[86] Lavaei, A. (2019). *Automated Verification and Control of Large-Scale Stochastic Cyber-Physical Systems: Compositional Techniques*. PhD thesis, Department of Electrical Engineering, Technische Universität München, Germany.

[87] Lavaei, A., Nejati, A., Jagtap, P., and Zamani, M. (2021). Formal safety verification of unknown continuous-time systems: A data-driven approach. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, HSCC '21, New York, NY, USA. Association for Computing Machinery.

[88] Lavaei, A., Perez, M., Kazemi, M., Somenzi, F., Soudjani, S., Trivedi, A., and Zamani, M. (2022). Compositional reinforcement learning for discrete-time stochastic control systems. *arXiv preprint arXiv:2208.03485*.

[89] Lavaei, A., Somenzi, F., Soudjani, S., Trivedi, A., and Zamani, M. (2020a). Formal controller synthesis for continuous-space MDPs via model-free reinforcement learning. In *International Conference on Cyber-Physical Systems (ICCPS)*, pages 98–107.

[90] Lavaei, A., Soudjani, S., and Zamani, M. (2019). Compositional construction of infinite abstractions for networks of stochastic control systems. *Automatica*, 107:125–137.

[91] Lavaei, A., Soudjani, S., and Zamani, M. (2020b). Compositional (in)finite abstractions for large-scale interconnected stochastic systems. *IEEE Trans. on Automatic Control*, 65(12):5280–5295.

[92] Lazaric, A., Restelli, M., and Bonarini, A. (2008). Reinforcement learning in continuous action spaces through sequential Monte Carlo methods. In *Advances in neural information processing systems*, pages 833–840.

[93] Le Corronc, E., Girard, A., and Goessler, G. (2013). Mode sequences as symbolic states in abstractions of incrementally stable switched systems. In *Proceedings of the 52th IEEE Conference on Decision and Control*, pages 3225–3230.

[94] Legat, B., Jungers, R. M., and Bouchat, J. (2021). Abstraction-based branch and bound approach to q-learning for hybrid optimal control. In *Learning for Dynamics and Control*, pages 263–274. PMLR.

[95] Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17(1):1334–1373.

[96] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971.

[97] Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 157–163.

[98] Littman, M. L. and Szepesvari, C. (1996). A generalized reinforcement-learning model: Convergence and applications. In *International Conference on Machine Learning*, pages 310–318.

[99] Ma, M. and Fan, L. (2016). Implementing consensus based distributed control in power system toolbox. In *2016 North American Power Symposium (NAPS)*, pages 1–6.

[100] Mahadevan, S. (1996). Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine learning*, 22(1):159–195.

[101] Maitra, A. and Sudderth, W. (1993). Borel stochastic games with lim sup payoff. *The Annals of Probability*, 21(2):861–885.

[102] Majumdar, R., Mallik, K., Schmuck, A.-K., and Soudjani, S. (2021a). Symbolic control for stochastic systems via parity games. *arXiv:2101.00834*.

[103] Majumdar, R., Mallik, K., Schmuck, A.-K., and Soudjani, S. (2021b). Symbolic qualitative control for stochastic systems via finite parity games. *IFAC-PapersOnLine*, 54(5):127–132.

[104] Majumdar, R., Mallik, K., and Soudjani, S. (2020a). Symbolic controller synthesis for Büchi specifications on stochastic systems. In *Hybrid Systems: Computation and Control (HSCC)*, New York, NY, USA. ACM.

[105] Majumdar, R., Ozay, N., and Schmuck, A. (2020b). On abstraction-based controller design with output feedback. In Ames, A. D., Seshia, S. A., and Deshmukh, J., editors, *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020*, pages 15:1–15:11. ACM.

[106] Makdesi, A., Girard, A., and Fribourg, L. (2021). Efficient data-driven abstraction of monotone systems with disturbances. In *7th IFAC Conference on Analysis and Design of Hybrid Systems, ADHS 2021, Brussels, Belgium, July 7-9, 2021*, volume 54 of *IFAC-PapersOnLine*, pages 49–54. Elsevier.

[107] Mallik, K., Soudjani, S., Schmuck, A.-K., and Majumdar, R. (2017). Compositional construction of finite state abstractions for stochastic control systems. In *Conference on Decision and Control (CDC)*, pages 550–557. IEEE.

[108] Meyer, P.-J., Devonport, A., and Arcak, M. (2021). Abstraction-based control synthesis. In *Interval Reachability Analysis*, pages 93–101. Springer.

[109] Meyer, P. J., Girard, A., and Witrant, E. (2017, accepted). Compositional abstraction and safety synthesis using overlapping symbolic models. *IEEE Transactions on Automatic Control*.

[110] Mitsioni, I., Tajvar, P., Kragic, D., Tumova, J., and Pek, C. (2021). Safe data-driven contact-rich manipulation. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 120–127. IEEE.

[111] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, volume 48, pages 1928–1937.

[112] Mnih, V. et al. (2015). Human-level control through reinforcement learning. *Nature*, 518:529–533.

[113] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[114] Mohajerin Esfahani, P., Sutter, T., and Lygeros, J. (2015). Performance bounds for the scenario approach and an extension to a class of non-convex programs. *IEEE Transactions on Automatic Control*, 60(1):46–58.

[115] Naik, A., Shariff, R., Yasui, N., and Sutton, R. (2019). Discounted reinforcement learning is not an optimization problem. *ArXiv*, abs/1910.02140.

[116] Neider, D., Gaglione, J.-R., Gavran, I., Topcu, U., Wu, B., and Xu, Z. (2021). Advice-guided reinforcement learning in a non-Markovian environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[117] Ng, A. Y., Harada, D., and Russell, S. (1999a). Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287.

[118] Ng, A. Y., Harada, D., and Russell, S. J. (1999b). Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, pages 278–287.

[119] Oura, R., Sakakibara, A., and Ushio, T. (2020). Reinforcement learning of control policy for linear temporal logic specifications using limit-deterministic Büchi automata. *IEEE Control Systems Letters*, 4(3):761–766.

[120] Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450.

[121] Piche, S. W. (1994). Steepest descent algorithms for neural network controllers and filters. *IEEE Transactions on Neural Networks*, 5(2):198–212.

[122] Pnueli, A. (1977). The temporal logic of programs. In *Proc. 18th Symposium on Foundations of Computer Science*, pages 46–57. IEEE.

[123] Prajna, S., Jadbabaie, A., and Pappas, G. J. (2007). A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428.

[124] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[125] Ramadge, P. J. and Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230.

[126] Recht, B. (2018). A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, pages 253–279.

[127] Reissig, G., Weber, A., and Rungger, M. (2016). Feedback refinement relations for the synthesis of symbolic controllers. *IEEE TAC*, 62(4):1781–1796.

[128] Riedmiller, M. (2005). Neural fitted Q iteration – First experiences with a data efficient neural reinforcement learning method. In *Machine Learning: ECML 2005*, pages 317–328. Springer.

[129] Rungger, M. and Zamani, M. (2016). Scots: A tool for the synthesis of symbolic controllers. In *Proceedings of the 19th international conference on hybrid systems: Computation and control*, pages 99–104.

[130] Sadigh, D., Kim, E. S., Coogan, S., Sastry, S. S., and Seshia, S. A. (2014). A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. In *Conference on Decision and Control*, pages 1091–1096.

[131] Sadraddini, S. and Belta, C. (2018). Formal guarantees in data-driven model identification and control synthesis. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC 2018, Porto, Portugal, April 11-13, 2018*, pages 147–156. ACM.

[132] Salamati, A., Lavaei, A., Soudjani, S., and Zamani, M. (2021). Data-driven verification and synthesis of stochastic systems through barrier certificates. *arXiv preprint arXiv:2111.10330*.

[133] Samuel, S., Mallik, K., Schmuck, A., and Neider, D. (2020). Resilient abstraction-based controller design. In *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020*, pages 33:1–33:2. ACM.

[134] Schrijver, A. et al. (2003). *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer.

[135] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

[136] Scott, D. W. (1992). *Multivariate Density Estimation. Theory, Practice, and Visualization*. Wiley.

[137] Sickert, S., Esparza, J., Jaax, S., and Křetínskỳ, J. (2016). Limit-deterministic Büchi automata for linear temporal logic. In *International Conference on Computer Aided Verification (CAV)*, pages 312–332. Springer.

[138] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

[139] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

[140] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676):354.

[141] Singh, R., Gupta, A., and Shroff, N. B. (2020). Learning in Markov decision processes under constraints. *arXiv preprint arXiv:2002.12435*.

[142] Sistla, A. P. (1994). Safety, liveness and fairness in temporal logic. *Formal Aspects in Computing*, 6:495–511.

[143] Soudjani, S. (2014). *Formal Abstractions for Automated Verification and Synthesis of Stochastic Systems*. PhD thesis, Technische Universiteit Delft, The Netherlands.

[144] Soudjani, S. and Abate, A. (2011). Adaptive gridding for abstraction and verification of stochastic hybrid systems. In *Proceedings of the 8th International Conference on Quantitative Evaluation of Systems*, pages 59–69.

[145] Soudjani, S. and Abate, A. (2012a). Higher-Order Approximations for Verification of Stochastic Hybrid Systems. In Chakraborty, S. and Mukund, M., editors, *Automated Technology for Verification and Analysis*, volume 7561 of *Lecture Notes in Computer Science*, pages 416–434. Springer Verlag, Berlin Heidelberg.

[146] Soudjani, S. and Abate, A. (2012b). Probabilistic invariance of mixed deterministic-stochastic dynamical systems. In *ACM Proceedings of the 15th International Conference on Hybrid Systems: Computation and Control*, pages 207–216, Beijing, PRC.

[147] Soudjani, S. and Abate, A. (2013a). Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. *SIAM Journal on Applied Dynamical Systems*, 12(2):921–956.

[148] Soudjani, S. and Abate, A. (2013b). Aggregation of thermostatically controlled loads by formal abstractions. In *European Control Conference*, pages 4232–4237, Zurich, Switzerland.

[149] Soudjani, S. and Abate, A. (2014a). Precise approximations of the probability distribution of a Markov process in time: an application to probabilistic invariance. In *TACAS'14*, volume 8413 of *Lecture Notes in Computer Science*, pages 547–561. Springer.

[150] Soudjani, S. and Abate, A. (2014b). Probabilistic reach-avoid computation for partially-degenerate stochastic processes. *IEEE Transactions on Automatic Control*, 59(2):528–534.

[151] Soudjani, S. and Abate, A. (2015a). Aggregation and control of populations of thermostatically controlled loads by formal abstractions. *IEEE Transactions on Control Systems Technology*, 23(3):975–990.

[152] Soudjani, S. and Abate, A. (2015b). Quantitative approximation of the probability distribution of a markov process by formal abstractions. *Logical Methods in Computer Science*, 11(3):1–29.

[153] Soudjani, S., Abate, A., and Majumdar, R. (2015). Dynamic Bayesian networks as formal abstractions of structured stochastic processes. In *Proceedings of the 26th International Conference on Concurrency Theory*, pages 1–14.

[154] Soudjani, S., Abate, A., and Majumdar, R. (2017). Dynamic Bayesian networks for formal verification of structured stochastic processes. *Acta Informatica*, 54(2):217–242.

[155] Soudjani, S., Adzkiya, D., and Abate, A. (2016). Formal verification of stochastic max-plus-linear systems. *IEEE Transactions on Automatic Control*, 61(10):2861–2876.

[156] Soudjani, S. and Majumdar, R. (2018). Concentration of measure for chance-constrained optimization. *IFAC-PapersOnLine*, 51(16):277–282.

[157] Strehl, A. L., Li, L., Wiewiora, E., Langford, J., and Littman, M. L. (2006). Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888.

[158] Sun, D., Jha, S., and Fan, C. (2020). Learning certified control using contraction metric. *arXiv preprint arXiv:2011.12569*.

[159] Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44.

[160] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[161] Tabuada, P. (2009). *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer Publishing Company, Incorporated, 1st edition.

[162] Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Commun. ACM*, 38(3):58–68.

[163] Tkachev, I. and Abate, A. (2011). On infinite-horizon probabilistic properties and stochastic bisimulation functions. In *Proceedings of the 50th IEEE Conference on Decision and Control*, pages 526–531.

[164] Tkachev, I., Mereacre, A., Katoen, J., and Abate, A. (2013). Quantitative automata-based controller synthesis for non-autonomous stochastic hybrid systems. In *Hybrid Systems: Computation and Control*, pages 293–302. ACM.

[165] Tkachev, I., Mereacre, A., Katoen, J.-P., and Abate, A. (2017). Quantitative model-checking of controlled discrete-time Markov processes. *Information and Computation*, 253:1–35.

[166] Toro Icarte, R., Waldie, E., Klassen, T., Valenzano, R., Castro, M., and McIlraith, S. (2019). Learning reward machines for partially observable reinforcement learning. *Advances in Neural Information Processing Systems*, 32:15523–15534.

[167] Velasquez, A., Bissey, B., Barak, L., Beckus, A., Alkhouri, I., Melcer, D., and Atia, G. (2021). Dynamic automaton-guided reward shaping for monte carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35(13), pages 12015–12023.

[168] Verdier, C. F., Kochdumper, N., Althoff, M., and Mazo Jr, M. (2020). Formal synthesis of closed-form sampled-data controllers for nonlinear continuous-time systems under stl specifications. *arXiv preprint arXiv:2006.04260*.

[169] Wan, Y., Naik, A., and Sutton, R. S. (2021). Learning and planning in average-reward Markov decision processes. In *International Conference on Machine Learning*, pages 10653–10662. PMLR.

[170] Wang, X., Nair, S., and Althoff, M. (2020). Falsification-based robust adversarial reinforcement learning. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 205–212. IEEE.

[171] Watanabe, K., Renninger, N., Sankaranarayanan, S., and Lahijanian, M. (2021). Probabilistic specification learning for planning with safety constraints. In *Intelligent Robots and Systems (IROS)*, page TBA. IEEE.

[172] Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, King's College, Cambridge.

[173] Wei, C.-Y., Jahromi, M. J., Luo, H., Sharma, H., and Jain, R. (2020). Model-free reinforcement learning in infinite-horizon average-reward Markov decision processes. In *International conference on machine learning*, pages 10170–10180. PMLR.

[174] Weng, T.-W., Zhang, H., Chen, P.-Y., Yi, J., Su, D., Gao, Y., Hsieh, C.-J., and Daniel, L. (2018). Evaluating the robustness of neural networks: An extreme value theory approach. In *International Conference on Learning Representations*.

[175] Wood, G. and Zhang, B. (1996). Estimation of the lipschitz constant of a function. *Journal of Global Optimization*, 8(1):91–103.

[176] Wooding, B., Vahidinasab, V., and Soudjani, S. (2020). Formal controller synthesis for frequency regulation utilising electric vehicles. In *2020 International Conference on Smart Energy Systems and Technologies (SEST)*, pages 1–6. IEEE.

[177] Xu, Z., Gavran, I., Ahmad, Y., Majumdar, R., Neider, D., Topcu, U., and Wu, B. (2020). Joint inference of reward machines and policies for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 590–598.

[178] Xu, Z., Wu, B., Ojha, A., Neider, D., and Topcu, U. (2021). Active finite reward automaton inference and reinforcement learning using queries and counterexamples. In *Machine Learning and Knowledge Extraction - 5th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2021, Virtual Event, August 17-20, 2021, Proceedings*, volume 12844 of *Lecture Notes in Computer Science*, pages 115–135. Springer.

[179] Xue, B., Zhang, M., Easwaran, A., and Li, Q. (2020). Pac model checking of black-box continuous-time dynamical systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3944–3955.

[180] Yoon, H., Chou, Y., Chen, X., Frew, E., and Sankaranarayanan, S. (2019). Predictive runtime monitoring for linear stochastic systems and applications to geofence enforcement for UAVs. In *International Conference on Runtime Verification*. Springer.

[181] Yuan, L. Z., Hasanbeig, M., Abate, A., and Kröning, D. (2019). Modular deep reinforcement learning with temporal logic specifications. *arXiv preprint arXiv:1909.11591*.