Automatic Control Synthesis with Temporal Logic Requirements

Automatic Control Synthesis with Temporal Logic Requirements

Stochastic, Uncertain, and Nonlinear Systems

Birgit van Huijgevoort

Automatic Control Synthesis with Temporal Logic Requirements

Stochastic, Uncertain, and Nonlinear Systems

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus prof.dr. S.K. Lenaerts, voor een commissie aangewezen door het College voor Promoties, in het openbaar te verdedigen op

vrijdag 10 november 2023 om 13:30 uur

door

Birgit Charlotte van Huijgevoort

geboren te Breda

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

Voorzitter:	prof. dr. ir. M.J. Bentum
Promotor:	prof. dr. S. Weiland
Co-promotor:	dr. ir. S. Haesaert
Leden:	prof. dr. E. Ábrahám (RWTH Aachen University)
	prof. dr. D.V. Dimarogonas (KTH Royal Institute of Technology)
	dr. ir. M.A. Reniers
Adviseur:	dr. S.E.Z. Soudjani (Newcastle University UK, and Max Planck institute for software systems)

Het onderzoek dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

This dissertation has been completed in fulfillment of the requirements of the Dutch Institute of Systems and Control (DISC) for graduate study.

A catalogue record is available from the Eindhoven University of Technology Library.

ISBN: 978-90-386-5864-3

Cover design: Birgit van Huijgevoort & Rob Sanders.

Print: ADC-Dereumaux.

Summary

Automatic Control Synthesis with Temporal Logic Requirements

Stochastic, Uncertain, and Nonlinear Systems

The role of technology in everyday life is increasing rapidly, and so is the importance of reliable and safe behavior of high-tech systems. This is especially the case for safety-critical systems, where malfunctions have disastrous consequences. Even though it is not always possible to completely avoid malfunctions, for example, due to uncertain influences, it is crucial to minimize the likelihood of these events. In this thesis, I look into improving the reliability of safety-critical systems by giving guarantees on their functional behavior. To this end, I explore the area of *automatic, correct-by-design control synthesis* for systems evolving over a continuous state space and develop formal methods to synthesize provably correct controllers subject to temporal logic requirements.

Multiple uncertain factors, such as stochastic influences on the dynamics, play a huge role in the behavior of safety-critical systems. When ignoring these uncertain influences, it is impossible to achieve guarantees that actually lead to reliable and safe behavior. That is why it is crucial to manage those uncertain factors and to obtain probabilistic guarantees on the behavior of safety-critical systems.

To manage the uncertainty in automatic control design, I develop a model-based formal method and `MATLAB` tool for *stochastic* systems, where I exploit and analyze the trade-off between deviation bounds on the output and probabilistic transitions through an *approximate simulation relation*. This method enables an efficient computation of the *similarity quantification* and increases the accuracy of the robust satisfaction probability. Besides that, I improve the computational efficiency and scalability of *correct-by-design* control synthesis by introducing multiple precision layers, which include a trade-off between computation time and accuracy of the results. Furthermore, we[1] extend this framework to uncertain models by including parametric uncertainty in the stochastic model obtained by *measurement data*. I exploit the usage of data even further by giving a glimpse of direct data-driven control with respect to functional guarantees in which the explicit modeling step is replaced by a data-driven characterization of the system.

[1] this work is performed together with a colleague from Newcastle University, UK.

Contents

1 Introduction

For the design and implementation of controllers for complex and safety-critical systems, it is crucial to obtain guarantees on their high-level behavior. To this end, we incorporate formal methods into the control design, which allows us to automatically synthesize a provably correct controller that minimizes the likelihood of malfunctions and gives guarantees on the behavior. When considering practical applications of these methods, we must handle multiple sources of uncertainties, such as stochastic influences on the systems dynamics and model uncertainties. In this chapter, we introduce the relevant background, motivation, and research subquestions related to preserving the functionality guarantees under uncertain circumstances, and give an overview of the main contributions.

1.1 Formal methods for control design

Technology and automation are everywhere and high-tech systems are getting more intertwined with our lives every day. Also, the controllers steering these systems are getting more intelligent and are continuously taking more decisions on their own (Pannu 2015). Due to this increase in autonomy of systems that interact closely with humans, reliability is a necessity (Liu et al. 2019). However, technology is not advanced enough to guarantee fully reliable behavior of such high-tech systems (Banerjee et al. 2018; Schwarting et al. 2018). This is especially the case for engineering systems that represent complex physical phenomena that cannot be captured by a (finite) deterministic system. To alleviate the lack of guarantees, we often use additional safety measures that reduce the impact of malfunctions. For example, car manufacturing robots are operating within a safety cage to contain the damage when a malfunction occurs. But also, for autonomous vehicles, we have safety measures, since it is still necessary for a driver to be present and touch the steering wheel from time to time (Pearl 2018). Similarly, we have advanced

autopilot systems in airplanes, but in case of malfunctions or emergencies, we still rely on a pilot in the cockpit. In (Zugaj and Narkiewicz 2009, p. 78), it is stated that "*they* (autopilots) *may not be able to react sufficiently efficient when an unpredicted malfunction appears*". These are clear examples that we do not trust reliable and safe operation of these systems enough (Sheng et al. 2019) and we cannot completely *rely* on their correct behavior. To push the technological advancements toward fully autonomous high-tech systems, we need a different level of reliability. This cannot be achieved by improving the existing approaches used in industry but requires a completely different approach often starting at the design process of these systems. More specifically, the current approach to maximize reliability of operation relies completely on testing, while this does not give us the guarantees we desire and need in the future.

Realistically, we cannot always avoid malfunctions and fully guarantee correct behavior, but we can minimize their likelihood. This is especially the case for *safety-critical systems*, where malfunctions can have disastrous consequences. Consider for example the domain of aviation or autonomous driving, where a malfunction can lead to people dying. Another example is the oil industry, where malfunctions lead to the rupture of oil or gas pipes. Such a malfunction does not only affect people's lives but also leads to huge losses for the environment and economy. Since the safety and reliability of these systems are very important, they are called *safety-critical* (Rausand 2014). This is the opposite for systems where other safety measures already reduce the impact of control malfunctions, e.g., a safety cage for car manufacturing robots. For systems that are not safety-critical, there is a trade-off between the cost of a malfunction and control design cost. For safety-critical systems, such a trade-off does not exist since safety and reliability are the most important aspects. Hence, *for safety-critical systems, it is crucial to minimize the likelihood of malfunctions and it is important to obtain guarantees on their behavior.* Motivated by the importance of reliable behavior for such systems, we formulate the global driver of this research as follows.

Research Objective

Improve reliability of safety-critical systems by giving guarantees on their behavior.

The behavior of high-tech systems is usually controlled by one or more controllers. The area of control theory is often focused on stability and performance guarantees, while we want to achieve *functional guarantees* on *high-level behavior*. As an example, consider an autonomous driving vehicle, where stability guarantees are necessary to avoid skidding when turning aggressively (Lima et al. 2018), and performance guarantees are used in lane keeping (Zhang et al. 2013) or to minimize energy consumption. We focus on the functional guarantees of high-level behavior, such as driving from A to B while avoiding static and dynamic obstacles. Such complex functional requirements can be formulated in a mathematical way using *temporal logic*.

The general procedure of designing a controller is illustrated in Figure 1.1a and is often referred to as the *V-model* (Skjetne and Egeland 2006; Sarhadi and Yousefpour 2015). This design procedure is iterative and is mainly relying on testing a system repeatedly. According to a study on the reliability of autonomous driving (Kalra and Paddock 2016), it takes 275 million failure-free miles to achieve accurate behavior with 95% confidence on reliability. This would imply driving 24 hours a day for 12.5 years with 100 test vehicles. Obviously, this a very expensive and time-consuming procedure, that does not even give us actual guarantees on safe behavior. This is because testing can only show the presence of errors, not their absence (Dijkstra et al. 1970, Corollary on page 6).

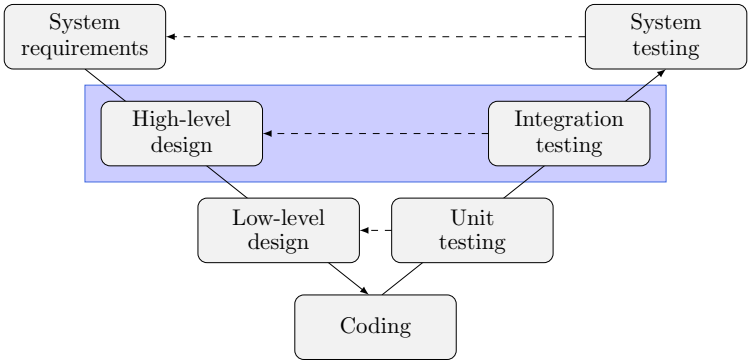
To obtain guarantees on safe behavior, we want to prove the absence of errors, which would imply considering all possible behaviors of a system. This is impossible by only testing the system's behavior through a finite number of testing scenarios. Besides that, when a test fails it is difficult to determine the exact reason it fails.

Luckily there exists a way to alleviate these issues, namely using so-called *formal methods* (Kreiker et al. 2011), that are originally applied for verification (also known as model checking) by the computer science community (Woodcock et al. 2009). Currently, formal methods are also developed to automatically design provably correct controllers (Tabuada 2009; Belta et al. 2017). Such methods can be used as a complement to the ordinary high-level design step as illustrated in Figure 1.1b. This formal procedure is known as *temporal logic control, correct-by-design control synthesis* or *automated control synthesis* and allows us to formally prove the absence of errors and to give formal guarantees on the functional behavior of systems. Besides that, this reduces the required testing and the number of testing scenarios substantially. Hence, we conclude that *it is crucial to incorporate formal methods in control design of safety-critical systems*, since this allows us to achieve reliable functioning by giving guarantees on controlled systems behavior.

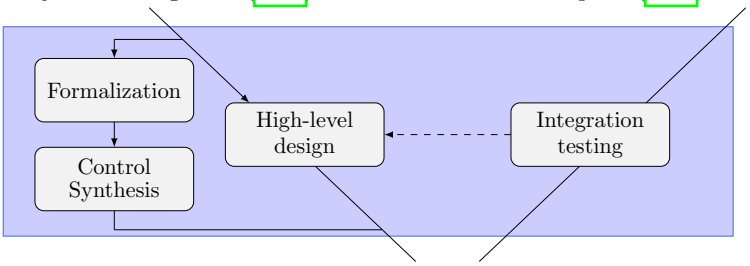
Guarantees can be provided in either a quantitative or qualitative setting, which respectively means either guaranteeing that desired behavior is satisfied or violated, or giving a probability of violating desired behavior. In this thesis, we are interested in the latter, since in practice, it is not possible to completely avoid malfunctions due to uncertainties coming from very diverse origins.

For example, when driving on an icy road or flying through a thunderstorm. In those cases, the dynamics of the system are influenced by certain *uncertain* factors. Such factors include the weather, but also people. A bus filled with people and luggage to (or even over) its capacity behaves differently than an empty bus. Also, power surges in a power grid network can occur when a lot of people require a lot of energy at the same time. Since accidents due to for example extreme weather conditions are unavoidable, we cannot evaluate the behavior over all possible weather conditions uniformly. Instead, these uncertain influences on the dynamics of a system can be quantified by a probability distribution.

It should therefore not be surprising that the uncertainty in the behavior of safety-critical systems is best described by stochastic models. Designing controllers for stochastic models using formal methods leads to many challenges. In this case, the control synthesis objective becomes a probabilistic synthesis problem, and the



(a) Simplified representation of the V-model to describe the design procedure of control systems. Detailed versions can be found in, e.g., Skjetne and Egeland (2006) and Sarhadi and Yousefpour (2015).



(b) Correct-by-design control synthesis as a complement to the ordinary high-level design phase.

Figure 1.1: General procedure of control design in **(a)**, with correct-by-design control synthesis complementing the high-level design step in **(b)**.

probability that the controlled system satisfies the desired behavior, typically cannot be computed analytically (Abate et al. 2008). Hence, we have to either approximate this probability or compute an accurate lower bound on the probability to meet the specification of a controlled system. By doing so, we can still obtain guarantees on the system's behavior by expressing the probability of occurrence of malfunctions. For many safety-critical systems, ignoring these influences or considering them bounded makes it impossible to obtain realistic guarantees.

1.2 Challenges

Multiple challenges arise when developing formal methods for the control design of dynamical systems. Formal methods are techniques that use rigorously specified *mathematical models* to formally prove correct behavior, hence giving *formal guarantees*. We refer to control synthesis with formal guarantees as *provably correct control synthesis*. As shown in Figure 1.2, a model-based control design can be split up into two steps; a modeling step and a control synthesis step.

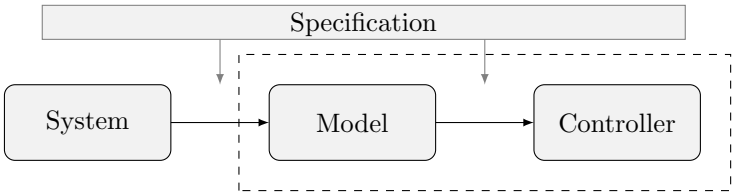


Figure 1.2: Graphical representation of model-based control design with two steps; a modeling step and a control synthesis step (indicated by the dashed box).

The challenges associated with these steps are *model uncertainty*, *model complexity*, and *complexity of the specification*. These properties can be split up even further as described next. First, we focus on the control synthesis step, as indicated by the dashed box in Figure 1.2.

Model complexity. Assume that we have obtained a model of the system. The model characteristics are inherited from the system, e.g., the behavior of a deterministic system is described by a deterministic model, while the behavior of a stochastic system is described by a stochastic model. Such *model characteristics* can lead to *complex* models, which pose challenges for the provably correct control design of dynamical systems.

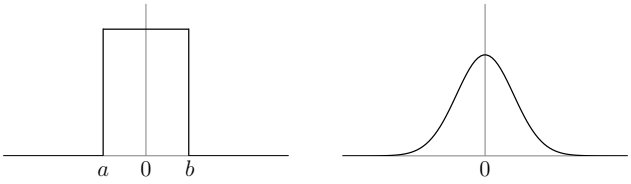
The behavior of the model is described either in continuous-time, discrete-time, or a combination thereof, i.e., as a hybrid model. This leads to models that can also be continuous-time, discrete-time, or hybrid. However, it should be noted that through time-discretization (Gottlieb and Ketcheson 2016) continuous-time systems can be modeled in discrete-time, and this works well in practice since controllers are digital devices and, therefore, run in discrete-time.

Similarly, we can distinguish between models with a state space that is discrete, continuous[1] or hybrid. Furthermore, the state evolution of the model (and system) can be deterministic or nondeterministic. If we associate probabilities with the nondeterminism in state evolutions, we refer to probabilistic or stochastic systems. The latter is more commonly used within the control community.

Model complexity is a broad term that can be associated with the model characteristics as described before, with the state dimension of the system (continuous state space), the number of states of the system (discrete state space), a quantification of the level or degree of nonlinearity, the dimension of the stochastic influence and the size of the (possibly unbounded) support of its probability distribution (see Figure 1.3), and so on.

Complexity of the specification. As mentioned before, we consider specifications written using *temporal logic* (Pnueli 1977), which can describe many different properties such as safety (bad events do not happen), liveness (good events do happen) and reachability (some situations can eventually be reached). Using temporal and logical operators, we can combine properties and formulate more *complex* specifications.

[1] In the Computer Science community, they often refer to state spaces that are countable or uncountable instead of respectively discrete or continuous.



(a) Uniform distribution with bounded support $[a, b]$.

(b) Gaussian distribution with unbounded support $[-\infty, \infty]$.

Figure 1.3: Different probability density functions (Yates and Goodman 1999, Chapter 3) with a bounded support in **(a)**, and an unbounded support in **(b)**. The y-axes are scaled the same.

A common language to describe specifications is Linear Temporal Logic (LTL). We give a glimpse into LTL specifications next, but for a complete understanding we refer the interested reader to the following books on formal control synthesis subject to temporal logic specifications: Belta et al. (2007) and Tabuada (2008).

LTL is given over a set of *atomic propositions* $p \in \text{AP}$ that each can be either true or false. For example, if the system is inside region P_1, atomic proposition p_1 is true, otherwise it is false. The syntax of LTL is recursively defined over a set of *atomic propositions* $p \in \text{AP}$ as

$$\phi ::= \top \mid p \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc\phi \mid \phi_1 \,\mathsf{U}\, \phi_2, \tag{1.1}$$

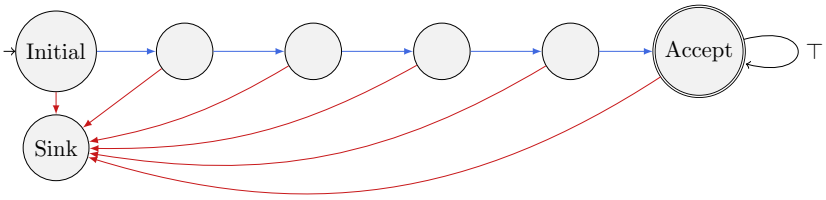
where ϕ, ϕ_1, and ϕ_2 are all LTL specifications, and the different syntax rules are split up by the vertical lines $|$. The term *recursively defined* indicates that the specification can be built recursively from these syntax rules. The semantics are as follows. The $\top$ stands for *true* and is unconditionally true. Specification $\phi = p$ is satisfied if atomic proposition p is true. Specification $\phi = \phi_1 \wedge \phi_2$ is satisfied if both ϕ_1 and ϕ_2 are true. Specification $\neg\phi$ is satisfied if ϕ is false. Specification $\bigcirc\phi$ is satisfied if ϕ is true at the next instance. Specification $\phi_1 \,\mathsf{U}\, \phi_2$ is satisfied if ϕ_1 is true until ϕ_2 is true. Additional operators can be built from operators in the minimal syntax in (1.1), namely,

- The or-operator, $\phi_1 \vee \phi_2 := \neg(\neg\phi_1 \wedge \neg\phi_2)$, which holds if either ϕ_1 or ϕ_2 is true.
- The implication, $\phi_1 \implies \phi_2 := \neg\phi_1 \vee \phi_2$, which holds if ϕ_1 equal to true implies that ϕ_2 is also equal to true.
- The eventually-operator, $\Diamond\phi := \top \,\mathsf{U}\, \phi$, which holds if ϕ is true at some time instance in the future.
- The always-operator, $\Box\phi := \neg\Diamond\neg\phi$, which holds if ϕ is true at all time instances.

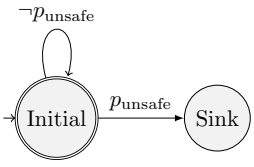
An LTL specification can equivalently be given as an automaton, which also gives an indication of the complexity of the specification. For example, a safety specification $\phi_{\text{safety}} = \Box\neg p_{\text{unsafe}}$, with p_{unsafe} the atomic proposition associated to an unsafe set of

outputs, can equivalently be given by the automaton in Figure 1.4b. This automaton starts in the state Initial and evolves depending on which atomic proposition is true. In this case, the initial state is also *accepting* indicated by the double line around it. When p_{unsafe} is false it follows the self-loop, otherwise, it goes to the sink state. The specification is satisfied if the accepting state is visited *infinitely often*.

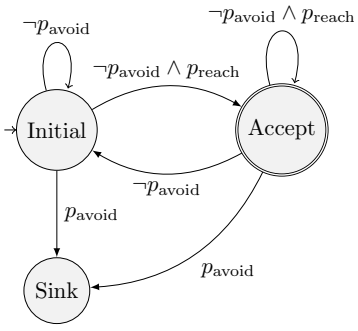
Some specifications are easier to solve than others. Even though it is difficult to quantify the complexity of a specification precisely, we provide some insight into this. In Figure 1.4, we show multiple automatons. Safety specifications $\phi_{\text{safetyF}} = \bigwedge_{i=0}^{5} \bigcirc^i \neg p_{\text{unsafe}}$ and ϕ_{safety} with respectively a finite and infinite time-horizon are respectively shown in (a) and (b). A reach-avoid specification $\phi_{RA} = \square(\neg p_{\text{avoid}} \wedge \lozenge p_{\text{reach}})$ is shown in (c). Those specifications are in general rather easy to satisfy, while the one in (d) is more complex. However, the complexity also depends on the regions themselves, e.g., a safety specification with small safe regions is more complex than one with a large safe region. Similarly, reach-avoid specification ϕ_{RA} becomes more complicated if the avoid region (atomic proposition p_{avoid}), is close to the goal region (atomic proposition p_{reach}). A major impact on the complexity is the time-horizon of the specification, where we distinguish between specifications over a *finite* or *infinite horizon*, of which examples are shown in Figures 1.4b and 1.4a.



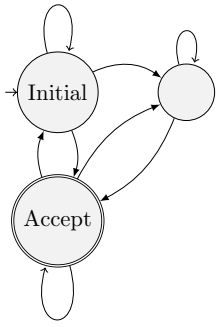
(a) Safety specification with a finite time-horizon. Blue and red transitions are respectively labeled with $\neg p_{\text{unsafe}}$ and p_{unsafe}.



(b) Safety specification.



(c) Reach-avoid specification.



(d) More complex specification, with omitted labels.

Figure 1.4: Automatons corresponding to different types of specifications with increasing complexity.

Model uncertainty. With *model uncertainty* we refer to the uncertainty encountered when modeling a system (first step in Figure 1.2), indicating that there is most likely a mismatch between the system and its model. It is a broad concept and can be split up as follows. On the one hand, we can have parametric uncertainty with respect to the dynamics or the noise distribution (for stochastic systems). On the other hand, we can have uncertainty with respect to the structure of the model itself. It is important to understand that model uncertainty restricts the application of formal methods for the control design of realistic systems. More specifically, uncertainty will always have an effect on reliability in the sense that increasing uncertainty will decrease reliability. Luckily, we can obtain information about the system about either its states, its inputs, or its (noisy) outputs, which helps to reduce model uncertainty.

Next, we elaborate more on the challenges in this thesis and give a brief overview of what has been done already and what are still open challenges.

1.2a Challenges due to model complexity

To understand the challenges due to model complexity, some background on the origin of formal methods is required. Formal methods are initially developed for reliability of software programs (Hinchey et al. 2008). For example, to avoid deadlocks, but also for checking completeness, traceability, verifiability, and reusability (Ghose 2000). These methods are applied in various engineering domains, such as process control, population census, railway signaling, air traffic control, telecommunications, and radiotherapy (Bowen and Hinchey 2005; Hinchey and Bowen 2012). Furthermore, formal methods are also used to automatically generate code (Abrial 1996; Berry 2008).

Since their application is originally focused on software programs, formal methods have been developed for systems with a *discrete state space*, evolving over *discrete-time*. Formal methods are extended to probabilistic systems using probabilistic model checking (Kwiatkowska et al. 2005; Kwiatkowska et al. 2007). Probabilistic systems define probabilities over nondeterministic state transitions and are in that sense similar to stochastic systems when considering their state evolutions. Similarly, results on control synthesis over quantitative and qualitative probabilistic properties exist for finite-state Markov decision processes[2] (Baier and Katoen 2008), however, their extension to continuous-state stochastic systems is very challenging. The fact that these methods have originally been limited to discrete state spaces is because a countable number of states allows considering all possible behaviors of a system, thus yielding formal guarantees on the behavior.

However, we want to obtain formal guarantees for engineering systems, which mainly are *dynamical systems* that evolve both over time and space in a continuous sense. Most engineering systems are best modeled by a combination of continuous evolutions and discrete transitions, leading to a hybrid state space. This continuous or hybrid behavior leads to many challenges with respect to verification (Schupp et al. 2015) and control synthesis (Belta et al. 2007).

[2] A Markov decision process can be used to describe the dynamics of a stochastic system.

Some of these challenges are already resolved for control synthesis of both discrete-time and continuous-time *deterministic* systems through for example *(approximate) simulation relations* (Girard and Pappas 2007; Tabuada 2008; Belta et al. 2017), *approximate simulation functions* (Girard and Pappas 2009) or *barrier certificates* (Bisoffi and Dimarogonas 2018).

The step towards *stochastic systems* over a continuous state space is very challenging and lacks effective computational methods that are both accurate and scalable to higher-dimensional models. This is especially the case when the stochastic influence has a distribution with an *unbounded* support (illustrated in Figure 1.3 and discussed in detail in (Yates and Goodman 1999, Chapter 3)), which we refer to as an *unbounded disturbance*. Many methods for stochastic systems are, therefore, restricted to bounded disturbances (Majumdar et al. 2020; Majumdar et al. 2021) or often provide trivial lower bounds for unbounded disturbances (Anand et al. 2021). These challenges prevent the application of approximate methods to realistic stochastic systems. It is necessary to overcome these challenges since the step toward stochastic systems is crucial in order to obtain reliable behavior of safety-critical systems.

1.2b Challenges due to the complexity of the specification

Not only the complexity of the model is an issue, but also the complexity of the specification. Many methods excel at handling a single specific property (Vinod et al. 2019; Santoyo et al. 2021). Developing methods for general specifications is becoming more interesting recently (Anand et al. 2021; Dutreix et al. 2022; Zhong et al. 2023b), but usually have other restrictions, e.g., with respect to the model structure or time horizon of the specification. Especially *infinite-horizon* properties are challenging when considering stochastic systems (Tkachev and Abate 2011; Tkachev et al. 2017) and most methods are restricted to specifications over finite horizon (Lavaei et al. 2017; Jagtap et al. 2020; Santoyo et al. 2021; Nejati et al. 2022). With the increasing importance of reliable behavior, the specifications of safety-critical systems are also becoming more complex. Hence, methods should be able to handle complex specifications in order to be applicable to realistic systems.

1.2c Challenges due to model uncertainty

Obtaining a model is in itself a challenging task, especially when the behavior of the system is complex. For many realistic systems, it is impossible to obtain a model that describes the behavior of the system with 100% accuracy. Since the guarantees obtained through formal methods only hold with respect to the considered mathematical model, we cannot disregard *model uncertainty* in order to obtain provably correct controllers for safety-critical systems.

This *model uncertainty*, can for example be uncertainty about the model's parameters or the noise distribution of a stochastic system, but also about the model structure itself. Similarly, when using data measurements of a system to identify an accurate

model, we must take the uncertainty of the sensors into account. In the area of control design, this is known as measurement noise.

Managing such uncertainties by the correct interpretation of measurement data is a well-known topic in control design. Multiple learning-based methods are currently investigated for automated control synthesis (Kapoor et al. 2020; Kazemi and Soudjani 2020; Kalagarla et al. 2021), however, methods that consider stochastic systems evolving over continuous spaces started to appear only recently (Kazemi et al. 2022; Lavaei et al. 2022b). Also, these uncertainties must be taken into account in order to give accurate formal guarantees on the reliable behavior of safety-critical systems.

1.3 Research questions

With respect to the challenges of applying formal methods to control design of safety-critical systems, we focus on taking the various uncertain influences into account. We use the term *uncertainty* to describe both stochastic dynamics and model uncertainty. Currently, formal methods cannot be used to synthesize controllers for realistic systems, since the available methods cannot handle their complexity yet. This is especially the case when considering stochastic systems, for which there are no efficient computational methods available that are both accurate and scalable to higher-dimensional models, and that can handle specifications that need to be valid over infinite-horizon time intervals. Besides that, there is a lack of coverage of model uncertainty in the literature, even though it is crucial to address the uncertainty when using a model to describe the behavior of the (stochastic) system. Therefore, we formulate the following research subquestion.

Research Question

How to manage uncertainty when automatically synthesizing controllers subject to temporal logic requirements?

Before, we mentioned multiple sources of uncertainties, namely the stochastic influence on system dynamics and model uncertainty. Model uncertainty is quite a broad concept, which consists of among others, parametric uncertainty, measurement noise, and unknown systems, i.e., unknown model structures. In this thesis, we consider parametric uncertainty and unknown systems, while ignoring measurement noise. Besides that, we focus on *discrete-time* systems over a continuous state space and describe their dynamics either through *(stochastic) difference equations* or *Markov Decision Processes*.

The research subquestion highlights the fact that we want to have *automatic* controller synthesis with functional guarantees. This is illustrated in Figure 1.5. The goal is to develop a synthesis algorithm (blue box) that takes as input a model and a specification. To accurately describe the behavior of a dynamical

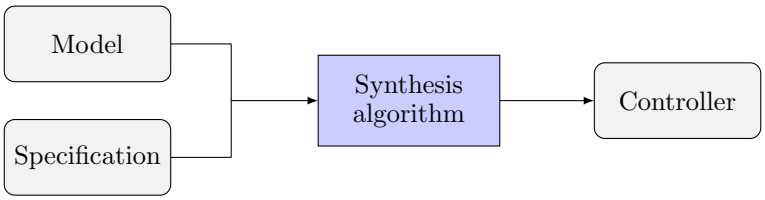


Figure 1.5: Automatic control synthesis algorithm that takes a mathematical model and specification as input, and outputs a controller that guarantees that the controlled system provably satisfies the specification.

system in a mathematical way we use a model with state evolutions in a *continuous state space*. The specification, on the other hand, describes the desired high-level behavior of the system. Hence, instead of focusing on low-level behavior, such as stability and disturbance rejection, we focus on high-level behavior. We refer to guarantees on high-level behavior as *functionality* guarantees. To describe this high-level behavior mathematically, we use *temporal logic*. Based on the mathematical description of both the system behavior (through a model) and the desired behavior (through a temporal logic specification), the synthesis algorithm should *automatically* design a controller, such that the desired behavior is provably satisfied by the controlled system. In this sense, *automatic control synthesis* implies that there is no handwritten, experimentally validated code present anymore.

To evaluate the research subquestion, we identify five subquestions. We start by considering the stochastic transitions of the model as the uncertain influence (subquestions 1-3). Next, we pose subquestions on how to handle model uncertainty and whether we can use data-driven control to tackle this additional uncertainty (subquestions 4 and 5). The subquestions are posed next and a more detailed explanation follows.

1) How to automatically synthesize the most reliable controller for a stochastic system influenced by an unbounded disturbance, such that it provably satisfies specifications over infinite-time horizons, and accurately quantify this reliability with a satisfaction probability?

2) How to develop efficient computational tools for solving subquestion 1?

3) How to improve the computational efficiency and scalability of provably correct control synthesis methods for stochastic systems, while maintaining accurate lower bounds on the satisfaction probability of their specifications?

4) After obtaining a stochastic model with explicit parametric uncertainty from data, how to synthesize provably correct controllers that are robust against parametric uncertainty?

5) Without using an explicit model, how to obtain guarantees on the behavior of unknown systems subject to temporal logic properties?

1.3a Control of stochastic models

We consider safety-critical systems described by stochastic difference equations and aim to design a provably correct controller, such that certain specifications are satisfied with a high probability. More specifically, we model the behavior of the system in discrete-time with an additive stochastic disturbance on the state dynamics as follows:

$$x_{t+1} = f(x_t, u_t) + w_t$$
$$y_t = h(x_t).$$

Here, x_t denotes the state at the current time step t, u_t the input and w_t is an additive disturbance coming from a probability distribution with an unbounded support (cf. Figure 1.3b). The deterministic part of the state update x_{t+1} is described by a function $f(x_t, u_t)$ based on the current state and input. Furthermore, y_t is the output, and $h(x_t)$ is the function mapping states to outputs. This class of discrete-time dynamics has a continuous state space, for which it is often not possible to compute the satisfaction probability analytically (Abate et al. 2008). However, there are two possible ways to solve this issue. We can either use sufficient conditions to analyze the continuous-state model directly or construct a finite-state approximation of the model, known as an abstraction (Lavaei et al. 2022a). The first is known as abstraction-free or discretization-free techniques, the latter are abstraction-based techniques. Both techniques have to deal with a trade-off between accuracy and efficiency.

Abstraction-free methods have been developed to suffer less from the curse of dimensionality, and are therefore, more scalable to models with a higher-dimensional state space. However, abstraction-free methods are more limited with respect to the structure of the model and the type of specifications they can handle. Furthermore, they have computational issues and often lack guarantees of finding a controller. More specifically, abstraction-free methods are either using certificates, or optimization approaches. The latter, including stochastic model predictive control (SMPC) (Mesbah 2016) do not provide formal guarantees and are often only applicable to stochastic reachability problems of linear systems (Lavaei et al. 2022a). Methods based on finding certificates, such as simulation functions (Lavaei et al. 2019) or (control) barrier certificates (Jagtap et al. 2020) have no general effective computational method and are limited with respect to the structure of the model or the specification (Lavaei et al. 2022a). These techniques either use sum-of-square (SOS) optimization (Papachristodoulou et al. 2013; Wongpiromsarn et al. 2015) or counter-example guided inductive synthesis (CEGIS) (Jagtap et al. 2020) to compute a certificate. The former is limited to specific *safety* specifications (Wongpiromsarn et al. 2015), while the iterative nature of the latter makes it difficult to analyze the computational complexity. Besides that, CEGIS does not provide completeness guarantees, which means that there is no guarantee that a certificate will be found, even though it exists (Jagtap et al. 2020; Lavaei et al. 2022a). A major shortcoming of barrier certificate methods is that they either require strict super-martingale conditions (Kushner 1967, Chapter I) that often do not exist (Steinhardt and Tedrake 2012; Jagtap et al. 2020) or use relaxed super-martingale

conditions, but are limited to specifications over a finite horizon (Nejati et al. 2020; Anand et al. 2022). Therefore, the application of these abstraction-free methods is limited to specific models and/or specifications.

Subquestion 1. Since abstraction-based methods can be applied to a wide range of models and specifications, we focus on abstraction-based methods in the first chapter of his thesis. The available literature on these methods is quite large (Zamani et al. 2014; Soudjani et al. 2015; Haesaert et al. 2017b; Cauchi and Abate 2019; Haesaert and Soudjani 2020; Lavaei et al. 2020a; Lavaei et al. 2022a), however, the present methods lack assessment of the accuracy of the computed satisfaction probability and computational efficiency of the controller synthesis. Furthermore, they are often limited to assumptions on a bounded stochastic influence of noise signals or to specifications over a finite horizon (Majumdar et al. 2020; Nejati et al. 2020; Majumdar et al. 2021; Anand et al. 2022). First, we focus on developing a computational method that is suitable for such systems and complex specifications. To this end, we formulate the following subquestion.

Subquestion 1

How to automatically synthesize the most reliable controller for a stochastic system influenced by an unbounded disturbance, such that it provably satisfies specifications over infinite-time horizons, and accurately quantify this reliability with a satisfaction probability?

Proposed methodology for Subquestion 1. As illustrated in Figure 1.6, the following steps are performed in abstraction-based control synthesis. After computing a finite-state approximation of the original, continuous-state model, the controller is synthesized based on the specification. This finite-state controller is refined back to the continuous space, after which it can be applied to the original model. To maintain the probabilistic guarantees, we have to quantify the similarity between the original and abstract models. This can be done through a *simulation relation*. The main conservatism of abstraction-based methods is caused by this similarity quantification. Furthermore, for this similarity quantification, the original and abstract models are coupled through their disturbances and inputs. After which, the similarity is quantified through deviation bounds. In Haesaert et al. (2017b), a simulation relation is introduced that computes two bounds, the output deviation or precision ϵ and probability deviation or confidence δ. This simulation relation together with the corresponding robust computation of the satisfaction probability from Haesaert and Soudjani (2020) can be applied to properties unbounded in time. However, an efficient approach to compute the deviation bounds has not been developed yet. Most simulation relation methods do not explicitly design the coupling between the model and its abstraction, and are, therefore, restricted with respect to the possible deviation bounds and simulation relations. Hence, it is of interest to explicitly design the coupling, such that the set of possible simulation relations can be derived. By introducing this additional freedom in the similarity quantification, the computed satisfaction probability is more accurate, and it allows

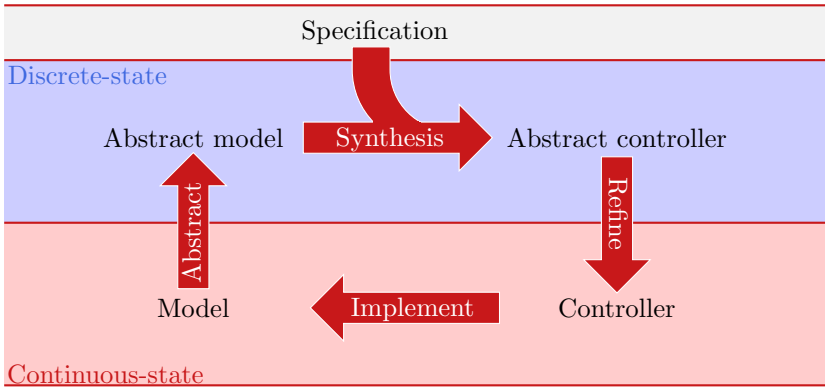


Figure 1.6: Schematic overview of the steps performed in abstraction-based control synthesis, with a continuous-state layer in red and a discrete-state layer in blue.

for efficient computation. A detailed formulation of this problem and the results can be found in *Chapter 2* entitled *Quantified abstraction for stochastic systems.*

Subquestion 2. Besides establishing the theory underlying the provably correct design of controllers, it is equally important to develop tools that facilitate their application. For stochastic systems, a collection of tools that can perform formal controller synthesis is already available. However, they are often not capable to handle the desired complexity of the specification and the stochastic influence on the dynamics. Therefore, we formulate the second subquestion as follows.

Subquestion 2

How to develop efficient computational tools for solving subquestion 1?

Proposed methodology for Subquestion 2. It is of interest to develop a tool that is applicable to properties that are unbounded in time and that can handle stochastic systems with a possibly unbounded disturbance. To this end, we develop a tool that employs the method developed in Chapter 2. Details about this `MATLAB` tool can be found in *Chapter 3*, entitled ***`SySCoRe`**: Synthesis via Stochastic Coupling Relations.*

Subquestion 3. The method introduced in Chapter 2 shows that when quantifying the similarity between a model and its finite-state abstraction, there is a trade-off between precision and confidence. Hence, we can either quantify the similarity with a high precision and a low confidence, or the other way around. However, in many cases, it is not always necessary to have a high precision over the whole state space. For such cases, it is of interest to develop a method with variable precision. Hence, we want to design a method where we can switch between multiple deviation bounds. This is expected to increase the total accuracy of the computed robust satisfaction probability.

Besides accuracy, it is also important to consider the computational efficiency and scalability of the approach. More specifically, we are interested in scalability to higher-dimensional (in terms of state space) systems, more complex specifications, and so on. These factors are limiting the applicability of correct-by-design control synthesis methods to realistic systems. Therefore, we pose the following subquestion.

Subquestion 3

How to improve the computational efficiency and scalability of provably correct control synthesis methods for stochastic systems, while maintaining accurate lower bounds on the satisfaction probability?

Proposed methodology for Subquestion 3. To this end, we exploit the variable precision even further, by using it to combine an abstraction-free method with an abstraction-based method. By doing so, we have the benefits of both methods. That is, we use an abstraction-free method that is generally very fast but has a low precision in the areas where possible. In the areas where we need to have high precision, we use abstraction-based methods. Since we only use this abstraction-based method on small areas, we suffer less from the curse of dimensionality than compared to using this method for the complete state space. The abstraction-free method guides the system towards high-precision areas such that the final specification is satisfied with overall high precision. Therefore, this method is expected to solve the given subquestion. Details and results are given in *Chapter 4* entitled *Specification-guided temporal logic control for stochastic systems: a multi-layered approach*.

1.3b Data-driven approaches

Next, we go beyond stochastic systems with a given model by expanding towards model uncertainty.

Subquestion 4. Normally, when designing a provably correct controller, it is designed based on one specific model with fixed parameter values. This means that your guarantees also only hold for this specific model. If parameters or parameter values are uncertain, we aim to obtain guarantees for all the possible parameters within the set, and, therefore, obtain robust guarantees on the functional behavior of the system. This is the main reason why we pose the fourth subquestion as follows.

Subquestion 4

After obtaining a stochastic model with explicit parametric uncertainty from data, how to synthesize provably correct controllers that are robust against parametric uncertainty?

Proposed methodology for Subquestion 4. More specifically, we are interested in automatically synthesizing a provably correct controller for stochastic systems by identifying a parametric model of the system with uncertainty incorporated in the parameter values. Hence, we consider a model with a constant, but unknown parameterization. This controller should be independent of the parameter, such that the obtained guarantees hold for all the models within the considered parameter set. In *Chapter 5* entitled *A Bayesian approach to temporal logic control of uncertain systems*, we follow a two-step methodology. First, we obtain data from the *unknown* system, which we use to compute a set that with a certain probability contains the (true) unknown parameter, and we obtain a parameterized set of models. Next, we consider the nominal model within this set and synthesize a provably correct controller for this model using an abstraction-based approach. Finally, we refine the obtained *abstract controller* to a controller for the parametric model, such that we can maintain the obtained guarantees on the uncertainty set. This method, therefore, relates the satisfaction of temporal requirements by the identified model to that of the original unknown stochastic model with respect to the size of the data set. The most crucial part of this method is the introduction of the concept of *sub-similarity relation*, which is inspired by the coupling compensator approach developed in Chapter 2.

Subquestion 5. In the following Chapter 6, we go even further by considering a direct data-driven method, that does not require any exact knowledge on the structure of the model. Most methods require an explicit modeling step, which is prone to errors that impact the guarantees on the behavior. Hence, by excluding this explicit modeling step, we circumvent this issue and obtain even stronger guarantees on the system's behavior. Hence, we introduce the last subquestion.

Subquestion 5

Without using an explicit model, how to obtain guarantees on the behavior of unknown systems subject to temporal logic properties?

Proposed methodology for Subquestion 5. In Chapter 6, entitled *Direct data-driven control with signal temporal logic specifications*, we make a first step towards such a direct data-driven method by considering *deterministic* systems and signal temporal logic. Furthermore, we restrict ourselves to properties bounded in time, while we envision extensions to *stochastic* systems and infinite-horizon properties. The method described in this chapter consists of two main steps. First, we use a data-driven method within the behavioral framework to obtain a characterization of the system using only a single input-output trajectory. Next, we write the signal temporal logic specification into mixed-integer linear programming constraints and formulate the control synthesis as an optimization problem.

1.4 Contributions and thesis outline

This thesis consists of two main parts; Part I: Control of stochastic models, and Part II: Data-driven approaches. Within these two parts, there are 5 main chapters. Each main chapter focuses on a specific research subquestion and can be read independently. However, to get a full theoretical understanding, readers are advised to start with Chapter 2 before reading Chapters 3-5. As illustrated in Figure 1.7, Chapter 3 discusses the implementation of Chapter 2 through a `MATLAB` toolbox. Chapters 4 and 5 are extensions of Chapter 2 and contain multiple references to this chapter. Chapter 6 can be read completely separately. A concise list of the symbols and abbreviations used in this thesis is given at the end.

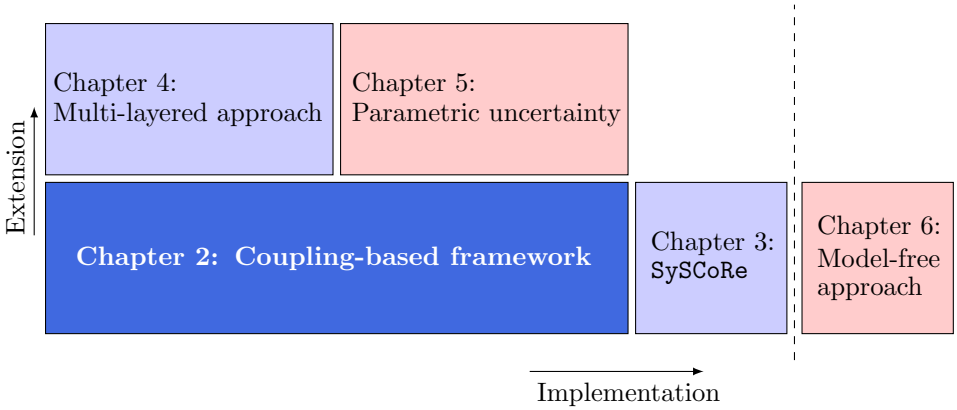


Figure 1.7: Graphical overview of the outline of this thesis, with the blue chapters belonging to Part I and the red chapters belonging to Part II.

In Part I, which is Chapters 2-4, we consider the uncertainty only being the stochastic influence on the dynamics and focus on computational challenges, such as accuracy and scalability to higher-dimensional models, as detailed by subquestions 1-3. In Part II, which is Chapters 5 and 6, we consider data-driven methods to handle model uncertainty. In Chapter 5, we focus on stochastic models with additional parametric uncertainty as in subquestion 4. In Chapter 6, we give a first step towards direct data-driven control that does not require any model at all, as in subquestion 5. This first step is performed for *deterministic* systems. We end this thesis by giving a conclusion and a vision of future work in Chapter 7.

Next, we highlight the contributions associated to each main chapter in this thesis and the papers that are not explicitly included.

Chapter 2

In this chapter, we consider discrete-time models with a continuous state space, whose state evolution is influenced by an additive, (possibly unbounded) stochastic influence. The models can be either linear time-invariant or nonlinear, and the

(possibly infinite-horizon) specifications are given in syntactically co-safe linear temporal logic. We develop a method that explicitly designs the coupling between a model and its (finite-state) abstraction by introducing a *coupling compensator*. This allows us to characterize the set of possible simulation relations and yields more freedom in the similarity quantification. Furthermore, this means that it is possible to efficiently compute the deviation bounds corresponding to the similarity quantification. As a result, when explicitly designing the coupling the lower bound on the satisfaction probability is more accurate. For linear time-invariant systems, we further describe how to include model reduction, which allows scalability to higher-dimensional models.

Corresponding papers:
B.C. van Huijgevoort, and S. Haesaert. *"Similarity quantification for linear stochastic systems: A coupling compensator approach."* Automatica, Vol. 144, 110476, 2022.

B.C. van Huijgevoort, S. Weiland, and S. Haesaert. *"Temporal logic control of nonlinear stochastic systems using a piecewise-affine abstraction."* IEEE Control Systems Letters, Vol. 7, pages 1039-1044, 2022.

Chapter 3

In this chapter, we develop a `MATLAB` tool that synthesizes provably correct controllers for stochastic systems influenced by an unbounded disturbance and that are subject to properties unbounded in time. The programming was a joint effort of the author of this thesis, Dr. S. Haesaert, with some help from O. Schön, MSc. from Newcastle University. The (developmental) software is owned by Dr. S. Haesaert.

Corresponding paper:
B.C. van Huijgevoort, O. Schön, S. Soudjani, and S. Haesaert. *"SySCoRe: Synthesis via Stochastic Coupling Relations."* 26th ACM international conference on hybrid systems: computation and control (HSCC), pages 1-11, 2023.

Chapter 4

In this chapter, we start with extending the method of Chapter 2 to improve scalability. More specifically, we include a variable precision by introducing multiple layers with different bounds on the confidence and precision. This improves the accuracy of the computed lower bound on the satisfaction probability substantially. Next, we include the possibility to have abstraction-free layers, which improves the applicability of this method to higher-dimensional systems while still achieving an overall high accuracy.

Corresponding papers:
B.C. van Huijgevoort, and S. Haesaert. *"Multi-layered simulation relations for linear stochastic systems."* IEEE European control conference (ECC), pages 728-733, 2021.

B.C. van Huijgevoort, S. Soudjani, and S. Haesaert. *"Specification-guided temporal logic control for linear stochastic systems: a multi-layered approach."* In preparation.

Chapter 5

In this chapter, we extend the method of Chapter 2 to stochastic systems with explicit parametric uncertainty by introducing a sub-simulation relation. Besides that, we use data to obtain a credible set for the uncertain parameter.

The research involved in this topic has been done by the author of this thesis in collaboration with O. Schön, MSc from Newcastle University (UK). O. Schön took the lead in the research while being supervised by both Dr. S. Haeasaert and Dr. S. Soudjani. As this work required deep knowledge of the coupling compensator in correct-by-design control synthesis, we have helped out with the understanding of the topic and with technical details and proofs. Similarly, an extension of the methods in the `SySCoRe` toolbox (Chapter 3) was key to perform the simulations of the paper, so we introduced the toolbox as well as assisting with the extension towards parametric uncertainty. For this, the author of this thesis has been extensively involved in the technical support. Besides that, a poster on this topic has been presented by the author of this thesis at HSCC '23.

Corresponding publications:
O. Schön, B.C. van Huijgevoort, S. Haesaert, and S. Soudjani. *"Correct-by-Design Control of Parametric Stochastic Systems."* IEEE Conference on Decision and Control (CDC), pages 5580-5587, 2022.

O. Schön, B.C. van Huijgevoort, S. Haesaert, and S. Soudjani. *"Poster Abstract: Data-Driven Correct-by-Design Control of Parametric Stochastic Systems."* 26th ACM international conference on hybrid systems: computation and control (HSCC), pages 1-2, 2023.

O. Schön, B.C. van Huijgevoort, S. Haesaert, and S. Soudjani. *"Bayesian Regression for Temporal Logic Control of Uncertain Systems."* Manuscript under review.

Chapter 6

In Chapter 6, we make an initial step towards direct data-driven methods that do not require an explicit model. Those methods are often referred to as *model-free*. We focus on deterministic systems and envision possible future directions for this promising line of research.

Corresponding paper:
B.C. van Huijgevoort, C. Verhoek, R. Tóth, and S. Haesaert. *"Direct data-driven signal temporal logic control of linear systems."* In preparation.

Other contributions

Here, we briefly discuss the contributions that are not explicitly included in this thesis.

Together with our colleagues from Newcastle University (UK), we have extended the work in Chapter 5 to stochastic systems where the stochastic influence does not come from a Gaussian distribution, but from a Gaussian mixture model. Since any probability distribution can be approximated by a Gaussian mixture model, this theoretically extends our work to stochastic systems where the stochastic influence comes from any distribution. Besides that, we also extend our work to compositional systems, where it is possible to split up the system and specifications in a particular manner.

Corresponding paper:

O. Schön, B.C. van Huijgevoort, S. Haesaert., and S. Soudjani. *"Verifying the Unknown: Correct-by-Design Control Synthesis for Networks of Stochastic Uncertain Systems."* Accepted for publication at the 62nd IEEE Conference on Decision and control (CDC).

We participated in the ARCH workshop on stochastic modeling for multiple years and the results are published (not peer-reviewed). This workshop and friendly competition focuses on formal verification and control of continuous- and hybrid-state systems. Together, we develop generic benchmarks and use them to compare our software tools to each other.

Corresponding papers:

A. Abate, H. Blom, J. Delicaris, S. Haesaert, B.C. van Huijgevoort, A. Lavaei, A. Remke, O. Schön, S. Schupp, F. Shmarov, S. Soudjani, L. Willemsen, and P. Zuliana. *"ARCH-COMP23 Category Report: Stochastic Modelling"*, In preparation.

A. Abate, H. Blom, J. Delicaris, S. Haesaert, A. Hartmanns, B.C. van Huijgevoort, A. Lavaei, H. Ma, M. Niehage, A. Remke, O. Schön, S. Schupp, S. Soudjani, and L. Willemsen. *"ARCH-COMP22 Category Report: Stochastic Modelling"*, Proceedings of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH), EPiC Series in Computing, Vol. 90, pages 113-141, 2022.

A. Abate, H. Blom, N. Cauchi, K. Degiorgio, M. Fränzle, E.M. Hahn, S. Haesaert, H. Ma, M. Oishi, C. Pilch, A. Remke, M. Salamati, S. Soudjani, B.C. van Huijgevoort, and A.P. Vinod. *"ARCH-COMP19 Category Report: Stochastic Modelling"*, 6th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH), EPiC Series in Computing, Vol. 61, pages 62-102, 2019.

Part I

Control of stochastic models

2

Quantified abstraction for stochastic systems

Synthesizing controllers for continuous-state stochastic systems automatically, while giving guarantees on the probability of satisfying (infinite-horizon) temporal logic specifications crucially depends on abstractions with a quantified accuracy. This is especially the case when considering accurate deviation bounds between a stochastic continuous-state model and its finite (reduced-order) abstraction. This similarity quantification with deviation bounds is often formalized by approximate stochastic simulation relations. However, computing these simulation relations is challenging and tends to give conservative relations. A key part of the simulation relations is the (implicit) choice of coupling between the stochastic transitions of the models. In this chapter, we introduce a parameterization of this coupling and resolve it dynamically via what we will define as a *coupling compensator*. First, we use this coupling compensator for linear stochastic systems to give a comprehensive approach to compute the simulation relations. More precisely, we develop a computational method that characterizes the set of possible simulation relations and gives a trade-off between the error contributions on the systems output and deviations in the transition probability.
Secondly, we extend the use of this coupling compensator to nonlinear systems. To handle the nonlinearity of the system effectively, we use finite-state abstractions based on piecewise-affine approximations together with tailored simulation relations that leverage the local affine structure.
Lastly, we show the effect of the coupling compensator on the guaranteed satisfaction probability for several case studies.

2.1 Introduction

Airplanes, cars, and power systems are examples of safety-critical control systems, whose reliable and autonomous functioning is critical. It is of interest to design controllers for these systems that provably satisfy formal specifications such as linear temporal logic (LTL) formulae (Pnueli 1977). For systems described by stochastic discrete-time models, these formal specifications have to be verified probabilistically. Despite recent advances (Desharnais et al. 2003; Julius and Pappas 2009; Zamani et al. 2014; Soudjani et al. 2015; Haesaert et al. 2017b; Cauchi and Abate 2019; Lavaei et al. 2019; Haesaert and Soudjani 2020; Lavaei et al. 2020a; Lavaei et al. 2021), the provably correct design of controllers for such stochastic models with continuous state spaces remains a challenging problem. Many of those methods (Zamani et al. 2014; Soudjani et al. 2015; Haesaert et al. 2017b; Cauchi and Abate 2019; Haesaert and Soudjani 2020; Lavaei et al. 2020a) rely on constructing a stochastic finite-state model or abstraction that approximates the original model. Such abstraction-based methods are often more suitable for complex temporal logic specifications, but their application to real-world problems tends to suffer from scalability issues and conservative lower bounds on the satisfaction probability. In this chapter, we focus on reducing the conservatism of an abstraction-based approach that applies to unbounded stochastic disturbances and specifications with an infinite time horizon.

A key factor in the conservatism is the quantification of the similarity between the original and abstract model for which approximate simulation relations (Desharnais et al. 2003; Zamani et al. 2014; Haesaert et al. 2017b; Haesaert and Soudjani 2020) and stochastic simulation functions (Julius and Pappas 2009; Lavaei et al. 2019) can be used. These methods inherently build on an implicit coupling of probabilistic transitions (Segala and Lynch 1994; Tkachev and Abate 2014). The latter shows that the coupling between stochastic processes is crucial, and omitting its explicit choice may lead to conservative results. Therefore, in this chapter we focus on the following research question.

Research Question

How to explicitly design the coupling between a continuous-state model and its abstraction, such that we can efficiently compute accurate approximate stochastic simulation relations?

Besides abstraction-based methods that leverage finite-state approximations, abstraction-free methods also exist. Next to methods that target specific model classes and limited reach-(avoid) specifications (Kariotoglou et al. 2017; Vinod et al. 2019), recent results based on barrier certificates (Huang et al. 2017; Jagtap et al. 2020; Nejati et al. 2020; Anand et al. 2021) can handle larger sets of specifications. Even though these methods suffer less from the curse of dimensionality, they are often restricted to specific model structures or specifications. For example, the barrier certificates in Jagtap et al. (2020) and Nejati et al. (2020) are limited to finite-time

horizon specification. Besides that, they require supermartingale conditions (Anand et al. 2021) or relaxed versions thereof (Jagtap et al. 2020; Nejati et al. 2020). Furthermore, for all of these methods, it is not known whether a solution can be found even if one exists and the computational complexity grows substantially with the length and complexity of the specification.

On the other hand, abstraction-based methods are very common in the provably correct design of controllers (Zamani et al. 2014; Soudjani et al. 2015; Haesaert et al. 2017b; Cauchi and Abate 2019; Haesaert and Soudjani 2020; Lavaei et al. 2020a) and they can in general handle more challenging specifications. In Lavaei et al. (2021), it has been shown that (ϵ, δ)-stochastic simulation relations (Haesaert et al. 2017b; Haesaert and Soudjani 2020) that quantify both the probabilistic deviation and the deviation in (output) trajectories can be used for compositional verification of large scale stochastic systems with nonlinear dynamics and that this outperforms results that leverage simulation functions. Therefore, in the first part of this chapter we focus on the design of efficient (ϵ, δ)-stochastic simulation relations via tailored coupling designs. Moreover, we will show that this allows us to characterize the set of coupling simulations and to trade off the error contributions of the output of the system with deviations in the transition probability. The main difference between this chapter and Haesaert et al. (2017b) is that in Haesaert et al. (2017b) the focus is on the conditions under which simulation relations can be established, while here, we are interested in finding such relations with efficient computation methods.

This chapter introduces a coupling compensator to leverage the freedom in coupling-based similarity relations such as (Haesaert et al. 2017b) via computationally attractive set-theoretic methods. To achieve this, we exploit the use of coupling probability measures through a coupling compensator. We start with some preliminaries and a mathematical formulation of the objective in the next section. In Section 2.3, we define the coupling compensator in a general manner that applies to both linear and nonlinear stochastic systems. Next, in Section 2.4 we develop a method to efficiently compute the deviation bounds for finite-state abstractions of linear stochastic systems by formulating it as a set-theoretic problem using the concept of controlled-invariant sets. To tackle large-scale systems, i.e., systems with a higher-dimensional state space, we also apply the coupling compensator to reduced-order models. In Section 2.5, we extend the theory on the coupling compensator to nonlinear stochastic systems by using a piecewise-affine abstraction. In Section 2.6, we explain how to perform the controller synthesis. To evaluate the benefits of this method, we consider specifications written using syntactically co-safe linear temporal logic (Kupferman and Vardi 2001; Belta et al. 2017) and analyze the influence of both the deviation bounds on the satisfaction probability for linear systems (Section 2.7). In the same section, we apply the model-order reduction method to a system with a 7-dimensional state space and apply our method to a nonlinear system.

2.2 Preliminaries

We denote the set of positive real numbers by $\mathbb{R}^+$ and the n-dimensional identity matrix by I_n. We limit our attention to spaces that are finite, Euclidean, or Polish. For a set $\mathbb{X}$ in Euclidean space, which is measurable and separable, the Borel measurable space is denoted as $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$ with $\mathcal{B}(\mathbb{X})$ the σ-algebra of the Borel sets (Knapp 2016). A probability measure $\mathbb{P}$ over this space has realizations $x \sim \mathbb{P}$ with $x \in \mathbb{X}$. The set of probability measures on the measurable space $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$ is denoted by $\mathcal{P}(\mathbb{X})$. The weighted two-norm $||x||_D$ is defined as $||x||_D = \sqrt{x^\top D x}$. The Minkowski sum of two sets A and B is defined as $A \oplus B := \{a + b \mid a \in A, b \in B\}$.

Model. Consider a system whose behavior can be modeled by a discrete-time nonlinear stochastic difference equation

$$\mathbf{M} : \begin{cases} x(t+1) & = f(x(t), u(t), w(t)) \\ y(t) & = h(x(t)), \quad \forall t \in \{0, 1, 2, \dots\}, \end{cases} \tag{2.1}$$

initialized with $x(0) = x_0$ and with state $x(t) \in \mathbb{X} \subset \mathbb{R}^{n_x}$, input $u(t) \in \mathbb{U} \subset \mathbb{R}^{n_u}$, disturbance $w(t) \in \mathbb{W} \subset \mathbb{R}^{n_w}$, and output $y(t) \in \mathbb{Y} \subset \mathbb{R}^{n_y}$. We assume that the functions $f : \mathbb{X} \times \mathbb{U} \times \mathbb{W} \to \mathbb{X}$ and $h : \mathbb{X} \to \mathbb{Y}$ are Borel measurable and sufficiently smooth. Furthermore, $w(t)$ is an independent and identically distributed (i.i.d.) noise signal with realizations $w(t) \sim \mathbb{P}_w$. Throughout this chapter, we use $x(t+1)$, x_{t+1}, and x^+ interchangeably to denote a time update of a variable.

Remark 2.1. This model (2.1) can equivalently be written as a general Markov Decision Process (gMDP) as defined in Chapter 4.

A (finite) path $\boldsymbol{\omega}_{[0,t]} := x_0, u_0, x_1, u_1, \dots, x_t$ of $\mathbf{M}$ consists of states x_k and inputs u_k, for which $x_{k+1} = x(k+1)$ follows (2.1) for a given state $x(k) = x_k$, input $u(k) = u_k$ and disturbance $w(k)$ at time steps k. A control strategy $\boldsymbol{\mu} := \mu_0, \mu_1, \mu_2 \dots$ consists of maps $\mu_t(\boldsymbol{\omega}_{[0,t]}) \in \mathbb{U}$ assigning an input $u(t)$ to each finite path $\boldsymbol{\omega}_{[0,t]}$ generated by the model (2.1), that is $u(t) = \mu_t(\boldsymbol{\omega}_{[0,t]})$. In this work, we focus on stationary control strategies $\mathbf{C} : u(t) = \mu(\boldsymbol{\omega}_{[0,t]})$ that have a *finite memory* (formal definition given in Section 5.2b). We denote the controlled system with $\mathbf{M} \times \mathbf{C}$, similar to the notation of a feedback composition as in Tabuada (2009, Section 6.1)

Specifications. Consider specifications written using syntactically co-safe linear temporal logic (scLTL) (Kupferman and Vardi 2001; Belta et al. 2017) a subset of LTL (Pnueli 1977). Denote with $\text{AP} = \{p_1, \dots, p_N\}$ the set of atomic propositions and let 2^{AP} be the alphabet with letters $\pi \in 2^{\text{AP}}$. An infinite string of letters is a word $\boldsymbol{\pi} = \pi_0 \pi_1 \pi_2 \dots$ with associated suffix $\boldsymbol{\pi}_t = \pi_t \pi_{t+1} \pi_{t+2} \dots$.

Definition 2.1. *An scLTL formula ϕ is recursively defined over a set of atomic proposition as*

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

with $p \in \text{AP}$.

The semantics of scLTL is defined for the suffixes $\boldsymbol{\pi}_t$ as follows. An atomic proposition $\boldsymbol{\pi}_t \models p$ holds if $p \in \pi_t$, while a negation $\boldsymbol{\pi}_t \models \neg p$ holds if $\boldsymbol{\pi}_t \not\models p$. A conjunction

$\boldsymbol{\pi}_t \models \phi_1 \wedge \phi_2$ holds if both $\boldsymbol{\pi}_t \models \phi_1$ and $\boldsymbol{\pi}_t \models \phi_2$ hold. A disjunction $\boldsymbol{\pi}_t \models \phi_1 \vee \phi_2$ holds if either $\boldsymbol{\pi}_t \models \phi_1$ or $\boldsymbol{\pi}_t \models \phi_2$ holds. A next operator $\boldsymbol{\pi}_t \models \bigcirc\phi$ holds if $\boldsymbol{\pi}_{t+1} \models \phi$ is true. An until operator $\boldsymbol{\pi}_t \models \phi_1 \,\mathsf{U}\, \phi_2$ holds if there exists an $i \in \mathbb{N}$ such that $\boldsymbol{\pi}_{t+i} \models \phi_2$ and for all $j \in \mathbb{N}, 0 \leq j < i$ we have $\boldsymbol{\pi}_{t+j} \models \phi_1$. By combining multiple operators, the *eventually* operator $\Diamond\phi := \text{true}\,\mathsf{U}\,\phi$ can also be defined. A labeling function $L : \mathbb{Y} \to 2^{\mathrm{AP}}$ assigns letters $\pi = L(y)$ to outputs $y \in \mathbb{Y}$. A state trajectory $\mathbf{x} := x_0 x_1 x_2 \ldots$ satisfies a specification ϕ, written $\mathbf{x} \models \phi$, iff the generated word $\boldsymbol{\pi}$ satisfies ϕ at time 0, i.e., $\boldsymbol{\pi}_0 \models \phi$. The satisfaction probability of a specification is the probability that words generated by the controlled system $\mathbf{M} \times \mathbf{C}$ satisfy the specification ϕ, denoted as $\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi)$.

General problem. The goal of this work is to automatically develop a controller $\mathbf{C}$, such that the probability that the controlled system $\mathbf{M} \times \mathbf{C}$ satisfies a specification ϕ is higher than some threshold. Mathematically, this is formulated as follows.

Problem 2.1. *Given model* $\mathbf{M}$ *as in* (2.1)*, an scLTL specification* ϕ *and a probability* $p_\phi \in [0, 1]$*, design a controller* $\mathbf{C}$*, such that*

$$\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi) \geq p_\phi. \tag{2.2}$$

2.3 Coupling compensator for similarity quantification: Problem statement and approach

2.3a Problem statement

The design of controller $\mathbf{C}$ and its exact quantification $\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi)$ is computationally hard for continuous-state stochastic models (Abate et al. 2008). Therefore, the approximation and similarity quantification of continuous-state models is a basic step in the provably correct design of controllers. These steps implicitly depend on the coupling between the continuous-state model and its approximation. This section introduces the underlying coupling problem and proposes an approach to efficiently solve this problem for general nonlinear systems.

Problem statement. Suppose that model $\mathbf{M}$ given in (2.1), has an abstraction written as

$$\hat{\mathbf{M}} : \begin{cases} \hat{x}(t+1) & = \hat{f}(\hat{x}(t), \hat{u}(t), \hat{w}(t)), \\ \hat{y}(t) & = \hat{h}(\hat{x}(t)), \end{cases} \tag{2.3}$$

initialized with $\hat{x}(0) = \hat{x}_0$, and with state $\hat{x} \in \hat{\mathbb{X}}$, input $\hat{u} \in \hat{\mathbb{U}}$, disturbance $\hat{w} \in \mathbb{W}$ and output $\hat{y} \in \hat{\mathbb{Y}} = \mathbb{Y}$, and with functions $\hat{h} : \hat{\mathbb{X}} \to \mathbb{Y}$ and $\hat{f} : \hat{\mathbb{X}} \times \hat{\mathbb{U}} \times \mathbb{W} \to \hat{\mathbb{X}}$. Here, $\hat{\mathbb{X}}$ and $\hat{\mathbb{U}}$ can be finite and $\hat{w}(t)$ is an i.i.d. noise sequence with realizations $\mathbb{P}_{\hat{w}}$.

We quantify the difference between the original model $\mathbf{M}$ and the abstract model $\hat{\mathbf{M}}$ by bounding the difference between the outputs y and $\hat{y}$. For this, we need to resolve the choice of inputs $u, \hat{u}$, and the stochastic disturbance. The former is often done by equating $u(t) = \hat{u}(t)$ and analyzing the worst-case error. An *interface*

function (Girard and Pappas 2009) generalizes this by refining the control input $\hat{u}$ to u as a function of the current states

$$\mathcal{U}_v : \hat{\mathbb{U}} \times \hat{\mathbb{X}} \times \mathbb{X} \to \mathbb{U}. \tag{2.4}$$

More specifically, we get $u(t) = \mathcal{U}_v(\hat{u}(t), \hat{x}(t), x(t)) \forall t \in \{1, 2, \dots\}$, with $\mathcal{U}_v$ a static function. In a similar way, we can resolve the stochastic disturbance. We first relate the probability measures $\mathbb{P}_{\hat{w}}$ and $\mathbb{P}_w$ of the stochastic disturbances $\hat{w}$ and w as follows.

Definition 2.2 (Coupling of probability measures)**.** *A coupling (Hollander 2012) of two probability measures $\mathbb{P}_{\hat{w}}$ and $\mathbb{P}_w$ on the same measurable space $(\mathbb{W}, \mathcal{B}(\mathbb{W}))$ is any probability measure $\mathcal{W}$ on the product measurable space $(\mathbb{W} \times \mathbb{W}, \mathcal{B}(\mathbb{W} \times \mathbb{W}))$ whose marginals are $\mathbb{P}_{\hat{w}}$ and $\mathbb{P}_w$, that is*[1]*,*

$$\mathbb{P}_{\hat{w}} = \mathcal{W} \cdot \hat{\pi}^{-1}, \quad \mathbb{P}_w = \mathcal{W} \cdot \bar{\pi}^{-1}, \tag{2.5}$$

for which $\hat{\pi}$ and $\bar{\pi}$ are projections, respectively defined by

$$\hat{\pi}(\hat{w}, w) = \hat{w}, \quad \bar{\pi}(\hat{w}, w) = w, \ \forall (\hat{w}, w) \in \mathbb{W} \times \mathbb{W}.$$

Remark 2.2. There are many possible couplings. Two trivial examples are:

$$\begin{aligned} &\mathcal{W} = \mathbb{P}_w \otimes \mathbb{P}_{\hat{w}}, \text{ with } \mathbb{P}_w, \mathbb{P}_{\hat{w}} \text{ arbitrary} && \Leftrightarrow \quad w, \hat{w} \text{ are independent,} \\ &\mathbb{P}_w = \mathbb{P}_{\hat{w}} \text{ with } \mathcal{W} \text{ on the diagonal} && \Leftrightarrow \quad w = \hat{w}. \end{aligned}$$

We can also design $\mathcal{W}$ as a measurable function of the current state pair and actions, similarly to the interface function. This yields a Borel measurable stochastic kernel associating to each $(\hat{u}, \hat{x}, x)$ a probability measure

$$\bar{\mathcal{W}} : \hat{\mathbb{U}} \times \hat{\mathbb{X}} \times \mathbb{X} \to \mathcal{P}(\mathbb{W}^2) \tag{2.6}$$

that couples probability measures $\mathbb{P}_{\hat{w}}$ and $\mathbb{P}_w$ as in Definition 2.2. We can now define a composed model as illustrated in Figure 2.1 as follows.

Definition 2.3 (Composed model)**.** *Given an interface function (2.4) and a coupling measure (2.6) resolving the inputs and disturbances, respectively, the model $\hat{\mathbf{M}} \| \mathbf{M}$ composed of models $\hat{\mathbf{M}}$ and $\mathbf{M}$ can be defined as*

$$\begin{aligned} \begin{bmatrix} \hat{x}(t+1) \\ x(t+1) \end{bmatrix} &= \begin{bmatrix} \hat{f}(\hat{x}(t), \hat{u}(t), \hat{w}(t)) \\ f(x(t), \mathcal{U}_v(\hat{u}(t), \hat{x}(t), x(t)), w(t)) \end{bmatrix} \\ \begin{bmatrix} \hat{y}(t) \\ y(t) \end{bmatrix} &= \begin{bmatrix} \hat{h}(\hat{x}(t)) \\ h(x(t)) \end{bmatrix} \end{aligned} \tag{2.7}$$

with states $(\hat{x}, x) \in \hat{\mathbb{X}} \times \mathbb{X}$, input $\hat{u} \in \hat{\mathbb{U}}$, coupled disturbances $(\hat{w}, w) \sim \bar{\mathcal{W}}(\cdot \,|\hat{u}, \hat{x}, x)$, and outputs $\hat{y}, y \in \mathbb{Y}$.

[1] Requirement (2.5) on $\mathcal{W}$ can be equivalently given as

$$\begin{aligned} &\mathcal{W}(\hat{A} \times \mathbb{W}) = \mathbb{P}_{\hat{w}}(\hat{A}) \text{ for all } \hat{A} \in \mathcal{B}(\mathbb{W}) \\ &\mathcal{W}(\mathbb{W} \times A) = \mathbb{P}_w(A) \text{ for all } A \in \mathcal{B}(\mathbb{W}). \end{aligned}$$

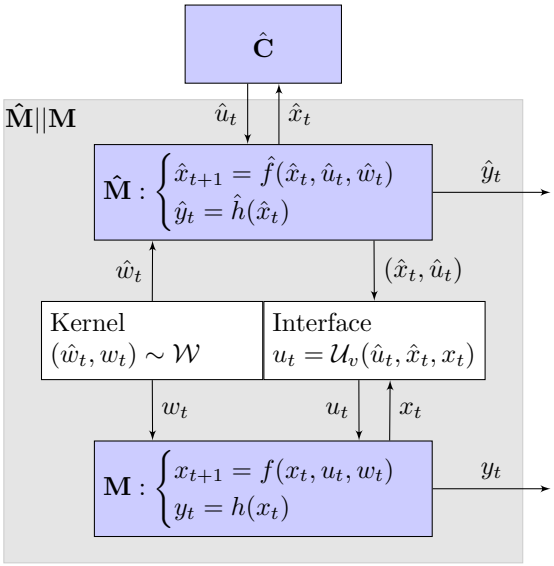


Figure 2.1: Composed model $\hat{\mathbf{M}}||\mathbf{M}$ as defined in Definition 2.3.

The deviation between $\hat{\mathbf{M}}$ and $\mathbf{M}$ can be expressed as the metric $\mathbf{d}_{\mathbb{Y}}(\hat{y}, y) := ||y - \hat{y}||$, with $\hat{y}, y \in \mathbb{Y}$ for the traces of the composed model. Similar notions have been used in Julius and Pappas (2009), Zamani et al. (2014), and Haesaert and Soudjani (2020). Note that the choice of coupling is a critical part of this model composition. The problem refining the research question from the introduction can now be formulated as follows.

Problem 2.2. *Explicitly design the coupling of probabilistic transitions to efficiently quantify the similarity between models* $\hat{\mathbf{M}}$ *and* $\mathbf{M}$ *as in* (2.3) *and* (2.1).

2.3b A coupling compensator approach

As in Haesaert et al. (2017b), consider an approximate simulation relation to quantify the similarity between the stochastic models $\hat{\mathbf{M}}$ and $\mathbf{M}$. The following definition is a special case of Definition 9 in Haesaert et al. (2017b) applicable to stochastic difference equations.

Definition 2.4 ((ϵ, δ)-stochastic simulation relation)**.** *Let stochastic difference equations* $\hat{\mathbf{M}}$ *and* $\mathbf{M}$ *with metric output space* $(\mathbb{Y}, \boldsymbol{d}_{\mathbb{Y}})$ *be composed into* $\hat{\mathbf{M}}||\mathbf{M}$ *based on the interface function* $\mathcal{U}_v$ (2.4) *and the Borel measurable stochastic kernel* $\bar{\mathcal{W}}$ (2.6)*. If there exists a measurable relation* $\mathcal{R} \subseteq \hat{\mathbb{X}} \times \mathbb{X}$*, such that*

1. $(\hat{x}_0, x_0) \in \mathcal{R}$,
2. $\forall (\hat{x}, x) \in \mathcal{R} : \boldsymbol{d}_{\mathbb{Y}}(\hat{y}, y) \leq \epsilon$, *with* $y = h(x)$ *and* $\hat{y} = \hat{h}(\hat{x})$,
3. $\forall (\hat{x}, x) \in \mathcal{R}, \forall \hat{u} \in \hat{\mathbb{U}} : (\hat{x}^+, x^+) \in \mathcal{R}$ *holds with probability at least* $1 - \delta$,

then $\hat{\mathbf{M}}$ *is* (ϵ, δ)*-stochastically simulated by* $\mathbf{M}$ *and this simulation relation is denoted as* $\hat{\mathbf{M}} \preceq^{\delta}_{\epsilon} \mathbf{M}$.

Here, ϵ and δ denote the output and probability deviation respectively. Furthermore, state updates $\hat{x}^+$ and x^+ are the abbreviations of $\hat{x}(t+1)$ and $x(t+1)$. The choice of interface $\mathcal{U}_v$ impacts how much of the deviations between $x(t)$ and $\hat{x}(t)$ is compensated at the next time instance $x(t+1)$ and $\hat{x}(t+1)$. Similarly, the coupling $\bar{\mathcal{W}}$ induces a term $w - \hat{w}$ that can compensate for state deviations. We can choose to explicitly parameterize the coupling based on this compensator term. To this end, the notion of a coupling compensator is defined next.

Definition 2.5 (Coupling compensator)**.** *Consider probability measures* $\mathbb{P}_{\hat{w}}$ *and* $\mathbb{P}_w$ *on the same measurable space* $(\mathbb{W}, \mathcal{B}(\mathbb{W}))$*. Given a bounded set* Γ *and a probability* $1-\delta$*, we say that* $\bar{\mathcal{W}}_\gamma$ *is a coupling compensator if it parameterizes the coupling, such that for any compensator value* $\gamma \in \Gamma$ *we obtain the event* $w - \hat{w} = \gamma$ *with probability at least* $1-\delta$*, that is,* $\bar{\mathcal{W}}_\gamma(w - \hat{w} = \gamma) \geq 1 - \delta$.

Remark 2.3. Without loss of generality, this definition is applicable to the case where $\hat{w}$ is in a lower-dimensional space than w. In that case, the disturbance space $\mathbb{W}$ is only partially used for the coupling compensator.

In the next section, we resolve Problem 2.2 for (ϵ, δ)-simulation relations by either choosing the coupling compensator as a linear mapping of the state deviations when $\hat{\mathbb{X}} \subset \mathbb{X}$, that is, $\bar{\mathcal{W}}(\cdot|\hat{u}, \hat{x}, x) = \bar{\mathcal{W}}_\gamma$ with $\gamma = F(x - \hat{x})$ or as a linear mapping of the projected state deviation when $\hat{\mathbb{X}}$ and $\mathbb{X}$ are of a different dimension.

In Section 2.5a, we resolve Problem 2.2 for nonlinear systems by choosing the coupling compensator as a linear mapping of the state deviations in a piecewise manner, that is $\bar{\mathcal{W}}(\cdot \mid \hat{u}, \hat{x}, x) = \bar{\mathcal{W}}_i(\cdot \mid \hat{u}, \hat{x}, x) = \bar{\mathcal{W}}_{\gamma_i}$ if $\hat{x} \in \hat{P}_i$, with $\gamma_i = F_i(x - \hat{x})$. Here, i denotes the partition of the piecewise-affine approximation.

2.4 Coupling compensator for finite abstractions of linear stochastic systems

2.4a A coupling compensator for linear stochastic systems

Consider a linear time-invariant (LTI) system whose behavior is modeled by the stochastic difference equation

$$\mathbf{M}: \begin{cases} x(t+1) & = Ax(t) + Bu(t) + B_w w(t) \\ y(t) & = Cx(t), \end{cases} \tag{2.8}$$

initialized with x_0 and with matrices $A \in \mathbb{R}^{n_x \times n_x}, B \in \mathbb{R}^{n_x \times n_u}, B_w \in \mathbb{R}^{n_x \times n_w}, C \in \mathbb{R}^{n_u \times n_x}$, state $x \in \mathbb{X}$, input $u \in \mathbb{U}$, and output $y \in \mathbb{Y}$. Furthermore, the stochastic disturbance $w \in \mathbb{W}$ is an i.i.d. Gaussian process.

Remark 2.4. Without loss of generality, we assume that the disturbance $w(t)$ has unbounded Gaussian distribution with mean 0 and identity covariance matrix, that is $w \sim \mathcal{N}(0, I)$. Any system (2.8) with disturbance $w \sim \mathcal{N}(\mu, \Sigma)$ can be rewritten to a system in the same class with an additional affine term (Allen et al. 2008).

To leverage model checking results (Baier and Katoen 2008) for finite-state Markov decision processes, we can abstract the model (2.8) to a finite-state representation.

Finite-state abstraction $\hat{\mathbf{M}}$. To obtain a finite-state model $\hat{\mathbf{M}}$, partition the state space $\mathbb{X}$ in a finite number of regions $\mathbb{A}_j \subset \mathbb{X}$, such that $\bigcup_j \mathbb{A}_j = \mathbb{X}$ and $\mathbb{A}_j \cap \mathbb{A}_l = \emptyset$ for $j \neq l$. Here, $j \in \{1, 2, \ldots, N_A\}$, with N_A the finite number of regions. Choose a representative point in each region, $\hat{X}_j \in \mathbb{A}_j$ and define the set of abstract states $\hat{x} \in \hat{\mathbb{X}}$ based on these representative points[2], that is, $\hat{\mathbb{X}} := \{\hat{X}_1, \hat{X}_2, \hat{X}_3, \ldots, \hat{X}_{N_A}\}$. Furthermore, a finite set of inputs is selected from $\mathbb{U}$ and defines $\hat{\mathbb{U}}$. To define the dynamics of the abstract model, consider the operator $\Pi : \mathbb{X} \to \hat{\mathbb{X}}$ that maps states $x \in \mathbb{X}$ to the representative point $\hat{X}_j \in \mathbb{A}_j$ whenever $x \in \mathbb{A}_j$ Using Π to obtain a finite-state abstraction of $\mathbf{M}$, we get the abstract model $\hat{\mathbf{M}}$

$$\hat{\mathbf{M}} : \begin{cases} \hat{x}(t+1) & = \Pi(A\hat{x}(t) + B\hat{u}(t) + B_w\hat{w}(t)) \\ \hat{y}(t) & = C\hat{x}(t), \end{cases} \tag{2.9}$$

with $\hat{x} \in \hat{\mathbb{X}} \subset \mathbb{X}, \hat{u} \in \hat{\mathbb{U}} \subset \mathbb{U}$, and $\hat{w} \sim \mathcal{N}(0, I)$, and initialized with $\hat{x}_0 \in \hat{\mathbb{X}}$. This initial state is the associated representative point, that is $\hat{x}_0 = \hat{X}_j$ if $x_0 \in \mathbb{A}_j$ or equivalently $\hat{x}_0 = \Pi(x_0)$. The abstract model $\hat{\mathbf{M}}$ can also be written as the following LTI system

$$\hat{\mathbf{M}} : \begin{cases} \hat{x}(t+1) = A\hat{x}(t) + B\hat{u}(t) + B_w\hat{w}(t) + \beta(t) \\ \hat{y}(t) \quad\quad = C\hat{x}(t), \end{cases} \tag{2.10}$$

by introducing the deviation $\beta(t)$ as in Haesaert and Soudjani (2020). The $\beta(t)$-term denotes the deviation caused by the mapping Π in (2.9) and takes values in the following bounded set $\mathcal{B} := \bigcup_j \{\hat{X}_j - x_j | x_j \in \mathbb{A}_j\}$. At each time step t, the deviation $\beta(t) \in \mathcal{B} \subseteq \mathbb{R}^{n_x}$ is a function of $\hat{x}(t), \hat{u}(t)$ and $\hat{w}(t)$, however, for simplicity we write $\beta(t)$.

Similarity quantification of $\hat{\mathbf{M}}$. To quantify the similarity between the abstract model $\hat{\mathbf{M}}$ and the original model $\mathbf{M}$, we use the notion of (ϵ, δ)-stochastic simulation relation given in Definition 2.4. Without loss of generality we limit the interface function to

$$u(t) := \hat{u}(t). \tag{2.11}$$

Next, we show that a coupling compensator can be computed based on the maximal coupling between two probability measures and that the linear compensator can be

[2]Beyond the given representative points, one generally adds a sink state to both the continuous- and the finite-state model to capture transitions that leave the bounded set of states.

used to solve the similarity quantification efficiently. Based on the composed model (c.f., Definition 2.3), we can define the error dynamics between (2.8) and (2.10) as

$$x_\Delta^+(t) = Ax_\Delta(t) + B_w(w(t) - \hat{w}(t)) - \beta(t), \tag{2.12}$$

where the state x_Δ and state update x_Δ^+ are the abbreviations of $x_\Delta(t) := x(t) - \hat{x}(t)$ and $x_\Delta(t+1)$, respectively. Furthermore, the stochastic disturbances $(\hat{w}, w)$ are generated by the coupling compensator $\bar{\mathcal{W}}_\gamma$ as in (2.6) with $w - \hat{w}$ the *coupling compensator term*.

The error dynamics can be used to efficiently compute the simulation relation, denoted as $\mathcal{R}$. In contrast to Julius and Pappas (2009) and Blute et al. (1997) and Desharnais et al. (2004), which quantify the deviation between the abstract and original model either completely on ϵ or completely on δ by fixing $\bar{\mathcal{W}}_\gamma$, we design a coupling compensator $\bar{\mathcal{W}}_\gamma$ with compensator value γ to achieve a preferred trade-off between ϵ and δ. Conditioned on event $w - \hat{w} = \gamma$ as in Definition 2.5 the error dynamics (2.12) reduce to

$$x_\Delta^+(t) = Ax_\Delta(t) + B_w\gamma(t) - \beta(t) \tag{2.13}$$

and hold with a probability of $\bar{\mathcal{W}}(w - \hat{w} = \gamma \,|\, \hat{u}, \hat{x}, x) = \bar{\mathcal{W}}_\gamma(w - \hat{w} = \gamma)$ that is at least larger than $1 - \delta$ for all $\gamma \in \Gamma$. For a given $\gamma \in \Gamma$, we can compute an optimal coupling $\bar{\mathcal{W}}_\gamma$ as follows. First, we introduce random variable $\hat{w}_\gamma \sim \mathcal{N}(\gamma, I)$ to replace the abstract disturbance

$$\hat{w}(t) = \hat{w}_\gamma(t) - \gamma(t). \tag{2.14}$$

Next, we find the coupling $\bar{\mathcal{W}}_\gamma$ for $\hat{w}$ and w by finding a maximal coupling of $\hat{w}_\gamma$ and w after which we can directly obtain $\bar{\mathcal{W}}_\gamma$ for $\hat{w}_\gamma$ and w. The computation of a maximal coupling in $\mathcal{P}(\mathbb{W} \times \mathbb{W})$ can be found in Hollander (2012) and builds on top of maximizing the probability mass that can be located on the diagonal $w - \hat{w}_\gamma = 0$. Denote with $\rho(\,\cdot\,|0, I)$ and $\hat{\rho}(\,\cdot\,|\gamma, I)$ the respective probability density functions of $w \sim \mathcal{N}(0, I)$ and $\hat{w}_\gamma \sim \mathcal{N}(\gamma, I)$. As in Hollander (2012), we construct a maximal coupling $\bar{\mathcal{W}}_\gamma$ that has on its diagonal $w - \hat{w}_\gamma = 0$ the sub-probability distribution

$$\rho \wedge \hat{\rho} := \min(\rho, \hat{\rho}), \tag{2.15}$$

where min denotes the minimal value of the probability density function for different values of w. We can now establish a relation between deviation δ and value γ.

Lemma 2.1. *Consider two normal distributions $\mathbb{P}_w := \mathcal{N}(0, I)$ and $\mathbb{P}_{\hat{w}_\gamma} := \mathcal{N}(\gamma, I)$ with $\gamma \in \Gamma$. Then there exists a coupled distribution $\mathcal{W}_\gamma$ such that*

$$w - \hat{w}_\gamma = 0 \text{ for } (\hat{w}_\gamma, w) \sim \mathcal{W}_\gamma$$

with a probability of at least

$$1 - \delta := \inf_{\gamma \in \Gamma} 2\,\mathrm{cdf}(-\tfrac{1}{2}||\gamma||). \tag{2.16}$$

Here, $\mathrm{cdf}(\cdot)$ denotes the cumulative distribution function of a one-dimensional Gaussian distribution $\mathcal{N}(0, 1)$.

Proof. First, an analytical expression for the maximal coupling of two disturbances $w \sim \mathcal{N}(0, I)$ and $\hat{w}_\gamma \sim \mathcal{N}(\gamma, I)$ is derived. Their probability density functions are denoted by $\rho(\cdot\,|0, I)$ and $\hat{\rho}(\cdot\,|\gamma, I)$, respectively. The maximal coupling is based on equation (2.15). The probability density function of this maximal coupling is denoted as $\rho_w : \mathbb{W} \times \mathbb{W} \to \mathbb{R}^+$ and can be computed as follows. Denote the sub-probability density function $\rho_{\min}(w) = \min(\rho(w), \hat{\rho}(w))$, with $\Delta_\gamma = \int_{\mathbb{R}^{n_w}} \rho_{\min}(w)\mathrm{d}w$ and define the coupling density function as

$$\rho_w(w, \hat{w}_\gamma) = \rho_{\min}(w)\delta_{\hat{w}_\gamma}(w) + \frac{(\rho(w) - \rho_{\min}(w))(\hat{\rho}(\hat{w}_\gamma) - \rho_{\min}(\hat{w}_\gamma))}{(1 - \Delta_\gamma)}, \tag{2.17}$$

with $\delta_{\hat{w}_\gamma}(w)$ the shifted Dirac delta function equal to $+\infty$ if equality $w = \hat{w}_\gamma$ holds and 0 otherwise. The first term of the coupling (2.17) puts only weight on the diagonal $w = \hat{w}_\gamma$. The second term puts the remaining probability density in an independent fashion. The sub-probability Δ_γ can be computed as

$$\Delta_\gamma = \int_{\mathbb{R}^{n_w}} \min(\rho(w), \hat{\rho}(w))\mathrm{d}w = \int_E \rho(w)\mathrm{d}w + \int_{\hat{E}} \hat{\rho}(\hat{w}_\gamma)\mathrm{d}\hat{w}_\gamma. \tag{2.18}$$

Here, half spaces $\hat{E}$ and E denote the respective regions satisfying $\rho > \hat{\rho}$ and $\rho \leq \hat{\rho}$. These regions can be represented as n_w-dimensional half-spaces.

As mentioned before, $\rho(\cdot\,|0, I)$ and $\hat{\rho}(\cdot\,|\gamma, I)$ are probability density functions of Gaussian distributions w and $\hat{w}_\gamma$ and therefore, ρ and $\hat{\rho}$ are strictly decreasing functions for increasing values of $||w||$ and $||w-\gamma||$ respectively. Furthermore, these two functions are equal except for a γ-shift. This implies that for a given point w if

- $||w|| < ||w - \gamma||$ then $\rho(w) > \hat{\rho}(w)$ (half space $\hat{E}$)
- $||w|| \geq ||w - \gamma||$ then $\rho(w) \leq \hat{\rho}(w)$ (half space E).

This last item shows that the half-spaces $\hat{E}$ (1st item) and E (2nd item) are separated by a hyper-plane through the point $w = \frac{1}{2}\gamma$ and perpendicular to the vector γ. This hyper-plane, denoted by H is characterized by $H := \left\{ w \in \mathbb{R}^{n_w} \mid \gamma^T w - \frac{1}{2}||\gamma||^2 = 0 \right\}$, and illustrated in Figure 2.2.

Since ρ and $\hat{\rho}$ are Gaussian density distribution that are equal up to γ-shift, as depicted in 2D in Figure 2.2, the integrals in (2.18) are equal to each other and $\Delta_\gamma = 2\int_E \rho(w)\mathrm{d}w$. It is trivial to see that this integral evaluates to $\Delta_\gamma = 2\,\mathrm{cdf}(-\frac{1}{2}||\gamma||)$. To obtain the worst case probability as in (2.16) we need to take into account all possible values of γ as $1 - \delta := \inf_{\gamma\in\Gamma} \Delta_\gamma = \inf_{\gamma\in\Gamma} 2\,\mathrm{cdf}(-\frac{1}{2}||\gamma||)$. This concludes the proof of Lemma 2.1. □

This lemma shows that by choosing a *maximal coupling* the error dynamics (2.13) hold with a probability of at least $1 - \delta$. We can now quantify the similarity via robust controlled positively invariant sets, also referred to as controlled-invariant sets in the remainder of the paper. Here, we consider the error dynamics (2.13) as a system with constrained input γ and bounded disturbance β.

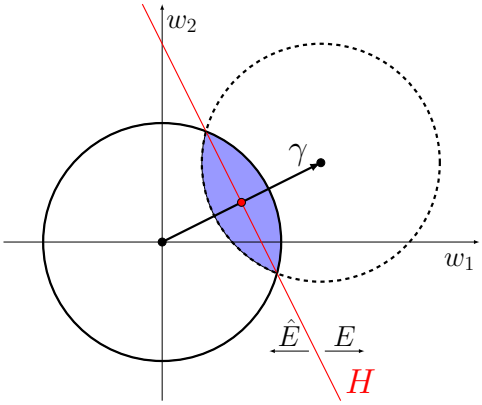


Figure 2.2: Level sets of probability density functions $\rho(\cdot|0, I)$ (black circle) and $\hat{\rho}(\cdot|\gamma, I)$ (dashed circle). Half spaces $\hat{E}$ and E are respectively the $\mathbb{R}^2$-plane left and right of hyper-plane H (red line). The area underneath $\min(\rho, \hat{\rho})$ *for these level sets* is indicated in blue.

Definition 2.6 (Controlled invariance)**.** *A set S is a (robust) controlled (positively) invariant set (Blanchini and Miani 2008) for the error dynamics given in (2.13) with $\gamma \in \Gamma$ and $\beta \in \mathcal{B}$, if for all states $x_\Delta \in S$, there exists an input $\gamma \in \Gamma$, such that for any disturbance $\beta \in \mathcal{B}$ the next state satisfies $x_\Delta^+ \in S$.*

We can quantify the similarity as follows.

Theorem 2.1. *Consider models $\mathbf{M}$ and $\hat{\mathbf{M}}$ with error dynamics (2.13) for which controlled-invariant set S is given.*

$$\text{If } \epsilon \geq \sup_{x_\Delta \in S} ||Cx_\Delta|| \text{ and } \delta \geq \sup_{\gamma \in \Gamma} \left(1 - 2\,\mathrm{cdf}(-\frac{1}{2}||\gamma||)\right)$$

then $\hat{\mathbf{M}}$ is (ϵ, δ)-stochastically simulated by $\mathbf{M}$ as in Definition 2.4, denoted as $\hat{\mathbf{M}} \preceq_\epsilon^\delta \mathbf{M}$.

Proof. Consider simulation relation

$$\mathcal{R} := \left\{(\hat{x}, x) \in \hat{\mathbb{X}} \times \mathbb{X} \,|\, (\hat{x}, x) \in S\right\}. \tag{2.19}$$

To prove that $\hat{\mathbf{M}}$ is (ϵ, δ)-stochastically simulated by $\mathbf{M}$ under the conditions given in Theorem 2.1, the simulation relation in Definition 2.4 is proven point by point.

1. *Initial condition.* Since $\hat{x}_0$ is inside the region that x_0 is in, the distance between $\hat{x}_0$ and x_0 is bounded by $\mathcal{B}$, that is, $\hat{x}_0 - x_0 \in \mathcal{B}$. Since it trivially holds that $\mathcal{B} \subseteq S$, (q.v. Theorem 5.2 in Blanchini and Miani (2008)) we also have $x_\Delta(0) = x_0 - \hat{x}_0 \in S$. This implies that the inclusion $(\hat{x}_0, x_0) \in \mathcal{R}$ holds for simulation relation (2.19).

2. ϵ-*Accuracy.* The inequality $\epsilon \geq \sup_{x_\Delta \in S} ||Cx_\Delta||$ yields

$$\forall (\hat{x}, x) \in \mathcal{R} : ||Cx_\Delta|| \leq \epsilon.$$

For LTI-systems $\mathbf{M}$ (2.8) and $\hat{\mathbf{M}}$ (2.10), this condition can also be written as $\forall (\hat{x}, x) \in \mathcal{R} : ||y - \hat{y}|| \leq \epsilon$. Hence, since $\epsilon \geq \sup_{x_\Delta \in S} ||Cx_\Delta||$ this condition holds.

3. *Invariance.* Let $\gamma(t) \in \Gamma$ then according to Lemma 2.1 there exists a coupled distribution $\mathcal{W}$ such that with probability $1 - \delta$ the error dynamics in (2.12) can equivalently be written as (2.13). The latter implies that $(\hat{x}^+, x^+) \in \mathcal{R}$ holds with probability at least $1 - \delta$, which proves the third statement in Definition 2.4.

Items one until three prove that $\hat{\mathbf{M}}$ is (ϵ, δ)-stochastically simulated by $\mathbf{M}$ under the conditions given in Theorem 2.1. □

Comparison to available methods. As mentioned before, in Blute et al. (1997), Desharnais et al. (2004), Julius and Pappas (2009), Soudjani et al. (2015), and Haesaert and Soudjani (2020) the deviation between the abstract and original model is quantified either completely on ϵ or completely on δ by fixing $\bar{\mathcal{W}}_\gamma$. This can now be recovered by choosing a specific compensator value γ. More specifically, the deviation is completely quantified on ϵ, when $\delta = 0$. This result is obtained by choosing $\gamma = 0$, hence by choosing $\bar{\mathcal{W}}_\gamma$ such that $w - \hat{w} = 0$ holds with probability 1, we recover the results in Haesaert and Soudjani (2020). Similarly, the deviation is completely quantified on δ, when ϵ is fully defined by the grid size. This is obtained by choosing $\gamma(t) = -B_w^{-1} A x_\Delta(t)$ such that $x_\Delta(t+1) = -\beta(t)$. Hence we recover the results in Blute et al. (1997), Desharnais et al. (2004), and Soudjani et al. (2015) that also only hold for non-degenerate systems for which B_w is invertible.

Computation of deviation bounds. Consider interface function (2.11), relation (2.19), and an ellipsoidal controlled-invariant set S, that is

$$S := \left\{ (\hat{x}, x) \in \hat{\mathbb{X}} \times \mathbb{X} \mid ||x - \hat{x}||_D \leq \epsilon \right\}, \tag{2.20}$$

where $||x||_D$ denotes the weighted 2-norm, that is, $||x||_D = \sqrt{x^T D x}$ with D a symmetric positive-definite matrix $D = D^T \succ 0$. The constraints in Theorem 2.1 can now be implemented as matrix inequalities for the error dynamics (2.13) with the linear parameterization of the compensator value as an extra design variable, i.e., $\gamma = F x_\Delta$. More precisely, we can formulate an optimization problem that minimizes the deviation bound ϵ for a given bound δ subject to the existence of an (ϵ, δ)-stochastic simulation relation between models $\hat{\mathbf{M}}$ and $\mathbf{M}$ as given in Theorem 2.1. Given δ, we can compute a bound on input γ and define a suitable set Γ as

$$\gamma \in \Gamma := \left\{ \gamma \in \mathbb{R}^{n_w} \mid ||\gamma|| \leq r = |2 \operatorname{idf}\left(\frac{1-\delta}{2}\right)| \right\}, \tag{2.21}$$

which is a sphere of dimension n_w with radius r. Here idf is the inverse distribution function, i.e., the inverse of the cumulative distribution function. We will show that

given bound δ, we can optimize bound ϵ and matrix D as in (2.20) by solving the following optimization problem

$$\min_{D_{inv},L,\epsilon} \; -\frac{1}{\epsilon^2} \tag{2.22a}$$

$$\text{s.t. } D_{inv} \succ 0,$$

$$\begin{bmatrix} D_{inv} & D_{inv}C^T \\ CD_{inv} & I \end{bmatrix} \succeq 0, \qquad \textit{(ϵ-deviation)} \tag{2.22b}$$

$$\begin{bmatrix} r^2 D_{inv} & L^T \\ L & \frac{1}{\epsilon^2} I \end{bmatrix} \succeq 0, \qquad \textit{(input bound)} \tag{2.22c}$$

$$\begin{bmatrix} \lambda D_{inv} & * & * \\ 0 & (1-\lambda)\frac{1}{\epsilon^2} & * \\ AD_{inv} + B_w L & -\frac{1}{\epsilon^2}\beta_l & D_{inv} \end{bmatrix} \succeq 0 \textit{ (invariance)} \tag{2.22d}$$

where $D_{inv} = D^{-1}$, $L = FD_{inv}$, $\beta_l \in vert(\mathcal{B})$ and $l \in \{0, 1, \ldots, q\}$. This optimization problem is parameterized in λ. We say that (2.22) has a feasible solution for values of $\delta, \epsilon \geq 0$ if there exist values for λ and D_{inv}, L such that the matrix inequalities in (2.22) hold. Now, we can conclude the following.

Theorem 2.2. *Consider models* $\mathbf{M}$ *and* $\hat{\mathbf{M}}$ *and their error dynamics* (2.13)*. If a pair* $\delta, \epsilon \geq 0$ *yields a feasible solution to* (2.22)*, then* $\hat{\mathbf{M}}$ *is* (ϵ, δ)*-stochastically simulated by* $\mathbf{M}$*.*

Proof. To prove Theorem 2.2, we start with reformulating the conditions of Theorem 2.1 into implications. Next, we show that these conditions can be written as the matrix inequalities in (2.22) and that they represent a set of sufficient conditions for the (ϵ, δ)-stochastic simulation relation. The proof is structured in the same order as the conditions of Theorem 2.1.

Controlled-invariant set (first and fourth inequality constraint): In Theorem 2.1, we assume that a controlled-invariant set S is given. In (2.20) we define this ellipsoidal controlled-invariant set S, with D a symmetric positive definite matrix, $D = D^T \succ 0$. This constraint can equivalently be written as $D_{inv} = D^{-1} \succ 0$.

Following Definition 2.6, for S to be a controlled-invariant set we need to have that for all states $x_\Delta \in S$, there exists an input $\gamma = Fx_\Delta \in \Gamma$, such that for any disturbance $\beta \in \mathcal{B}$ the next state satisfies $x_\Delta^+ \in S$. To achieve this it is sufficient to require that for any $\beta \in \mathcal{B}$

$$x_\Delta^T D x_\Delta \leq \epsilon^2 \implies ((A + B_w F)x_\Delta - \beta)^T D ((A + B_w F)x_\Delta - \beta) \leq \epsilon^2. \tag{2.23}$$

This implication can equivalently be written as constraint (2.22d). First, we use the S-procedure (Boyd et al. 1994, p. 23) and Schur complement (with $D \succ 0$) and conclude that the implication in (2.23) holds for any $\beta \in \mathcal{B}$ if there exists $\lambda \geq 0$ such that for any $\beta \in \mathcal{B}$

$$\begin{bmatrix} \lambda D & 0 & (A + B_w F)^T D \\ 0 & (1-\lambda)\epsilon^2 & -\beta^T D \\ D(A + B_w F) & -D\beta & D \end{bmatrix} \succeq 0$$

holds. Performing a congruence transformation with the non-singular matrix $\begin{bmatrix} D^{-1} & 0 & 0 \\ 0 & \frac{1}{\epsilon^2} I & 0 \\ 0 & 0 & D^{-1} \end{bmatrix}$ yields

$$\begin{bmatrix} \lambda D_{inv} & 0 & D_{inv} A^T + L^T B_w^T \\ 0 & (1-\lambda)\frac{1}{\epsilon^2} & -\beta^T \\ A D_{inv} + B_w L & -\beta & D_{inv} \end{bmatrix} \succeq 0, \tag{2.24}$$

with $D_{inv} = D^{-1}$ and $L = F D_{inv}$. It is computationally impossible to verify this matrix inequality point by point for any $\beta \in \mathcal{B}$. However, if $\mathcal{B}$ is a polytope, which we represent as $\mathcal{B} = \{\beta = bz, \bar{1}^T z \leq 1, z \geq 0, \}$ with b consisting of the q vectors β_l and $\bar{1} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T$. Then we only have to consider the q vertices of $\mathcal{B}$ and we conclude that the implication holds for any $\beta \in \mathcal{B}$ if there exists $\lambda \geq 0$ such that constraint (2.22d) in (2.22) is satisfied.

Output deviation ϵ (second inequality constraint): The *ϵ-deviation* requirement $\epsilon \geq \sup_{x_\Delta \in S} ||C x_\Delta||$ in Theorem 2.1 can be simplified to the following implication

$$x_\Delta^T D x_\Delta \leq \epsilon^2 \implies x_\Delta^T C^T C x_\Delta \leq \epsilon^2. \tag{2.25}$$

This implication holds if the inequality $C^T C \preceq D$ is satisfied. Applying the Schur complement to this inequality and performing a congruence transformation with the non-singular matrix $\begin{bmatrix} D^{-1} & 0 \\ 0 & I \end{bmatrix}$ yields constraint (2.22b). Hence, if constraint (2.22b) is satisfied, the inequality $C^T C \preceq D$ holds and the bound on ϵ also holds.

Probability deviation δ (third inequality constraint): The δ-deviation requirement $\delta \geq \sup_{\gamma \in \Gamma} 1 - 2\,\mathrm{cdf}(-\frac{1}{2}||\gamma||)$ in Theorem 2.1 has been rewritten to an *input bound* on compensator value γ as (2.21). This input bound $\gamma \in \Gamma$ with $\gamma = F x_\Delta$ has to hold for all $x_\Delta \in S$, which reduces to

$$x_\Delta^T D x_\Delta \leq \epsilon^2 \implies x_\Delta^T F^T F x_\Delta \leq r^2 \tag{2.26}$$

for which $F^T F \preceq \frac{r^2}{\epsilon^2} D$ is an equivalent sufficient constraint. This inequality can be rewritten in the same way as inequality $C^T C \preceq D$ and yields constraint (2.22c), where we denoted $L = F D_{inv}$. Hence, if constraint (2.22c) is satisfied, the inequality $F^T F \preceq \frac{r^2}{\epsilon^2} D$ holds and the input bound also holds.

Concluding, if a pair $\delta, \epsilon \geq 0$ yields a feasible solution to (2.22), then the implications (2.23), (2.25), and (2.26) hold. Consequently, S is a controlled-invariant set and the bounds in Theorem 2.1 are satisfied. Based on Theorem 2.1 we conclude that $\hat{\mathbf{M}}$ is (ϵ, δ)-stochastically simulated by $\mathbf{M}$. □

Leveraging Theorem 2.2, an algorithm to search the minimal deviation ϵ can be composed as in Algorithm 2.1. The efficiency of this algorithm depends on the efficiency of the line-search algorithm for λ (c.f. line 3) and on the optimization problem (c.f. line 4). The latter problem can be solved as a semi-definite programming problem with matrix inequalities as a function of $1/\epsilon^2$.

Algorithm 2.1 Optimizing ϵ given δ such that $\hat{\mathbf{M}} \preceq_{\epsilon}^{\delta} \mathbf{M}$

1: **Input:** $\mathbf{M}, \hat{\mathbf{M}}, \delta$
2: Compute r based on δ as in (2.21)
3: **for** λ between 0 and 1 **do**
4: $D_{inv}, L, \epsilon \leftarrow$ Solve optimization problem (2.22)
5: Set $D := (D_{inv})^{-1}, F := LD$,
6: Save parameters D, F, ϵ
7: **end for**
8: Take minimal value of ϵ and corresponding matrices D and F.

Concluding, the introduction of the coupling compensator in Section 2.3 allows the use of the well-studied theory of controlled-invariant sets to quantify the deviation between the original and abstract model on bounds ϵ and δ. Furthermore, it allows an efficient computation of the deviation bounds as a set-theoretic problem. By considering an ellipsoidal controlled-invariant set, this computation can be formulated as an optimization problem constrained by parameterized matrix inequalities.

2.4b A coupling compensator for model order reduction

The provably correct design of controllers faces the curse of dimensionality. For some models, this can be mitigated by including model order reduction in the abstraction. This additional abstraction step, yielding a lower dimensional continuous-state model, decreases the dimension of the abstract model and hence decreases the computation time. In this section, we show how the coupling compensator applies to model reduction.

First, we construct a reduced-order model $\mathbf{M}_r$, based on (2.8), with state space $\mathbb{X}_r \subset \mathbb{R}^{n_r}$ with $n_r < n$. The dynamics of $\mathbf{M}_r$ are given as

$$\mathbf{M}_r : \begin{cases} x_r(t+1) & = A_r x_r(t) + B_r u_r(t) + B_{rw} w_r(t) \\ y_r(t) & = C_r x_r(t), \end{cases} \tag{2.27}$$

initialized with $x_{r,0} = x_r(0)$ and with state $x_r \in \mathbb{X}_r$, input $u_r \in \mathbb{U}$, output $y_r \in \mathbb{Y}$, and disturbance $w_r \in \mathbb{W}$ that satisfy a Gaussian distribution $w_r \sim \mathcal{N}(0, I)$.

Similarity quantification of $\mathbf{M}_r$. As in Haesaert et al. (2017b), we resolve the inputs of models $\mathbf{M}$ (2.8) and $\mathbf{M}_r$ (2.27) by choosing interface function

$$u(t) := Ru_r(t) + Qx_r(t) + K(x(t) - Px_r(t)) \tag{2.28}$$

for some matrices R, Q, K, P, such that the Sylvester equation $PA_r = AP + BQ$ and $C_r = CP$ hold. The resulting error dynamics between (2.8) and (2.27) are

$$x_{r\Delta}^{+} = \bar{A} x_{r\Delta} + \bar{B} u_r + B_w(w - w_r) + \bar{B}_w w_r, \tag{2.29}$$

where the stochastic disturbances (w_r, w) are generated by the coupled probability measure $\mathcal{W}_\gamma$ as in (2.6) and where the state $x_{r\Delta}$ and state update $x_{r\Delta}^{+}$ are the

abbreviations of $x_{r\Delta}(t) := x(t) - P\hat{x}_r(t)$ and $x_{r\Delta}(t+1)$, respectively. Furthermore, we have $\bar{A} = A + BK$, $\bar{B} = BR - PB_r$ and $\bar{B}_w = B_w - PB_{rw}$. The term $(w - w_r)$ can now be used as a *coupling compensator term.*

Unlike existing work Haesaert et al. (2017a) and Haesaert et al. (2017b), we now use an approach similar to the one used before and substitute $w_r = w_\gamma - \gamma_r$ for w_r. Subsequently, we choose $\mathcal{W}_\gamma$ again as the coupling that maximizes the probability of event $w - w_\gamma = 0$. The error dynamics conditioned on this event reduce to

$$x^+_{r\Delta} = \bar{A}x_{r\Delta} + \bar{B}u_r + B_w\gamma_r + \bar{B}_w w_r. \tag{2.30}$$

Lemma 2.1 still applies and can be used to compute $1 - \delta$. If $\bar{B}_w = 0$ then (2.30) reduces to a set-theoretic control problem. In contrast, if this does not hold then by truncating the stochastic influence w_r, the error dynamics are still bounded and the probability δ can be modified to $\delta_r = \delta + \delta_{trunc}$, where δ_{trunc} is the error introduced by truncating w_r to the bounded set W. We consider the resulting error dynamics (2.30) as a system with constrained input γ_r and bounded disturbance $z = \bar{B}u_r + B_w w_r$. This is very similar to the error dynamics in (2.13), however, now instead of bounded disturbance β we have $z \in Z = \bar{B}\mathbb{U} + \bar{B}_w W$, with W the set of the truncated disturbance w_r. If we now consider simulation relation

$$\mathcal{R}_{MOR} = \{(x_r, x) \in \mathbb{X}_r \times \mathbb{X} \mid ||x - Px_r||_{D_r} \leq \epsilon_r\} \tag{2.31}$$

then we can recover the results in Theorem 2.1 to achieve an (ϵ_r, δ_r)-simulation relation between $\mathbf{M}_r$ and $\mathbf{M}$.

Computation of deviation bounds. Consider interface function (2.28) and simulation relation (2.31). Given bound δ_r and matrices P, Q, R, we can optimize bound ϵ_r and matrix D_r as in (2.31) by solving an optimization problem similar to (2.22). Since model order reduction influences the error dynamics, the invariance constraint in (2.22d) has to be altered to

$$\begin{bmatrix} \lambda D_{r,inv} & * & * \\ 0 & (1-\lambda)\frac{1}{\epsilon_r^2} & * \\ AD_{r,inv} + BE + B_w L & \frac{1}{\epsilon_r^2} z_l & D_{r,inv} \end{bmatrix} \succeq 0, \tag{2.32}$$

where $E = KD_{r,inv}$ and $z_l \in vert(Z)$. To make sure that the bound $u \in \mathbb{U}$ is satisfied an additional constraint can be formulated for matrix K in the same way as the matrix inequality for the input bound in (2.22c).

Similarity quantification between $\mathbf{M}$ and $\hat{\mathbf{M}}_r$. The finite-state abstract model $\hat{\mathbf{M}}_r$ of $\mathbf{M}_r$ (2.27) will now be substantially smaller than the finite-state abstraction of $\mathbf{M}$. Given the (ϵ_r, δ_r)-simulation relation between $\mathbf{M}_r$ and $\mathbf{M}$, the relation between $\hat{\mathbf{M}}_r$ and $\mathbf{M}$ can be computed by considering the relation between $\hat{\mathbf{M}}_r$ and $\mathbf{M}_r$. More precisely, we can follow Section 2.4 and compute a pair $(\epsilon_{abs}, \delta_{abs})$ that guarantees that $\hat{\mathbf{M}}_r$ is $(\epsilon_{abs}, \delta_{abs})$-stochastically simulated by $\mathbf{M}_r$. Following Theorem 5 in Haesaert et al. (2017b) on transitivity of $\preceq^{\delta}_{\epsilon}$ we have that if $\mathbf{M} \preceq^{\delta_r}_{\epsilon_r} \mathbf{M}_r$ and $\mathbf{M}_r \preceq^{\delta_{abs}}_{\epsilon_{abs}} \hat{\mathbf{M}}_r$ both hold, the simulation relation $\mathbf{M} \preceq^{\delta_{abs}+\delta_r}_{\epsilon_{abs}+\epsilon_r} \hat{\mathbf{M}}_r$ holds as well.

2.5 A piecewise-affine abstraction for nonlinear stochastic sytems

In this section, we describe how to apply the coupling compensator as introduced in Section 2.3 to nonlinear systems. To this end, we follow a similar approach as in Section 2.4. First, we give a brief introduction and a comprehensive overview of the relevant literature.

Introduction and literature. Safety-critical systems are difficult to analyze and verify as they evolve over continuous spaces in a stochastic and generally *nonlinear fashion*. Therefore, we need methods that can handle simultaneously complex safety-critical requirements, large-scale continuous states, and stochastic and *nonlinear state evolutions*.

Synthesizing a provably correct controller that guarantees the satisfaction of temporal logic specifications for nonlinear stochastic systems remains a very challenging problem and the number of methods that exist is very limited. More specifically, methods either focus on a specific type of specification (Jagtap et al. 2020; Nejati et al. 2020), use slope restrictions on the nonlinearity (pseudo-linearity) of the systems while requiring strict dissipativity requirements (Lavaei et al. 2019; Zhong et al. 2023a) or consider a stochastic disturbance with a bounded support (Majumdar et al. 2020; Majumdar et al. 2021).

When focusing on local behavior, many nonlinear systems behave almost linearly. Therefore, a widely adopted approach in classical control (Johansson 1999; Rodrigues and How 2003; Rodrigues and Boyd 2005) and for the verification of nonlinear *deterministic* systems (Asarin et al. 2007) is performing a *piecewise-affine approximation* of the nonlinear system. In this section, we focus on the following.

Subquestion

How to leverage piecewise-affine approximations for the provably correct control design of nonlinear stochastic systems?

As before, the similarity or deviations in probability and output of stochastic systems is expressed using approximate simulation relations (Haesaert et al. 2017b). Together with Haesaert and Soudjani (2020), this allows us to handle syntactically co-safe linear temporal logic specifications that are unbounded in time. In this section, we develop tailored methods for the provably correct control design of *nonlinear stochastic systems*. More specifically, the conditions under which simulation relations can be established exist for nonlinear stochastic systems (Haesaert et al. 2017b; Haesaert and Soudjani 2020). However, the main challenge is to find such relations with efficient computation methods, where existing methods can only handle nonlinear systems with bounded slope (Lavaei et al. 2019; Zhong et al. 2023a). In this work, we perform a piecewise-affine approximation step, such that we can develop a method like in Section 2.4. By doing so, the computational cost can be managed and we enable an efficient and accurate implementation.

Model. Consider a system whose behavior can be modeled by a discrete-time nonlinear stochastic difference equation

$$\mathbf{M}: \begin{cases} x(t+1) & = f(x(t)) + Bu(t) + B_w w(t) \\ y_t & = Cx(t), \quad \forall t \in \{0,1,2,\dots\}, \end{cases} \tag{2.33}$$

with state $x \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$, input $u \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$ disturbance $w \in \mathbb{W} \subseteq \mathbb{R}^{n_w}$, and output $y(t) \in \mathbb{Y} \subseteq \mathbb{R}^{n_y}$. Furthermore, we have matrices $B \in \mathbb{R}^{n_x \times n_u}, B_w \in \mathbb{R}^{n_x \times n_w}, C \in \mathbb{R}^{n_y \times n_x}$ and the nonlinear function $f : \mathbb{X} \to \mathbb{X}$ is assumed to be sufficiently smooth. The disturbance $w(t)$ is an independent and identically distributed (i.i.d.) noise signal with realizations $w \sim \mathbb{P}_w$ and the system is initialized at $x_0 \in \mathbb{X}$.

Remark 2.5. For easier exposition of the results, we assume that the output and input enter linearly. Systems with nonlinear terms $g(x(t))u(t)$ and $h(x(t))$ instead of respectively $Bu(t)$ and $Cx(t)$, can also be handled through piecewise-affine approximations.

2.5a Piecewise-affine abstraction

In this section, we discuss the first step in designing a provably correct controller, namely constructing a piecewise-affine abstraction of the nonlinear system in (2.33).

Local affine approximation of $f(x(t))$. To handle the non-linearity of $f(x(t))$ in (2.33), we start considering $f(x)$ a static function over a bounded domain $x \in \mathbb{G}$. Now, we can use affine functions to locally approximate $f(x)$ in this bounded set $\mathbb{G} \subseteq \mathbb{X}$. To this end, we use Taylor's Theorem (Adams and Essex 2009, Sec. 4.10), which for a one-dimensional (1D) function $f : \mathbb{R} \to \mathbb{R}$ states the following about its accuracy.

Proposition 2.1. *Suppose that $f : \mathbb{R} \to \mathbb{R}$ is defined on a closed interval $\mathbb{G}$ with $x, \nu \in \mathbb{G}$ and its $N_T + 1$st order derivative $f^{(N_T+1)}$ exists on the same interval. If $f_{N_T}(x)$ is the N_Tth-order Taylor polynomial at the point ν, that is*

$$f_{N_T}(x) = f(\nu) + \textstyle\sum_{l=1}^{N_T} \frac{f^{(l)}(\nu)}{l!}(x-\nu)^l, \tag{2.34}$$

then for each $x \in \mathbb{G}$, there exists a ζ on the interval between ν and x, such that the remainder $R_{N_T}(x) = f(x) - f_{N_T}(x)$ in the approximation $f(x) \approx f_{N_T}(x)$ is given by

$$R_{N_T}(x) = \tfrac{f^{(N_T+1)}(\zeta)}{(N_T+1)!}(x-\nu)^{N_T+1}. \tag{2.35}$$

The proof is given in Adams and Essex (2009) and extended to multivariable functions in Lou (2021, Sec. 2.4). Using Taylor's inequality (Weisstein 2014), an upper bound of the remainder (2.35) can be found,

$$|R_{N_T}(x)| \leq \sup_{\zeta \in \mathbb{G}} \left(\left| \tfrac{f^{N_T+1}(\zeta)}{(N_T+1)!} \right| \right) \cdot |x-\nu|^{N_T+1}, \tag{2.36}$$

which holds for all $x \in \mathbb{G}$. The derivation of this upper bound and its extension to multivariate functions is given in Lou (2021, Sec. 2.4). By taking the supremum

over $x \in \mathbb{G}$, an upper bound on the approximation error of the N_Tth-degree Taylor polynomial, denoted by remainder R_{N_T} can be computed. To linearize the function $f(x(t))$ in (2.33) we consider $N_T = 1$, yielding the first-order Taylor polynomial as affine function

$$f(x(t)) \approx f_1(x(t)) = Ax(t) + a \text{ for } x \in \mathbb{G}, \tag{2.37}$$

with matrix $A = \nabla f(\nu)$ and vector $a = f(\nu) - \nabla f(\nu)\nu$. Define the bounded difference between $f(x(t))$ and its affine approximation (2.37) by $\kappa(x) = f(x) - f_1(x) = R_1(x)$ as in (2.35). Associate to this difference, bounded set $\mathcal{K} \subset \mathbb{R}^{n_x}$ computed using (2.36) in all directions, that is $\mathcal{K} = \{x \in \mathbb{G} \mid (2.36) \text{ for } N_T = 1\}$. Now, we get the following.

Theorem 2.3. *Given a nonlinear function $f(x(t))$ that is sufficiently smooth and its first order Taylor polynomial approximation (2.37). Then for all $x \in \mathbb{G}$, there exists a bounded vector $\kappa \in \mathcal{K}$, such that $f(x) = Ax + a + \kappa(x)$.*

The proof of this theorem follows from the extension of Proposition 2.1 to higher-dimensional functions as in Lou (2021). For ease of notation, we denote the state-dependent error $\kappa(x(t))$ as $\kappa(x)$.

Piecewise-affine finite-state abstraction. To construct such an abstraction, we need two different partitionings of the state space. A coarse partitioning to construct the piecewise-affine approximation of the nonlinear dynamics and a fine grid to get a finite-state approximation of the affine dynamics. To obtain the coarse partitions, we partition the state space $\mathbb{X}$ with polytopic cells $\hat{P}_i$ with $i \in \{1, \ldots, N_P\}$, such that it covers the complete state space, that is $\bigcup_i \hat{P}_i = \mathbb{X}$ and such that the partitions do not overlap $\hat{P}_i \cap \hat{P}_k = \emptyset$ for $i \neq k$. Similarly, to obtain the fine grid we grid the state space $\mathbb{X}$ in a finite number of regions $\mathbb{A}_j \subset \mathbb{X}$, such that $\bigcup_j \mathbb{A}_j = \mathbb{X}$ and $\mathbb{A}_j \cap \mathbb{A}_l = \emptyset$ for $j \neq l$ hold. In each region, a representative point $\hat{X}_j \in \mathbb{A}_j$ is chosen. Together, these points make up the set of abstract states, $\hat{x} \in \hat{\mathbb{X}} = \left\{\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_{N_A}\right\}$. Such a partitioned and gridded two-dimensional state space is shown in the top illustration of Figure 2.3. Note that for LTI systems as in Section 2.4, we only need the fine grid.

After performing the affine approximation for each of the partitions $\hat{P}_i \subseteq \mathbb{G}$, we select a finite number of inputs from $\mathbb{U}$ to form the abstract input space $\hat{\mathbb{U}}$. Now, we can approximate the behavior of the affine approximation of the nonlinear dynamics by a finite-state abstract system. To this end, consider the operator $\Pi : \mathbb{X} \to \hat{\mathbb{X}}$ that maps states from the original state space to the abstract state space. Then in region $\hat{P}_i$ the dynamics of the abstract system equal

$$\hat{x}(t+1) = \Pi(A\hat{x}(t) + B\hat{u}(t) + a + B_w\hat{w}(t)), \tag{2.38}$$

with states $\hat{x} \in \hat{P}_i \subset \hat{\mathbb{X}}$, initial state $\hat{x}_0 = \Pi(x_0)$, inputs $\hat{u} \in \hat{\mathbb{U}}$, and disturbances $\hat{w} \in \mathbb{W}$ with realizations $\hat{w} \sim \mathbb{P}_{\hat{w}}$. Next, we introduce a bounded vector $\beta \in \mathcal{B} \subset \mathbb{X}$, that pushes the state to its representative point. Then, with a slight abuse of notation[3] the state dynamics of the local abstract system (2.38) for $\hat{x} \in \hat{P}_i$ satisfy

[3]Here, the Minkowski sum of the two sets is neglected.

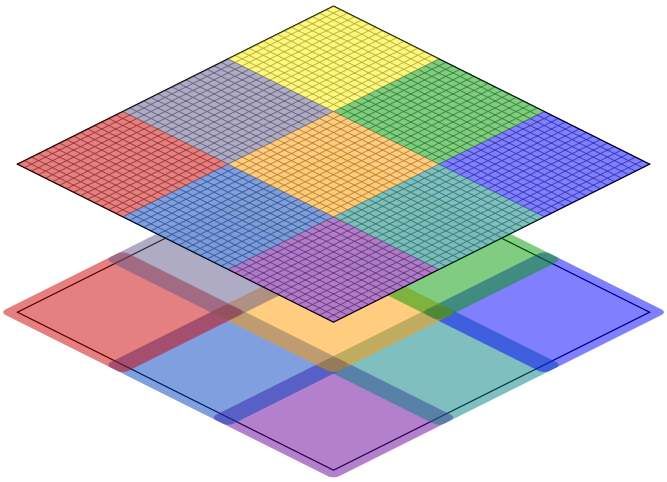


Figure 2.3: Partitioning and gridding of a two-dimensional state space $\mathbb{X}$. The small grid cells in the top plane are the regions $\mathbb{A}_j$, while the colored squares are partitions $\hat{P}_i$ (top) and P_i (bottom).

$\hat{x}(t+1) \in A\hat{x}(t) + B\hat{u}(t) + a + \hat{w}(t) + \mathcal{B}$. More precisely, there exist $\beta \in \mathcal{B}$, such that $\hat{x}(t+1) = A\hat{x}(t) + B\hat{u}(t) + a + \hat{w}(t) + \beta(t)$. Now, we can write the state dynamics of the local abstract system as an affine system described by

$$\hat{x}(t+1) = A\hat{x}(t) + B\hat{u}(t) + a + B_w\hat{w}(t) + \beta(t) \text{ for } \hat{x} \in \hat{P}_i. \tag{2.39}$$

To define a piecewise-affine finite-state abstraction, we translate the abstract system (2.39) that locally approximates the nonlinear system (2.33) to a piecewise-affine system that approximates the complete nonlinear system as

$$\hat{\mathbf{M}} : \begin{cases} \hat{x}(t+1) = A_i\hat{x}(t) + B\hat{u}(t) + a_i + B_w\hat{w}(t) + \beta(t) \text{ for } \hat{x} \in \hat{P}_i \\ \hat{y} = C\hat{x}(t), \end{cases} \tag{2.40}$$

with states $\hat{x} \in \hat{\mathbb{X}} \subset \mathbb{X}$, initial state $\hat{x}_0$, inputs $\hat{u} \in \hat{\mathbb{U}}$, and disturbance $\hat{w} \in \mathbb{W}$.

Concluding, we have constructed a piecewise-affine finite-state system that approximates the behavior of a nonlinear continuous-state system (2.33). This piecewise-affine system consists of local affine dynamics defined over partitions $\hat{P}_i$.

2.5b Piecewise stochastic simulation relation

In this section, we discuss how to quantify the difference between the original nonlinear stochastic model and the abstract model obtained via piecewise affine approximations.

Similarity quantification. To quantify the similarity between the nonlinear stochastic difference equation $\mathbf{M}$ in (2.33) and its finite-state abstraction $\hat{\mathbf{M}}$ in (2.40), we follow the method as described in Section 2.3. Again, we start by defining

a metric for the composed model as in Definition 2.3. However, in this case, we adapt the metric such that it yields a local state-based deviation bound denoted by $\boldsymbol{\delta}(\cdot)$. The simulation relation similar to Definition 2.4 between the *nonlinear* stochastic models $\mathbf{M}$ (2.33) and its abstraction $\hat{\mathbf{M}}$ (2.40) is defined as follows.

Definition 2.7. ($(\epsilon,\boldsymbol{\delta})$-stochastic simulation relation): *Let stochastic models $\mathbf{M}$ and $\hat{\mathbf{M}}$ with metric output space $(\mathbb{Y}, \boldsymbol{d}_{\mathbb{Y}})$, interface function $\mathcal{U}_v$ (2.4), and stochastic kernel $\bar{\mathcal{W}}$ (2.6) be given. If there exists a measurable relation $\mathcal{R} \subseteq \hat{\mathbb{X}} \times \mathbb{X}$, with $(\hat{x}_0, x_0) \in \mathcal{R}$, and such that*

1. *$\forall (\hat{x}, x) \in \mathcal{R} : \boldsymbol{d}_{\mathbb{Y}}(\hat{y}, y) \leq \epsilon$, and*
2. *$\forall (\hat{x}, x) \in \mathcal{R}, \forall \hat{u} \in \hat{\mathbb{U}} : (\hat{x}^+, x^+) \in \mathcal{R}$ holds with probability at least $1 - \boldsymbol{\delta}(\hat{x})$, with $\boldsymbol{\delta} : \hat{\mathbb{X}} \to [0, 1]$.*

then $\hat{\mathbf{M}}$ is $(\epsilon, \boldsymbol{\delta})$-stochastically simulated by $\mathbf{M}$, and this simulation relation is denoted as $\hat{\mathbf{M}} \preceq^{\boldsymbol{\delta}}_{\epsilon} \mathbf{M}$.

As in Section 2.3, to achieve a simulation relation over the composed model, we first couple the inputs u and $\hat{u}$ by using an interface function (2.11). This function assigns input u to the abstract input $\hat{u}$ given the states $\hat{x}$ and x of the abstract and original model, respectively. Next, we couple disturbances $w \sim \mathbb{P}_w$ and $\hat{w} \sim \mathbb{P}_{\hat{w}}$ by following Definition 2.2. We can trivially extend this to Borel measurable stochastic coupling kernels as in (2.6).

For such an $(\epsilon,\boldsymbol{\delta})$-stochastic simulation relation, we refer to ϵ as the (metric) output deviation and to $\boldsymbol{\delta}$ as the probabilistic or stochastic deviation function. Note that unlike in Definition 2.4 the stochastic deviation is not uniform for the whole state space. Instead, it is introduced as a function $\boldsymbol{\delta} : \hat{\mathbb{X}} \to [0, 1]$ that depends on the abstract state $\hat{x}$. If $\boldsymbol{\delta}(\hat{x})$ is a piecewise constant function, then we refer to the simulation relation as a *piecewise stochastic simulation relation.*

Through Definition 2.7, we have defined a measure to quantify the difference between two models on a global level, that is, over the full state space. The question is now how we can compute it based on the given piecewise-affine structure of the abstractions.

Piecewise similarity quantification. As before, we consider a simulation relation given as

$$\mathcal{R} := \left\{ (\hat{x}, x) \in \hat{\mathbb{X}} \times \mathbb{X} \mid ||x - \hat{x}||_D \leq \epsilon \right\}, \tag{2.41}$$

with a suitable weighting matrix D. It can be seen that

$$C^\top C \preceq D \tag{2.42}$$

implies that the first condition of Definition 2.7 is satisfied. Next, we use relation (2.41) to show that a global $(\epsilon,\boldsymbol{\delta})$-stochastic simulation relation can be computed with a piecewise constant probability deviation function $\boldsymbol{\delta} : \hat{\mathbb{X}} \to [0, 1]$ defined on state partition $\hat{P}_i$ as $\boldsymbol{\delta}(\hat{x}) = \delta_i$ if $\hat{x} \in \hat{P}_i$. The function $\boldsymbol{\delta}$ assigns a constant local

probability deviation to each partition in the abstract state space $\hat{\mathbb{X}}$ based on a local similarity quantification derived using the local error dynamics.

Local stochastic error dynamics. Consider a local interface function $u(t) = \mathcal{U}_{vi}(\hat{u}(t), \hat{x}(t), x(t))$ as

$$u(t) = \hat{u}(t) + K_{f,i}(x(t) - \hat{x}(t)), \tag{2.43}$$

with feedback matrix $K_{f,i} \in \mathbb{R}^{n_u \times n_x}$ and a local stochastic kernel $\bar{\mathcal{W}}_i$, assigning to each $(\hat{u}, \hat{x}, x)$ a probability measure

$$\bar{\mathcal{W}}_i : \hat{\mathbb{U}} \times \hat{P}_i \times \mathbb{X} \to \mathcal{P}(\mathbb{W}^2). \tag{2.44}$$

For $\hat{x} \in \hat{P}_i$, we have $\hat{x}(t+1) = A_i\hat{x}(t) + B\hat{u}(t) + a_i + \hat{w}(t) + \beta(t)$ and if $\|x - \hat{x}\|_D \leq \epsilon$ holds, then there exists a $\kappa(x)$ such that

$$x(t+1) = A_i x(t) + Bu(t) + a_i + B_w w(t) + \kappa(x) \text{ with } \kappa(x) \in \mathcal{K}_i.$$

Given that $x(t)$ belongs to P_i defined as

$$P_i := \{x \ \in \mathbb{X} \mid \exists \hat{x} \in \hat{P}_i : \|x - \hat{x}\|_D \leq \epsilon\} \tag{2.45}$$

and shown for a 2D state space in Figure 2.3. For $\hat{x}(t) \in \hat{P}_i$ and $x(t) \in P_i$, we get the error dynamics $x_\Delta(t) := x(t) - \hat{x}(t)$ equal to

$$x_\Delta^+ = (A_i + BK_{f,i})x_\Delta(t) + B_w(w(t) - \hat{w}(t)) + \kappa(x) - \beta(t) \tag{2.46}$$

with ${x_\Delta}^+$ and x_Δ abbreviating respectively $x_\Delta(t+1)$ and $x_\Delta(t)$, and with $\kappa \in \mathcal{K}_i$, $\beta \in \mathcal{B}$, and $(\hat{w}(t), w(t)) \sim \mathcal{W}_i$.

Local coupling and interface functions with $\boldsymbol{\delta} = \delta_i$. Following Section 2.4, we make sure that the second condition of Definition 2.7 is satisfied by finding a global invariant set $\{x_\Delta \mid \|x_\Delta\|_D \leq \epsilon\}$ parameterized with a global D for the error dynamics (2.46). Together with D, we have to compute an optimal local interface function (2.43) and coupling (2.44) for all partitions $\hat{P}_i$, with $i \in \{1, \dots N_p\}$. More precisely, we design $\mathcal{W}_i$ and $K_{f,i}$ such that the probability $1 - \delta_i$ with which $\|{x_\Delta}^+\|_D \leq \epsilon$ holds is maximized. To this end, consider local coupling $\hat{w} = w + F_i(x - \hat{x})$ that holds with probability $1 - \delta_i$. The coupling term F_i introduced in Section 2.4 reduces the complexity of the design of $\mathcal{W}_i$ as it allows to write the design problem as a set of implications or matrix inequalities. That is, a relation between this term and the probability deviation δ_i can be derived as upper bound

$$||F_i(x - \hat{x})|| \leq r_i := \left|2 \operatorname{idf}\left(\tfrac{1-\delta_i}{2}\right)\right|. \tag{2.47}$$

Here, idf denotes the inverse distribution function of a Gaussian distribution $\mathcal{N}(0, I)$.

As concluded from the error dynamics in (2.46), together with the coupling, the interface function can be used to further compensate for the error in the state by computing a feedback-term $K_{f,i}$. To satisfy the bound $u \in \mathbb{U}$ we shrink $\hat{\mathbb{U}}$ by α, such that $\hat{\mathbb{U}} \subset \alpha\mathbb{U}$ and we compute u_u, such that

$$||K_{f,i}(x - \hat{x})|| \leq u_u \tag{2.48}$$

implies $K_{f,i}(x - \hat{x}) \in (1 - \alpha)\mathbb{U}$. Taken together, we conclude the following.

Lemma 2.2 (Piecewise requirements)**.** *Consider stochastic models* $\mathbf{M}$ (2.33) *and* $\hat{\mathbf{M}}$ (2.40) *for which a simulation relation* $\mathcal{R}$ (2.41) *with weighting matrix* D *satisfying* (2.42) *is given. If there exist matrices* F_i, *and* $K_{f,i}$ *such that for a given* $\boldsymbol{\delta}(\hat{x}) = \delta_i$ *if* $\hat{x} \in \hat{P}_i$, *the following implications are satisfied* $\forall x_\Delta$*:*

$$x_\Delta^\top D x_\Delta \leq \epsilon^2 \implies \begin{cases} x_\Delta^\top F_i^\top F_i x_\Delta & \leq r_i^2 \\ x_\Delta^\top K_{f,i}^\top K_{f,i} x_\Delta & \leq u_u^2 \\ x_{\Delta t+1}^\top D x_{\Delta t+1} & \leq \epsilon^2, \end{cases} \tag{2.49}$$

with $x_{\Delta t+1}$ *in* (2.46) *and* r_i *in* (2.47)*, then there exists coupling kernels* $\bar{\mathcal{W}}_i$ *and interfaces* $\mathcal{U}_{vi}$ *such that*

$$\forall (\hat{x}, x) \in \hat{P}_i \times \mathbb{X}, \forall \hat{u} \in \hat{\mathbb{U}} : (\hat{x}^+, x^+) \in \mathcal{R} \tag{2.50}$$

holds with probability $1 - \delta_i$ *for all* $\hat{P}_i$*, with* $i \in \{1, \dots N_p\}$ *and with* $\bigcup_i \hat{P}_i = \hat{\mathbb{X}}$.

Proof. It can readily be seen that the first and second implication in (2.49) are sufficient conditions for the bounds on respectively the *coupling compensator term* (2.47) and the feedback-term (2.48). Assume that bounded sets $\mathcal{K}_i$ are given and define the sets $S_i := \left\{(\hat{x}, x) \in \hat{P}_i \times \mathbb{X} \mid ||x - \hat{x}||_D \leq \epsilon\right\}$. The last implication in (2.49) is a sufficient condition for sets S_i to be controlled invariant sets according to Definition 2.6 with disturbance $\beta + \kappa \in \mathcal{B} \oplus \mathcal{K}_i$. If the implications in (2.49) hold, then the bounds in Theorem 2.1 are satisfied and S_i are controlled-invariant sets. Following the proofs of Theorem 2.1 and Lemma 2.2 we can conclude that this implies the existence of coupling kernels $\bar{\mathcal{W}}_i$ and interfaces $\mathcal{U}_{vi}$ such that Lemma 2.2 holds. An algorithm to obtain matrix D and bounded sets $\mathcal{K}_i$ is explained in Appendix 2.A. □

Remark 2.6. Note that locally (consider only one partition, i.e. $N_P = 1$), this approach through piecewise-affine abstractions is equivalent to the approach for LTI systems (c.f. Section 2.4) up to an affine term that disappears when computing the error dynamics.

From local to piecewise similarity quantification. To obtain a global similarity quantification, we define a piecewise stochastic kernel $\bar{\mathcal{W}}$ and a piecewise interface function $\mathcal{U}_v$. Since $\hat{\mathbb{U}} \times \hat{P}_i \times \mathbb{X}$ for $i \in \{1, \dots, N_P\}$ is a partitioning of $\hat{\mathbb{U}} \times \hat{\mathbb{X}} \times \mathbb{X}$ we use the local stochastic coupling kernel $\bar{\mathcal{W}}_i : \hat{\mathbb{U}} \times \hat{P}_i \times \mathbb{X} \to \mathcal{P}(\mathbb{W}^2)$ to compute the piecewise stochastic coupling kernel $\bar{\mathcal{W}} : \hat{\mathbb{U}} \times \hat{\mathbb{X}} \times \mathbb{X} \to \mathcal{P}(\mathbb{W}^2)$ as

$$\bar{\mathcal{W}}(\cdot \mid \hat{u}, \hat{x}, x) = \bar{\mathcal{W}}_i(\cdot \mid \hat{u}, \hat{x}, x) \text{ if } \hat{x} \in \hat{P}_i. \tag{2.51}$$

Similarly, the interface function can be composed as

$$\mathcal{U}_v(\hat{u}(t), \hat{x}(t), x(t)) = \mathcal{U}_{vi}(\hat{u}(t), \hat{x}(t), x(t)) \text{ if } \hat{x} \in \hat{P}_i. \tag{2.52}$$

We can now show that these functions constitute to an $(\epsilon, \boldsymbol{\delta})$-stochastic simulation relation for simulation relation (2.41).

Theorem 2.4 (Piecewise stochastic similarity)**.** *Let stochastic models* $\mathbf{M}$ (2.33) *and* $\hat{\mathbf{M}}$ (2.40) *be given. Then the interface function* $\mathcal{U}_v$ (2.52) *and the global Borel measurable stochastic kernel* $\bar{\mathcal{W}}$ (2.51) *computed for the simulation relation* (2.41) *based on* (2.49) *define an* (ϵ,$\boldsymbol{\delta}$)*-stochastic simulation relation in a piecewise manner as given in Definition 2.7 if*

- *it holds that* $(\hat{x}_0, x_0) \in \mathcal{R}$*, and if*
- *the simulation relation satisfies matrix inequality* (2.42)*.*

Proof. The proof builds on Lemma 2.2 and can be sketched as follows. The first condition of Definition 2.7 holds by choosing matrix D, such that (2.42) holds. This is proven in Section 2.4. Lemma 2.2 shows that if (2.49) is satisfied then a local stochastic kernel $\bar{\mathcal{W}}_i$ as in (2.44) exists, such that (2.50) holds with probability $1 - \delta_i$. By choosing the interface function $\mathcal{U}_v$ as (2.52) and the global stochastic kernel as in (2.51) the second condition in Definition 2.7 is satisfied. □

2.6 Temporal logic control

In this section, we discuss how to compute the satisfaction probability of (infinite-horizon) temporal logic specifications based on the dynamic programming mappings from Haesaert and Soudjani (2020). The method described next applies to both linear and nonlinear systems.

In correct-by-design control synthesis, an scLTL specification ϕ can be written as a *deterministic finite-state automaton* (DFA), characterized by the tuple $\mathcal{A}_\phi = (Q, q_0, \Sigma_\mathcal{A}, \tau_\mathcal{A}, Q_f)$ (Belta et al. 2017). Here, the set of states is denoted by Q with initial state q_0. The input alphabet and transition function are respectively denoted by $\Sigma_\mathcal{A} = 2^{\mathrm{AP}}$ and $\tau_\mathcal{A} : Q \times \Sigma_\mathcal{A} \to Q$. Finally, Q_f denotes the set of accepting states. The word $\boldsymbol{\pi}$ satisfies the specification ϕ, that is $\boldsymbol{\pi} \models \phi$, if the word $\boldsymbol{\pi}$ is *accepted* by the DFA $\mathcal{A}_\phi$. This means that there exists a trajectory $q_0, q_1, \ldots, q_f$ with $q_f \in Q_f$ starting at q_0 and evolving according to $q_{t+1} = \tau_\mathcal{A}(q_t, \pi_t)$. By analyzing the product composition between the system $\mathbf{M}$ and the specification DFA $\mathcal{A}_\phi$, denoted as $\mathbf{M} \otimes \mathcal{A}_\phi$, we can compute the satisfaction probability. The composition $\mathbf{M} \otimes \mathcal{A}_\phi$ consists of states $(x_t, q_t) \in \mathbb{X} \times Q$. For a given input $u_t = u(t)$, it evolves from (x_t, q_t) to (x_{t+1}, q_{t+1}) by following the stochastic transition from $x_t = x(t)$ to $x_{t+1} = x(t+1)$ in (2.33) and from q_t to $q_{t+1} = \tau_\mathcal{A}(q_t, L(Cx_t))$. Hence, computing the satisfaction probability is equivalent to solving a reachability problem of the composition $\mathbf{M} \otimes \mathcal{A}_\phi$, which can be written as a dynamic program. With a slight abuse of notation, we refer to the stationary policy of this composed system as $\boldsymbol{\mu} : \hat{\mathbb{X}} \times Q \to \hat{\mathbb{U}}$.

We use the abstract model to compute the satisfaction probability since this is not possible for the original model due to its continuous states. The satisfaction probability with policy $\boldsymbol{\mu}$ in time horizon $[1, \ldots, N]$ is expressed by the value function $V_N^{\boldsymbol{\mu}}(\hat{x}, q) : \hat{\mathbb{X}} \times Q \to [0, 1]$, which is equivalent to the probability that

the trajectory starting at (x, q) and generated by applying $\boldsymbol{\mu}$ to $\mathbf{M} \otimes \mathcal{A}_\phi$ reaches the target set Q_f within this time horizon. The value function is defined as $V_N^{\boldsymbol{\mu}}(\hat{x}, q) := \mathbb{E}_{\boldsymbol{\mu}}\left(\max_{0\leq t\leq N} \mathbf{1}_{Q_f}(q_t)|(\hat{x}_0, q_0)\right)$, with indicator function $\mathbf{1}_{Q_f}$ equal to 1 if $q \in Q_f$ and 0 otherwise. The value function can also be computed recursively for a policy $\boldsymbol{\mu}_i = (\mu_{i+1}, \ldots, \mu_N)$ with horizon $N - i$ as $V_{N-k+1}^{\boldsymbol{\mu}_{k-1}}(\hat{x}, q) = \mathbf{T}^{\mu_k}(V_{N-k}^{\boldsymbol{\mu}_k})(\hat{x}, q)$, initialized with $V_0 \equiv 0$. Here, operator $\mathbf{T}^{\mu_k}(\cdot)$ is defined as $\mathbf{T}^{\mu_k}(V)(\hat{x}, q) := \mathbb{E}_{\mu_k}(\max\left\{\mathbf{1}_{Q_f}(q^+), V(\hat{x}^+, q^+)\right\})$, with DFA transitions $q^+ = \tau_{\mathcal{A}_\phi}(q, L(Cx^+))$. For a stationary policy $\boldsymbol{\mu}$, the infinite-horizon value function is computed as $V_\infty^{\boldsymbol{\mu}} = \lim_{N\to\infty}(\mathbf{T}^{\boldsymbol{\mu}})^N V_0$ initialized with $V_0 \equiv 0$. The policy-optimal converged value function V_∞^* is computed with the operator $\mathbf{T}^*(\cdot) := \sup_\mu \mathbf{T}^\mu(\cdot)$. The corresponding satisfaction probability can now be computed as $\mathbb{P}^{\boldsymbol{\mu}} := \max(\mathbf{1}_{Q_f}(\bar{q}_0, V_\infty^*(\hat{x}_0, \bar{q}_0)))$ with $\bar{q}_0 = \tau(q_0, L(Cx_0))$ and with $\hat{x}_0 = \Pi(x_0)$.

To cope with the output deviation ϵ and with probability deviations described by the function $\boldsymbol{\delta}(\hat{x})$, we define a robust dynamic programming mapping similar to Haesaert et al. (2017b), as

$$\mathbf{T}_{\epsilon,\boldsymbol{\delta}}^{\mu_k}(V)(\hat{x}, q) := \mathbf{L}\left(\mathbb{E}_\mu(\min_{q^+\in Q^+} \max\left\{\mathbf{1}_{Q_f}(q^+), V(\hat{x}^+, q^+)\right\}) - \boldsymbol{\delta}(\hat{x})\right), \tag{2.53}$$

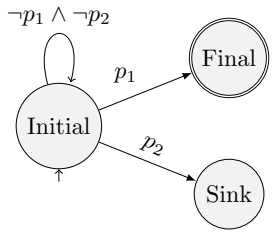
with $\mathbf{L} : \mathbb{R} \to [0, 1]$ a truncation function $\mathbf{L}(\cdot) := \min(1, \max(0, \cdot))$ and with $Q^+(q, \hat{y}^+) := \{\tau_{\mathcal{A}}(q, L(y^+)) \mid ||y^+ - \hat{y}^+|| \leq \epsilon\}$. We can now compute the robust satisfaction probability by considering the first time instance based on x_0, that is, $\mathbb{R}^{\boldsymbol{\mu}} := \max(\mathbf{1}_{Q_f}(\bar{q}_0), V_\infty^{\boldsymbol{\mu}}(\hat{x}_0, \bar{q}_0))$ with $\bar{q}_0 = \tau_{\mathcal{A}}(q_0, L(Cx_0))$ and with $\hat{x}_0 = \Pi(x_0)$. This probability is robust since it gives a lower bound on the probability in (2.2), i.e., $\mathbb{R}_{\epsilon,\boldsymbol{\delta}}^{\boldsymbol{\mu}}(\hat{\mathbf{M}} \times \hat{\mathbf{C}} \models \phi) \leq \mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi)$.

Remark 2.7. The robust operator defined in (2.53) becomes equivalent to the one in Haesaert and Soudjani (2020) by using a global probability deviation δ instead of the function $\boldsymbol{\delta}(\hat{x})$. This operator, denoted as $\mathbf{T}_{\epsilon,\delta}^{\mu_k}(\cdot)$ is used recursively to compute the robust satisfaction probability for linear systems.

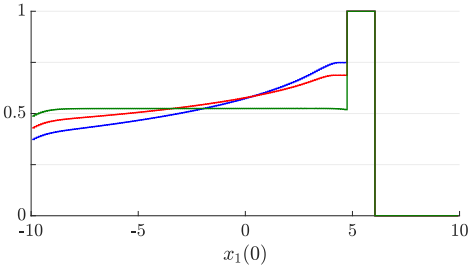
2.7 Results of the coupling compensator

To evaluate the benefits of the coupling compensator method, we consider specifications written using syntactically co-safe linear temporal logic (Kupferman and Vardi 2001; Belta et al. 2017), and analyze the influence of both the deviation bounds on the satisfaction probability. To this end, we consider two simple linear case studies of parking a car in a 1- and 2-dimensional space. Next, we illustrate the model-order reduction capabilities by considering a higher-dimensional case study of a building automation system with affine dynamics. Besides that, with this case study, we also show that the dimension of the input and disturbance can be smaller than the state dimension. Finally, we show the piecewise-affine abstraction method by considering a forced, stochastically perturbed Van der Pol oscillator.

The simulations have all been performed on a computer with a 2.3 GHz Quad-Core Intel Core i5 processor and 16 GB MHz memory using our `MATLAB` toolbox SySCoRe (details in Chapter 3). For all simulations, we report the average computation time and memory usage. The computation time is determined by taking the average of 5



(a) DFA $\mathcal{A}_{\phi_{park}}$, with labels p_1 and p_2 corresponding to regions P_1 and P_2 respectively.



(b) Robust satisfaction probability, where the blue, red, and green lines are obtained with (ϵ, δ) equal to $(0.05, 0.018)$, $(0.2, 0.012)$ and $(0.5, 0)$ respectively.

Figure 2.4: DFA corresponding to the specification $\phi_{park} = \neg p_2 \,\mathsf{U}\, p_1$ in **(a)** and robust satisfaction probability of the 1D car parking example in **(b)**.

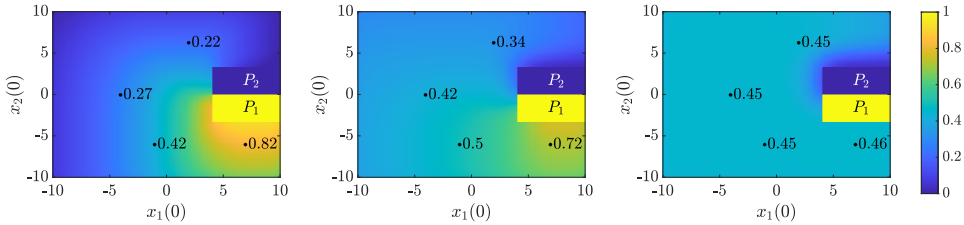
simulations and the memory usage is computed based on the sizes of the matrices stored in the workspace.

Car parking in 1- and 2-dimensional spaces

First, we consider a one-dimensional (1D) case study of parking a car. The dynamics of the car are modeled using (2.8) with $A = 0.9, B = 0.5$, $B_w = C = 1$, and with state space $\mathbb{X} = [-10, 10]$, input space $\mathbb{U} = [-1, 1]$, and output space $\mathbb{Y} = \mathbb{X}$. The unpredictable changes in the position of the car are captured by Gaussian noise $w \sim \mathcal{N}(0, 1)$. The goal of the controller is to guarantee that the car will be parked in parking spot p_1 while avoiding parking spot p_2. Using scLTL, this can be written as $\phi_{park} = \neg p_2 \mathsf{U} p_1$, which corresponds to the DFA $\mathcal{A}_{\phi_{park}}$ in Figure 2.4a. Here, we have chosen the regions $P_1 = [4.75, 6\rangle$ and $P_2 = [6, 10]$ defined on the output space $\mathbb{Y}$. First, we have computed a finite-state abstract model $\hat{\mathbf{M}}$ in the form of (2.10) by partitioning the state space with regions of size 0.1. Next, we have selected optimal values for deviation bounds ϵ and δ based on the optimization problem given in (2.22). Finally, we have computed the satisfaction probability using `SySCoRe` and achieved a computation time of approximately[4] 3.6 seconds and a memory usage of 4.52 MB. The results are shown in Figure 2.4b. Quantifying all the error on ϵ (green) yields a relatively low overall satisfaction probability that slightly decreases the further you are from the region P_1. The low overall probability is caused by the large ϵ value, which makes reaching the desired parking spot P_1 very difficult. On the other hand, quantifying all the error on δ (blue) yields a probability that starts relatively high, but steeply decreases the further you are from the region P_1. The presented method (red) can achieve a full trade-off between ϵ and δ thereby achieving a higher satisfaction probability for part of the state space.

As a second case study, we have considered parking a car in a two-dimensional (2D) space. More specifically, we have considered the model (2.8) with $A = 0.9I_2$,

[4] we observed a standard deviation of 0.2 seconds (5.6%).



(a) $(\epsilon, \delta) = (0.142, 0.051)$ **(b)** $(\epsilon, \delta) = (1.005, 0.016)$ **(c)** $(\epsilon, \delta) = (1.414, 0)$

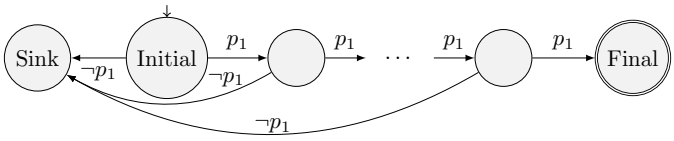
Figure 2.5: Robust satisfaction probability of the 2D car parking case study for different couplings. Figures 2.5a and 2.5c represent quantifying the deviation completely on δ or on ϵ respectively, while Figure 2.5b correspond to dividing the deviation between ϵ and δ.

$B = 0.7I_2$, $B_w = C = I_2$, and state space $\mathbb{X} = [-10, 10]^2$, input space $\mathbb{U} = [-1, 1]^2$, output space $\mathbb{Y} = \mathbb{X}$, and disturbance $w \sim \mathcal{N}(0, I_2)$. The goal is to synthesize a controller such that specification $\phi_{park} = \neg p_2 \mathsf{U} p_1$, with regions $P_1 = [4, 10] \times [-3.25, 0]$ and $P_2 = [4, 10] \times [0, 3.25]$ is satisfied. First, we have computed a finite-state abstract model $\hat{\mathbf{M}}$ in the form of (2.10) by partitioning the state space with square regions of size 0.2. Next, we have selected optimal values for deviation bounds ϵ and δ based on the optimization problem given in (2.22). Finally, we have computed the satisfaction probability using `SySCoRe` and achieved a computation time of approximately[5] 7.94 seconds and a memory usage of 27.53 MB. The results are shown in Figure 2.5 and are very similar to the 1D case, however, the influence from the avoid region (P_2) is more apparent in 2D. Furthermore, dividing the deviation between ϵ and δ (Figure 2.5b) shows a decent trade-off between quantifying the deviation completely on δ (Figure 2.5a) and ϵ (Figure 2.5c). In the sense that the satisfaction probability is relatively high overall, while not steeply decreasing the further you are from the region P_1 (or closer to region P_2).

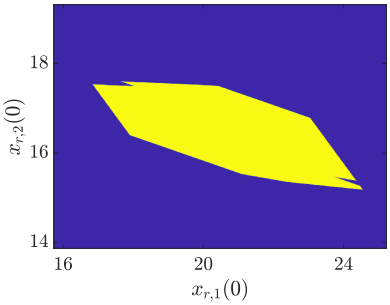
Building automation system

As a third case study, we have considered a Building Automation System (BAS) (Cauchi and Abate 2018) that is used in the benchmark study in Abate et al. (2020). The system consists of two heated zones with a common air supply. It has a 7-dimensional state with a 6-dimensional disturbance and a 1-dimensional control input as described in Cauchi and Abate (2018, Sec. 3.2). The 1-dimensional output equals the temperature in zone 1, which is the state x_1. The goal is to control the temperature in zone 1 such that it does not deviate from the set point ($20°C$) by more than $0.5°C$ over a time horizon equal to 1.5 hours, i.e., $\phi_T = \bigwedge_{i=0}^{5} \bigcirc^i p_1$ with p_1 the label corresponding to region $P_1 = [19.5, 20.5]$ defined on the output space $\mathbb{Y}$. The corresponding DFA $\mathcal{A}_{\phi_T}$ is given in Figure 2.6a. We have subsequently reduced the model to a 2-dimensional system and gridded the state space. We obtained $(\epsilon_r, \delta_r) = (0.2413, 0.0161)$ and $(\epsilon_{abs}, \delta_{abs}) = (0.1087, 0)$ for a $\|\beta\| \leq 1.8 \cdot 10^{-3}$.

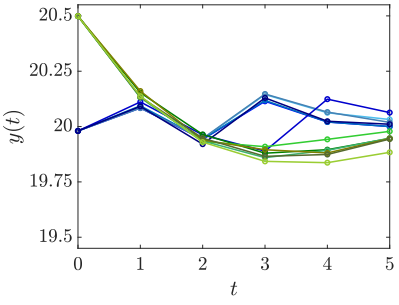
[5] we observed a standard deviation of 0.2 seconds (2.5%).



(a) DFA $\mathcal{A}_{\phi_T}$, with label p_1 corresponding to region P_1. Here, the dots indicate repeating states and transitions.



(b) Robust satisfaction probability, where blue and yellow correspond to a probability of 0 and 0.92 respectively.



(c) Trajectories starting at $x(0) = 20.5$ in green and starting at $x(0) = 19.98$ in blue.

Figure 2.6: DFA corresponding to $\phi_T = \bigwedge_{i=0}^{5} \bigcirc^i p_1$ in **(a)**, and robust satisfaction probability of the reduced order model used in the BAS case study with initial state $x_r(0) = [x_{r1}, x_{r2}]^\top$ in **(b)**. Multiple trajectories obtained by simulating the controlled system are shown in **(c)**.

This leads to a total deviation bound of $(\epsilon, \delta) = (0.35, 0.0161)$. Note that these results have been obtained for a slightly enlarged input set $u(t) \in [15, 33]$, originally $u(t) \in [15, 30]$. The satisfaction probability of 0.92 as shown in Figure 2.6b is consistent with Abate et al. (2020). The computation is performed using `SySCoRe` in approximately[6] 122 seconds and required a memory usage of 5.4 GB. Figure 2.6c shows some trajectories obtained from simulating the controlled system.

Comparison to available software tools. In Abate et al. (2020), the BAS benchmark has been used to compare the performance of `AMYTISS` (Lavaei et al. 2020a), `FAUST` (Soudjani et al. 2015), `SReachTools` (Vinod et al. 2019), and `StocHy` (Cauchi and Abate 2019). These tools all target the verification of stochastic systems with continuous state space. Of these tools, `SReachTools` is the most limited. It can only handle a very specific set of models with specifications limited to reach(-avoid) and invariance. In contrast, the tools `AMYTISS`, `FAUST`, and `StocHy` are all abstraction-based methods that can handle a wider set of temporal specifications. In comparison to the numerical results presented in the previous paragraph, these tools are more mature. `StocHy` is implemented in C++ and combines several advanced techniques such as symbolic probabilistic kernels and multi-threading. `AMYTISS` goes even further and utilizes parallel computations. A full comparison between

[6] we observed a standard deviation of 7 seconds (5.7%).

SySCoRe and other existing tools is given in Chapter 3. We briefly paraphrase those conclusions here for completeness. The results of SySCoRe and the other tools[7] are given in Table 2.1 (repeated from Table 3.4c.). If we compare our results, we notice that our implementation is performing on an equal footing. As indicated in the table, FAUST was unable to run this case study. StocHy required a very fine grid resulting in a very large computation time. AMYTISS obtains a good result, since it achieves a reasonable probability, however, the computation time is still quite large. It should be noted that AMYTISS performs very well when it can use all cores of a computer (Abate et al. 2020), which has not been done for this comparison. Our method yielded the second least conservative satisfaction probability, only SReachTools does better and it also performs very well with respect to computation time. It should, however, be noted that SReachTools is developed exactly for linear systems subject to stochastic reach-avoid problems with small disturbances. Hence, it is expected to perform best for this benchmark, but the tool is limited to a specific type of system and specification.

Method	Run time (sec)	Max. reach probability
AMYTISS	312.14	≈ 0.80
FAUST	-	-
SReachTools	4.59	≥ 0.99
StocHy	≥ 335.87	$\geq 0.80 \pm 0.23$
SySCoRe	102.86	≥ 0.92

Table 2.1: Results of the BAS case study for different tools. To compare the tools we exclude the deployment of the controller in SySCoRe since this step is not performed by the other tools.

Van der Pol oscillator

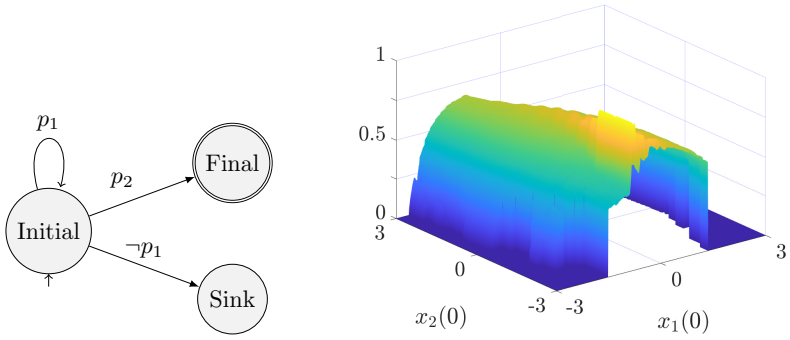
We have applied the piecewise-affine abstraction method from Section 2.5 to a forced, stochastically perturbed Van der Pol oscillator with state dynamics

$$\begin{aligned} x_1(t+1) &= x_1(t) + x_2(t)\tau + w_1(t) \\ x_2(t+1) &= f_2(x(t)) + u(t) + w_2(t), \end{aligned}$$

with nonlinear function $f_2(x(t)) = x_{2t} + (-x_{1t} + (1 - x_{1t}^2)x_{2t})\tau)$. Here, $\tau = 0.1$ is the sampling time and $w \sim \mathcal{N}(0, 0.04I_2)$ is a Gaussian disturbance[8]. The output equals the state, that is $y_t = x(t)$ and we have state space $\mathbb{X} = [-3, 3]^2$, input space $\mathbb{U} = [-1, 1]$ output space $\mathbb{Y} = \mathbb{X}$, safe region $P_1 = \mathbb{Y}$, and goal region $P_2 = [-1.2, -0.9] \times [-2.9, -2]$ defined on the output space $\mathbb{Y}$. The specification $\phi_{pol} = p_1 \mathsf{U} p_2$ means staying in the safe region while reaching the goal region. Here,

[7] obtained by running their repeatability packages on the same machine (computer with a 2.3 GHz Quad-Core Intel Core i5 processor and 16 GB MHz memory).

[8] The dynamics can equivalently be written in the form (2.33), by adding $B_w = 0.2I_2$ and using standard Gaussian variable $w \sim \mathcal{N}(0, I_2)$.



(a) DFA $\mathcal{A}_{\phi_{pol}}$. **(b)** Robust satisfaction probability.

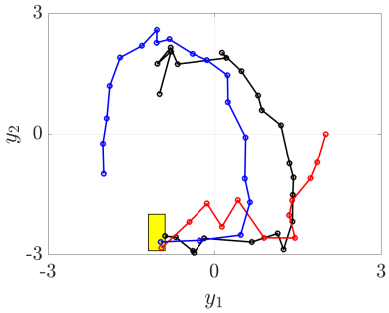
Figure 2.7: DFA corresponding to $\phi_{Pol} = p_1 \mathsf{U}\, p_2$ in **(a)** and robust satisfaction probability for the Van der Pol oscillator in **(b)**.

labels p_1 and p_2 correspond to regions P_1 and P_2 respectively. The DFA $\mathcal{A}_{\phi_{pol}}$ corresponding to this specification is given in Figure 2.7a.

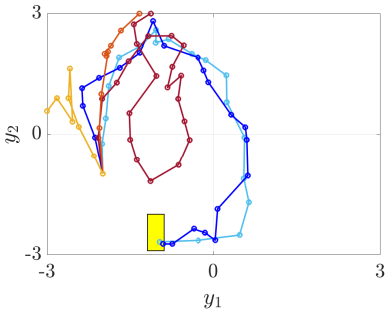
We obtained an abstract model with state dynamics as in (2.38) by partitioning the state space with square regions of width 0.01 leading to[9] $\beta \in \mathcal{B} = [-0.01, 0.01]^2$ and with $\hat{u} \in \hat{\mathbb{U}} = \{-0.6, 0, 0.6\}$ leaving some input action for the feedback part, namely $-0.4 \leq K_{f,i}(x - \hat{x}) \leq 0.4$. Next, we used 1600 equally sized square partitions to obtain a piecewise-affine abstraction as in (2.40). We then selected $\epsilon = 0.08$ and computed a corresponding probability deviation function $\boldsymbol{\delta}(\hat{x})$ such that the implications in (2.49) are satisfied. We computed a global stochastic kernel $\bar{\mathcal{W}}$ (2.51) and interface function $\mathcal{U}_v$ (2.52) and used Theorem 2.4 to obtain an $(\epsilon, \boldsymbol{\delta})$-stochastic simulation relation. Finally, we obtained a robust controller $\mathbf{C}$, and the robust satisfaction probability is shown in Figure 2.7b. The `MATLAB` implementation takes 53 minutes while using 178.8MB memory to store the variables in the workspace. Gridding takes approximately 45% of the computation time and 55% of the time is spent on computing the matrices in Lemma 2.2. Figure 2.8 shows some trajectories obtained from simulating the controlled system.

Comparison to available software tools. Similar case studies have been presented in (Majumdar et al. 2020; Abate et al. 2021), where Majumdar et al. (2020) considers an autonomous Van der Pol oscillator and Abate et al. (2021) combines the input with a multiplicative noise term. However, the results presented in Majumdar et al. (2020) and Abate et al. (2021) are limited to verification or a reachability analysis instead of the control synthesis performed in this paper. Furthermore, we have chosen a more stochastic variant with a Gaussian disturbance instead of a uniform distribution with bounded support as used in Majumdar et al. (2020) and Abate et al. (2021). The unbounded nature of the Gaussian disturbance contributes significantly to the difficulty of this case study.

[9] Normally, you get $\mathcal{B} = [-0.005, 0.005]^2$, however, our implementation uses an efficient tensor-based computation that leads to a bigger set for β.



(a) Satisfying trajectories.



(b) Satisfying trajectories in blue and violating trajectories in yellow, orange, and red. The robust satisfaction probability at $x(0)$ equals 0.38.

Figure 2.8: Trajectories of the controlled van der Pol oscillator, where goal region P_2 is indicated by the yellow box. In **(a)** satisfying trajectories starting at different initial states and in **(b)** trajectories starting at initial state $x(0) = \left[-2, -1\right]^\top$.

2.8 Conclusion

We have shown that the introduction of a coupling compensator increases the accuracy of the satisfaction probability of methods that use $(\epsilon, \delta)-$stochastic simulation relations. To the best of our knowledge, we are the first to define this coupling compensator and use it to reduce the complexity of the optimization problem that computes the similarity quantification. In the first part of this chapter, we have defined a structured methodology based on set-theoretic methods for linear stochastic difference equations. These set-theoretic methods leverage the freedom in coupling-based similarity relations and allow us to tailor the deviation bounds to the considered synthesis problem. We have applied this to compute the deviation bounds expressed with $(\epsilon, \delta)-$stochastic simulation relations for finite-state abstractions, reduced-order abstractions, and a combination thereof. In the second part of this chapter, we describe a temporal logic control method for nonlinear *stochastic* models that uses piecewise-affine approximations. By using a state-dependent probability deviation, a lower bound on the satisfaction probability is computed. The method described in this paper can handle (unbounded) scLTL specifications and applies to nonlinear systems with an unbounded additive disturbance. For both linear and nonlinear systems, we have illustrated that tailored deviation bounds that trade-off output and probability deviations can be beneficial to the satisfaction probability.

Future work Future work is mainly focused on improving the computational implementation. For example, by considering simulation relations with a polytopic shape instead of ellipsoidal, we might achieve more accurate results, while possibly also improving the computation time. Additionally, it would be interesting to see the impact of our approach on models with multiplicative noise and/or disturbances coming from a distribution with a bounded support.

Finally, the approach for nonlinear systems could be improved significantly. Our approach based on a piecewise-affine abstraction adds a lot of computational complexity, computation time, and memory usage. So, developing a more efficient approach specifically for nonlinear systems is an interesting topic for future work. One possible first step is efficiently learning the piecewise-affine abstraction using neural networks, as introduced in Abate et al. (2023).

Appendix

2.A Implementation for PWA abstractions

Here, we detail how to obtain matrices F_i and $K_{f,i}$ such that the implications in Lemma 2.2 are satisfied. To this end, we introduce Algorithm 2.2 to obtain bounded sets $\mathcal{K}_i$ and global matrix D that satisfies (2.42) and likely implies the existence of matrices F_i and $K_{f,i}$. The algorithm is based on using an optimistic

Algorithm 2.2 Get weighting matrix D and bounded sets $\mathcal{K}_i$.

1: **Input:** $\mathbf{M}, \hat{\mathbf{M}}, \epsilon$
2: Set $D = C^\top C$
3: Compute P_i and $\mathcal{K}_i$ using (2.45) and (2.36)
4: Choose $N_i \leq N_P$ to compute a suitable value for D
5: $D \leftarrow$ solve optimization problem (2.54)
6: Update P_i and $\mathcal{K}_i$ using (2.45) and (2.36)

preliminary estimate of matrix D and sets P_i and $\mathcal{K}_i$ (steps 2, 3). Next, we update these estimates by solving the following optimization problem for a small number of partitions $N_i \leq N_P$ (steps 4, 5).

$$\begin{aligned}
&\min_{D_{inv},L_i,Q_i,r_i} r_i^2 \text{ s.t. } D_{inv} \succ 0, \\
&\begin{bmatrix} D_{inv} & D_{inv}C^T \\ CD_{inv} & I \end{bmatrix} \succeq 0, \forall i \in \{1,\dots,N_i\} : \\
&\begin{bmatrix} \frac{1}{\epsilon^2}D_{inv} & L_i^T \\ L_i & r_i^2 I \end{bmatrix} \succeq 0, \begin{bmatrix} \frac{1}{\epsilon^2}D_{inv} & Q_i^T \\ Q_i & u_u^2 I \end{bmatrix} \succeq 0, && \text{(2.54a)} \\
&\begin{bmatrix} \lambda D_{inv} & * & * \\ 0 & (1-\lambda)\epsilon^2 & * \\ A_iD_{inv}+B_iQ_i+B_{w,i}L_i & \psi_l & D_{inv} \end{bmatrix} \succeq 0 && \text{(2.54b)}
\end{aligned}$$

where $D_{inv} = D^{-1}$, $L_i = F_i D_{inv}$, $Q_i = K_{f,i} D_{inv}$, and $\psi_l \in vert(\mathcal{B} \oplus \mathcal{K}_i)$. This optimization problem is parameterized in $\lambda \in [0,1]$ and constructed by following Section 2.4. Together with the given value of ϵ, we use matrix D to update sets P_i and $\mathcal{K}_i$ (step 6). Since we already obtained matrix D, we can compute matrices F_i and $K_{f,i}$ from Lemma 2.2 for all $i \in \{1,\dots,N_P\}$ in parallel by formulating an optimization problem similar to (2.54) with constraints (2.54a)-(2.54b).

3 SySCoRe: Synthesis via Stochastic Coupling Relations

In this chapter, we present SySCoRe, a MATLAB toolbox that synthesizes controllers for stochastic continuous-state systems to satisfy temporal logic specifications. Starting from a system description and a syntactically co-safe temporal logic specification, SySCoRe provides all necessary functions for synthesizing a robust controller and quantifying the associated formal robustness guarantees. It distinguishes itself from other available tools by supporting nonlinear dynamics, complex syntactically co-safe temporal logic specifications over infinite horizons, and model-order reduction. To achieve this, SySCoRe generates a finite-state abstraction of the provided model and performs probabilistic model checking. Then, it establishes a probabilistic coupling to the original stochastic system encoded in an approximate simulation relation, based on which a lower bound on the satisfaction probability is computed. SySCoRe provides non-trivial lower bounds for infinite-horizon properties and disturbances with an unbounded support since its computed error does not grow linearly in the horizon of the specification. It exploits a tensor representation to facilitate the efficient computation of transition probabilities. We showcase these features on several benchmarks and compare the performance of the tool with existing tools.

3.1 Introduction

The design of provably correct controllers is crucial for the development of safety-critical systems such as autonomous vehicles and smart energy grids (Alur 2015; Lee and Seshia 2016). To this end, methods for synthesizing controllers for dynamical systems that are guaranteed to satisfy temporal logic specifications have gained an increasing amount of attention in the control community (Baier and Katoen 2008;

Tabuada 2009; Belta et al. 2017; Lavaei et al. 2022a). Besides establishing the theory underlying these methods, it is equally important to develop tools that facilitate their application. For stochastic systems, a collection of tools that can perform formal controller synthesis is already available. A subset of these tools includes in alphabetical order: `AMYTISS` (Lavaei et al. 2020a), `FAUST` (Soudjani et al. 2015), `hpnmg` (Hüls et al. 2020), `HYPEG` (Pilch and Remke 2017), `Mascot-SDS` (Majumdar et al. 2020), the `Modest Toolset` (Hartmanns and Hermanns 2014), `ProbReach` (Shmarov and Zuliani 2015), `SReachTools` (Vinod et al. 2019), and `StocHy` (Cauchi and Abate 2019). A complete list of these tools with their descriptions and capabilities can be found in the ARCH Competition Report (stochastic category) (Abate et al. 2021). These tools perform the computations either using analytical methods or employing statistical model checking. The approaches in the analytical methods can further be divided into abstraction-based (Soudjani et al. 2015; Cauchi and Abate 2019; Lavaei et al. 2020a; Majumdar et al. 2020) and abstraction-free techniques (Vinod et al. 2019; Kochdumper et al. 2021). Abstraction-free techniques are generally less prone to suffering from the curse of dimensionality, however, they are often limited to simple invariance and reachability specifications. In contrast, abstraction-based tools can be applied to a breadth of systems and specifications. A survey on formal verification and control synthesis of stochastic systems is given in Lavaei et al. (2022a).

`SySCoRe` contributes to the category of tools that employ analytical abstraction-based methods. It is a `MATLAB` toolbox applicable to stochastic nonlinear systems with a possibly *unbounded support*. Furthermore, it can perform the controller synthesis to satisfy arbitrary syntactically co-safe specifications that can have *unbounded time horizons*. To this end, it uses the (ϵ, δ)-approximate simulation relation provided in Haesaert et al. (2017b), that explicitly designs the coupling between the continuous-state model and its (reduced) finite-state abstraction as in Chapter 2. Hence, `SySCoRe` extends the capabilities of the current tools by considering properties that are *unbounded in time* and by considering systems with an *unbounded disturbance*, that is a disturbance with an unbounded support.

`SySCoRe` is a comprehensive toolbox for temporal logic control of stochastic continuous-state systems, implementing all necessary steps in the control synthesis process. Moreover, it supports *model-order reduction* in the abstraction process with formal error quantification guarantees, which makes it applicable to a larger class of systems. To increase its computational efficiency, `SySCoRe` performs computations based on tensors and sparse matrices. Furthermore, computations based on efficient convex optimizations for polytopic sets are implemented where possible. The tool is developed with a focus on ease of use and extensibility, such that it can easily be adapted to suit individual research purposes. The development of `SySCoRe` is a step towards solving the tooling need for temporal logic control of stochastic systems as it expands both the class of models and the class of specifications for which abstraction-based methods can provide controllers with formal guarantees.

This chapter is organized as follows. We discuss in Section 3.2 the temporal logic control problem and the set-up in `SySCoRe`. We then give an overview of `SySCoRe` in Section 3.3 by introducing the associated functions and classes. Section 3.4 discusses multiple benchmarks that show the capabilities of `SySCoRe` and how it

compares to existing tools. We end this chapter with a summary and a discussion of possible extensions. Throughout, we give the core functions of `SySCoRe` in framed white boxes and example code in gray boxes.

3.2 Temporal logic control

The main purpose of `SySCoRe` is to perform the complete control synthesis procedure in abstraction-based temporal logic control. It is applicable to discrete-time models with a possibly unbounded stochastic disturbance and synthesizes a controller for satisfying syntactically co-safe linear temporal logic specifications that may have an unbounded time horizon. The computational approach is based on the theory of approximate simulation relations (Haesaert et al. 2017b), the coupling between models (Haesaert et al. 2017b; Huijgevoort and Haesaert 2022) and robust dynamic programming mappings (Haesaert and Soudjani 2020). In this section, we introduce the class of models and specifications handled by `SySCoRe` and show how to set up the problem. Furthermore, we provide a high-level description of the theory underlying the implementations in `SySCoRe`.

3.2a Problem parameters

Model. We consider discrete-time systems described by stochastic difference equations

$$M : \begin{cases} x_{t+1} = f(x_t, u_t) + B_w w_t \\ y_t = C x_t, \quad \forall t \in \{0, 1, 2, \dots\}, \end{cases} \tag{3.1}$$

with state $x_t \in \mathbb{X}$, input $u_t \in \mathbb{U}$, (unbounded) stochastic disturbance $w_t \in \mathbb{W}$, measurable function $f : \mathbb{X} \times \mathbb{U} \to \mathbb{X}$, and matrices B_w and C of appropriate sizes.

To handle nonlinear systems of the form (3.1), we perform a piecewise-affine (PWA) approximation that yields a system described by

$$\begin{cases} x_{t+1} = A_i x_t + B_i u_t + a_i + B_{w,i} w_t + \kappa_t \text{ for } x_t \in P_i \\ y_t = C x_t, \end{cases} \tag{3.2}$$

with P_i a partition of $\mathbb{X}$ and $\kappa_t \in \mathcal{K}_i$ the error introduced by performing the PWA approximation. For ease of notation, we denote the state-dependent error κ_{x_t} as κ_t. Furthermore, $A_i, B_i, B_{w,i}$ and a_i are matrices of appropriate sizes. Details of temporal logic control for nonlinear stochastic systems via piecewise-affine abstractions can be found in Chapter 2. Besides nonlinear systems, we also consider the special case of linear time-invariant (LTI) systems:

$$\begin{cases} x_{t+1} = A x_t + B u_t + B_w w_t \\ y_t = C x_t, \end{cases} \tag{3.3}$$

with A and B matrices of appropriate sizes.

Remark 3.1. This first release of `SySCoRe` assumes the disturbance w_t has unbounded Gaussian distribution $w_t \sim \mathcal{N}(0, I)$. The implementation for other classes of distributions is underway and will be included in the future release of the tool. Note that the assumption of standard Gaussian distribution with zero mean and identity covariance matrix is without loss of generality since any system (3.1)-(3.3) with disturbance $w \sim \mathcal{N}(\mu, \Sigma)$ can be rewritten to a system in the same class with an additional affine term (Allen et al. 2008).

To specify the model, that is a nonlinear system (3.1), a PWA system (3.2) or an LTI system (3.3), we have developed the classes `NonLinModel`, `PWAmodel`, and `LinModel`, respectively. The state space, input space, and the sets needed for defining the specification should be defined in these class descriptions.

Example. Consider a two-dimensional (2D) case study of parking a car with dynamics of the form (3.3) with $A = 0.9I_2$, $B = 0.7I_2$, and $B_w = C = I_2$. Furthermore, we have state space $\mathbb{X} = [-10, 10]^2$, input space $\mathbb{U} = [-1, 1]^2$, and disturbance $w \sim \mathcal{N}(0, I_2)$. After specifying the matrices A, B, C, B_w, and setting the values for the disturbance w with mean `mu` and covariance matrix `sigma` equal to zero and identity respectively, we can initialize a model in `SySCoRe` as follows:

```
% Set up an LTI model
sysLTI = LinModel(A,B,C,[],Bw,mu,sigma);
```

The state and input spaces are defined using `Polyhedron` from the `multi-parametric toolbox (MPT3)` (Herceg et al. 2013) as follows.

```
% Define bounded state space
sysLTI.X = Polyhedron(combvec([-10,10],[-10,10])');
% Define bounded input space
sysLTI.U = Polyhedron(combvec([-1,1],[-1,1])');
```

Specifications. In `SySCoRe`, we consider formal specification written using syntactically co-safe linear temporal logic (scLTL) (Kupferman and Vardi 2001; Belta et al. 2017), which consists of atomic proposition (AP) $\text{AP} = \{p_1, p_2, \ldots, p_N\}$ that are either true or false. To connect the system and the specification, we label the output space of the system, such that we can relate the trajectories of the system $\boldsymbol{y} = y_0, y_1, y_2, \ldots$ to the atomic propositions of the specification ϕ.

Example *(continued).* For the 2D car park, we consider reach-avoid specification ϕ_{park} with the region to reach P_1 and with avoid region P_2. First, we define the regions

```
% Specify regions for the specification
P1 = Polyhedron([4, -4; 4, 0; 10, 0; 10 -4]);
P2 = Polyhedron([4, 0; 4, 4; 10, 4; 10 0]);
```

and add them to the system object:

```
% Regions that get specific atomic propositions
sysLTI.regions = [P1;P2];
%  Propositions corresponding to the regions
sysLTI.AP = {'p1', 'p2'};
```

Implicitly, this means that outputs inside regions `P1` and `P2` are labeled using the corresponding atomic propositions `'p1'` and `'p2'`, respectively. Now, we can write the scLTL specification

$$\phi_{park} = \neg p_2 \,\mathsf{U}\, p_1, \tag{3.4}$$

using the syntax from Gastin and Oddoux (2001) as follows

```
% Define the scLTL specification
formula = '(!p2 U p1)';
```

Denote the system $\mathbf{M}$ under the controller $\mathbf{C}$ by $\mathbf{M} \times \mathbf{C}$ as in Tabuada (2009) and also introduced in Chapter 2 of this thesis. The goal is to synthesize a controller $\mathbf{C}$, such that the controlled system satisfies an scLTL specification ϕ, denoted as $\mathbf{M} \times \mathbf{C} \models \phi$. Since we consider stochastic systems, we compute the *satisfaction probability* denoted as $\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi)$. This goal is formulated mathematically next.

Problem statement. Given model $\mathbf{M}$, scLTL specification ϕ, and probability threshold $p_\phi \in [0, 1]$, design controller $\mathbf{C}$ such that

$$\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi) \geq p_\phi. \tag{3.5}$$

`SySCoRe` automatically synthesizes a controller by maximizing the right-hand side of (3.5) on a simplified abstract model and makes the computations robust to the abstraction errors. It provides a robust lower bound on the satisfaction probability, which then can be used by the user to compare with the probability threshold p_ϕ.

3.2b Stochastic coupling relations for control synthesis

To solve the above problem, we use an abstraction-based approach and the *dynamic programming mappings* from Haesaert and Soudjani (2020), which allows us to consider infinite-horizon properties. More specifically, the abstraction-based temporal logic control implemented in `SySCoRe` has six main steps, namely (1) translating the specification to an automaton, (2) constructing a (reduced) finite-state abstraction, (3) quantifying the similarity, (4) synthesizing a controller, (5) control refinement, and (6) deployment.

As visualized in Figure 3.1, we start from a temporal logic specification that expresses the desired behavior of the controlled system and translate it to an automaton

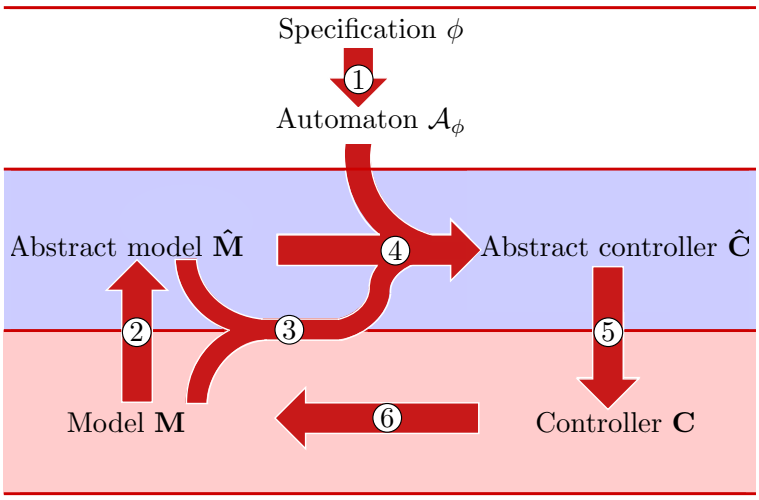


Figure 3.1: Steps in abstraction-based temporal logic control with 3 main layers: continuous-state (red), finite-state (blue), and specification (white). The numbers correspond to the following steps: (1) translating the specification to an automaton, (2) (reduced) finite-state abstraction, (3) similarity quantification, (4) synthesizing a controller, (5) control refinement, and (6) deployment.

(see top layer). A finite abstract model $\hat{\mathbf{M}}$ of the system is also constructed (step 2). For this abstract model $\hat{\mathbf{M}}$, its bounded deviation from the original model can be quantified using simulation relations as in Chapter 2 of this thesis (step 3). Computing these bounds is based on an efficient invariant set computation formulated as an optimization problem constrained by a set of parameterized matrix inequalities as described in Chapter 2. In step 4, we synthesize an abstract controller $\hat{\mathbf{C}}$ and compute the robust satisfaction probability, which gives a lower bound on the actual satisfaction probability. To compute the robust satisfaction probability and to synthesize an abstract controller $\hat{\mathbf{C}}$, `SySCoRe` solves a reachability problem over the abstract system combined with the automaton corresponding to the specification. This reachability problem is then solved as a dynamic programming problem. It is shown in Haesaert and Soudjani (2020) that leveraging the deviation bounds from step 3, the controller for the abstract model can be refined to the original continuous-state model while preserving the guarantees. To construct this controller $\mathbf{C}$, `SySCoRe` refines the abstract controller in step 5. The resulting controller $\mathbf{C}$ is a policy that can be represented with finite memory. Finally, `SySCoRe` deploys the controller on the model (step 6). It is important to note that the abstraction step (step 2 in Figure 3.1) can additionally contain model-order reduction or piecewise-affine approximation, which shows the comprehensiveness of `SySCoRe` enabled by establishing coupled simulation relations.

The next section gives a complete overview of the toolbox and specifies how each step in Figure 3.1 is implemented.

3.3 Toolbox overview

After setting up the problem by specifying the system using the classes `NonLinModel`, `PWAmodel` or `LinModel`, and the specification as an scLTL formula, we continue with the steps illustrated in Figure 3.1. Each step corresponds to a specific function as in Table 3.1. Note that the abstraction step may have multiple (formal) approximation stages depending on the type of the model or its dimension.

Table 3.1: Main functions of `SySCoRe` for steps (1)-(6), with optional steps (2a) and (2b).

Step	Function
(1) Translate the specification	`TranslateSpec`
(2) Finite-state abstraction	`FSabstraction`
(2a) Piecewise-affine approximation	`PWAapproximation`
(2b) Model-order reduction	`ModelReduction`
(3) Similarity quantification	`QuantifySim`
(4) Synthesize a controller	`SynthesizeRobustController`
(5) Control refinement	`RefineController`
(6) Deployment	`ImplementController`

3.3a Translating the specification

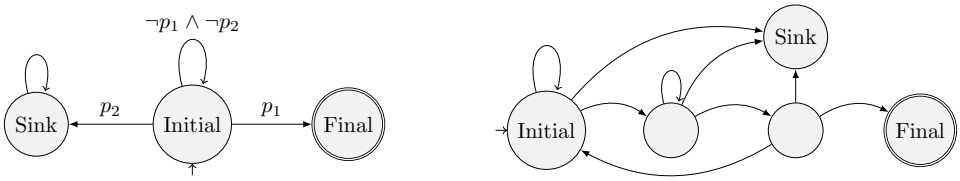
For control synthesis, the scLTL specification is written as a deterministic finite-state automaton (DFA) (Belta et al. 2017). Examples of such DFAs are given in Figure 3.2. We use the tool `LTL2BA`[1] to translate an scLTL specification, which constructs a non-deterministic Büchi automaton for a general LTL specification (Gastin and Oddoux 2001). Additionally, we check whether the given formula is written using scLTL (instead of full LTL) and then (if possible) rewrite the non-deterministic Büchi automaton to a DFA. This step is based on powerset conversion (Rabin and Scott 1959) that is used to convert a nondeterministic finite-state automaton to a DFA. The complete translation from an scLTL specification to a DFA is implemented in the function `TranslateSpec`.

```
% Translate an scLTL formula to a DFA
DFA = TranslateSpec(formula, AP);
```

The input `formula` is given using the syntax of `LTL2BA` in Gastin and Oddoux (2001).

Example *(continued).* For the 2D car park, we consider the reach-avoid specification ϕ_{park} in (3.4), which we translate to a DFA using `TranslateSpec` with `AP` and `formula` given respectively in code lines 13 and 15.

[1]Tool available at `http://www.lsv.fr/~gastin/ltl2ba/index.php`



(a) DFA corresponding to specification ϕ_{park} in (3.4) for the running example.

(b) Atypical DFA

Figure 3.2: Acyclic DFA in **(a)** versus cyclic DFA in **(b)**.

```
% Translate the spec to a DFA
DFA = TranslateSpec(formula, sysLTI.AP);
```

Besides reach-avoid specifications, it is also possible to describe many other types of specifications, such as more complex reach-avoid specification, e.g., $\phi_{PD} = \Diamond(p_1 \wedge (\neg p_2 \mathsf{U}\, p_3))$, or time-bounded and unbounded safety specifications, e.g. $\phi_{BAS} = \bigwedge_{i=0}^{5} \bigcirc^i p_1$ and $\phi_{vdPol} = p_1 \mathsf{U}\, p_2$. These specifications are written in SySCoRe as

```
formula_PD = 'F(p1 & (!p2 U p3))';                                    (3.6a)
formula_BAS = '(p1 & X p1 & X X p1 & X X X p1 ...                     (3.6b)
& X X X X p1 & X X X X X p1)';
formula_vdPol = '(p1 U p2)';                                          (3.6c)
```

Note that it is also possible to directly pass a DFA as an input to SySCoRe instead of giving the specification as an scLTL formula. SySCoRe can natively handle both acyclic and cyclic DFAs (see Figure 3.2), in contrast to many other tools (Soudjani et al. 2015; Rungger and Zamani 2016; Cauchi and Abate 2019; Lavaei et al. 2020a) that do not natively support DFAs but often rely on external tools such as PRISM (Kwiatkowska et al. 2002) to compute the controller.

3.3b Abstraction

SySCoRe includes two possible abstraction methods, namely finite-state abstraction for continuous-state systems in (3.1)-(3.3) and model-order reduction for continuous-state LTI systems (3.3). However, to create a finite-state abstraction of a nonlinear system (3.1), we require an additional approximation step before constructing a piecewise-affine finite-state abstraction. Note that the piecewise affine approximation itself is considered an integral part of the finite-state abstraction method.

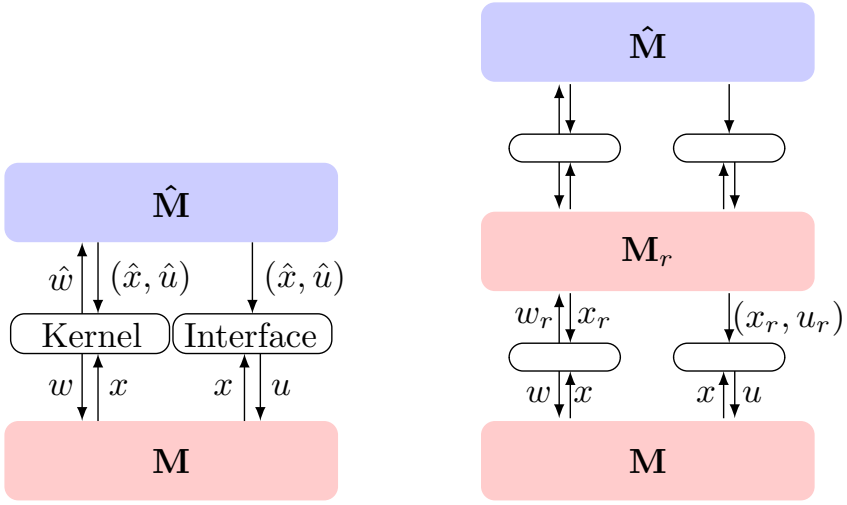
Piecewise affine approximation. To approximate a nonlinear system (3.1) by a PWA system (3.2), we partition the state space and use a standard first-order

Taylor expansion to approximate the nonlinear dynamics in each partition by affine dynamics. Additionally, we compute the error introduced by this approximation. In SySCoRe, this is performed by the function PWAapproximation.

```
% Perform piecewise-affine approximation
sysPWA = PWAapproximation(sysNonLin, Np);
```

Here, the nonlinear system (3.1) is given by sysNonLin, and the number of partitions in each direction is given by Np. The result is a PWA system (3.2) sysPWA.

Interface function. SySCoRe can construct a reduced-order abstract model $\mathbf{M}_r$ and a finite-state abstract model $\hat{\mathbf{M}}$ of the original model $\mathbf{M}$. Let us denote the control inputs of these models respectively by u_r and $\hat{u}$. The abstract control inputs u_r and $\hat{u}$ need to be refined to a control input u for $\mathbf{M}$ as illustrated in Figure 3.3.



(a) Coupling between models $\mathbf{M}$ and its finite-state abstraction $\hat{\mathbf{M}}$ through their inputs and disturbances via an interface function and a coupling kernel.

(b) Coupling between continuous-state models $\mathbf{M}$ and $\mathbf{M}_r$, and between $\mathbf{M}_r$ and its finite-state abstraction $\hat{\mathbf{M}}$.

Figure 3.3: Coupling between different models. Red and blue boxes correspond to respectively continuous-state and finite-state. In **(a)** only a finite-state abstraction is performed, while in **(b)** both model-order reduction and a finite-state abstraction are shown.

The input refinement is performed by one or multiple interface functions, namely

$$u_{r,t} = \hat{u}_t \qquad \textit{(default)} \qquad (3.7a)$$

$$u_{r,t} = \hat{u}_t + K(x_{r,t} - \hat{x}_t) \qquad \textit{(option 1)} \qquad (3.7b)$$

$$u_t = u_{r,t} + Qx_{r,t} + K_{MOR}(x_t - Px_{r,t}). \qquad \textit{(option 1, MOR)} \qquad (3.7c)$$

To refine the input $\hat{u}$ of a finite-state model to the input u_r of a continuous-state reduced-order model, we implemented two different interface functions in the format

of (3.7a) and (3.7b). For many cases the default interface function (3.7a) should work fine, however, the option (3.7b) gives more influence on the refined controller by including a feedback term. When the interface function (3.7b) is used, we have to take this into account when constructing the finite-state abstraction to avoid the input bounds being violated, therefore, the interface function must be chosen before constructing the finite-state abstraction. We further use the interface function (3.7c) to refine the input u_r of a reduced-order model to the input u of the full-order model. It should be noted that if only a finite-state abstraction is performed without using model-order reduction (MOR), we have $P = I, Q = 0$ and $x_t = x_{r,t}$, hence we obtain interface functions (3.7a) and (3.7b) with $u_t = u_{r,t}$ and $x_t = x_{r,t}$.

Example *(continued).* It is required to select an interface function for the input refinement before starting with the temporal logic control steps. For this running example only use the default interface function (3.7a) without model-order reduction, that is $u_t = \hat{u}_t$. However, if desired, the user can select the option (3.7b) by setting `int_f = 1` and passing this to the functions.

In the remainder of this section, we discuss how to obtain the reduced-order and finite-state abstract models.

Model-order reduction. It is essential to include model-order reduction for high-dimensional models. For LTI systems (3.3) this yields a reduced-order model $\mathbf{M}_r$ of the form

$$M_r : \begin{cases} x_{r,t+1} = A_r x_{r,t} + B_r u_t + B_{rw} w_{r,t} \\ y_{r,t} = C_r x_{r,t}, \end{cases} \tag{3.8}$$

with $x_r \in \mathbb{X}_r, u \in \mathbb{U}$, $y \in \mathbb{Y}$, $w_r \in \mathbb{W}$, and matrices A_r, B_r, B_{rw} and C_r of appropriate sizes.

In `SySCoRe`, the function `ModelReduction` constructs a reduced-order model `sysLTIr` of dimension `dimr` based on the original model `sysLTI` by using *balanced truncations* on a closed loop system with a feedback matrix F. This feedback matrix is computed by solving *discrete-time algebraic Riccati equations* that can be tuned using constant `f` (Pappas et al. 1980). The syntax of `ModelReduction` is

```
% Construct reduced-order model
[sysLTIr, F] = ModelReduction(sysLTI, dimr, f)
```

We couple the inputs u, u_r from $\mathbf{M}$ (3.3) and $\mathbf{M}_r$ (3.8) using the interface function (3.7c) as illustrated in Figure 3.3b. This is based on the theoretical results presented in Haesaert et al. (2017b) and Huijgevoort and Haesaert (2022). To compute matrices P and Q for the interface function, we have the function `ComputeProjection` that adds the matrices automatically to the object `sysLTIr`.

```
% Compute matrices P and Q
sysLTIr = ComputeProjection(sysLTI, sysLTIr);
```

Finite-state abstraction. We grid the state space to construct a finite-state abstraction $\hat{\mathbf{M}}$ of the continuous-state models (3.2), (3.3) or (3.8). More specifically,

we compute the abstract state space $\hat{\mathbb{X}}$ as the set consisting of the centers of the grid cells. Next, the dynamics of the abstract model are defined by using the operator $\Pi : \mathbb{X} \to \hat{\mathbb{X}}$ that maps states x to the center of the grid cell it is in. Details on how to construct a finite-state abstraction of a nonlinear system or an LTI system can be found in Section 2.5a and Section 2.4a respectively.

In `SySCoRe`, the construction of the finite-state abstraction is implemented in the functions `GridInputSpace` and `FSabstraction`. The function `GridInputSpace` constructs the abstract input space `uhat` by selecting a finite number of inputs from the input space `sysLTI.U`.

```
% Construct abstract input space
[uhat, InputSpace] = GridInputSpace(lu, sys.U, options);
```

Here, `lu` is the number of abstract inputs in each direction and `options` are used to select an interface function from (3.7). If interface function (3.7b) or (3.7c) is chosen, `GridInputSpace` also divides the continuous input space into a part for actuation and for feedback and returns these spaces as output `InputSpace`. This is done to make sure that the input bounds $u \in \mathbb{U}$ of the original model are satisfied. Next, we use `FSabstraction` to compute a *probability matrix* that contains the transition probabilities between states for all possible inputs in `uhat`.

```
% Construct abstract model
sysAbs = FSabstraction(sys, uhat, l, tol, DFA, options);
```

Here, `sys` is the continuous-state system, `uhat` is the abstract input space $\hat{\mathbb{U}}$, `l` is the number of grid cells in each direction and `tol` is the tolerance for truncating to zero. This means that if a probability is smaller than the value set by `tol`, then we set it to zero to increase sparsity and hence decrease computation time. Via efficient tensor computations, we split the computation of the probability matrix into two parts: one for the deterministic part of the transitions computed as a sparse matrix, and one for the stochastic part of the transitions. This reduces the required memory allocation and computation time drastically. For development purposes `options` can be used to select whether or not to use this efficient *tensor* computation. The complete probability matrix can then be obtained by using a tensor multiplication, however, we do not store the complete probability matrix and compute it when necessary to save memory.

Example *(continued).* To construct a finite-state abstraction of the car park model `sysLTI` (defined in code lines 1-13), we compute the abstract input space `uhat`:

```
18 % Construct abstract input space uhat
19 lu = 3;  % number of abstract inputs
20 uhat = GridInputSpace(lu, sysLTI.U);
```

and construct the abstract model `sysAbs` using the `DFA` constructed in code line 17 as follows:

```
% Construct finite-state abstraction
l = [200, 200];  % number of grid cells
tol = 10^-6;
sysAbs = FSabstraction(sysLTI, uhat, l, tol, DFA, ...
      'TensorComputation', true);
```

3.3c Similarity quantification

To quantify the similarity between the model and its abstraction (either reduced order or finite state), we compute ϵ and δ such that they satisfy the (ϵ, δ)-stochastic simulation relation as defined in Definition 2.4 (Section 2.4a). Here, ϵ and δ represent bounds on the output and probability deviations, respectively. This simulation relation allows us to consider scLTL specifications with unbounded time properties (Haesaert and Soudjani 2020).

When using model-order reduction, we construct two simulation relations, one relation $\mathcal{R}_{MOR}$ between the original model $\mathbf{M}$ (3.3) and reduced-order model $\mathbf{M}_r$ (3.8), and one relation $\mathcal{R}$ between $\mathbf{M}_r$ and the finite-state model $\hat{\mathbf{M}}$. The simulation relations are of the form

$$\mathcal{R}_{MOR} := \{(x_r, x) \in \mathbb{X}_r \times \mathbb{X} \mid ||x - Px_r||_{D_r} \leq \epsilon_r\} \tag{3.9a}$$

$$\mathcal{R} := \left\{(\hat{x}, x_r) \in \hat{\mathbb{X}} \times \mathbb{X}_r \mid ||x_r - \hat{x}||_D \leq \epsilon\right\}, \tag{3.9b}$$

with $||x||_D = \sqrt{x^\top D x}$ the weighted two-norm, where $D = D^\top \succeq 0$ is positive semi-definite. Following Section 2.4b, these simulation relations can be combined into one total simulation relation between $\mathbf{M}$ and $\hat{\mathbf{M}}$. Following Haesaert and Soudjani (2020, Section IV.A), we can now compute the initial state of the reduced-order model as the state $x_{r,0}$ that minimizes $||x_0 - Px_{r,0}||_{D_r}$, that is $x_{r,0} := (P^\top D_r P)^{-1} P^\top D_r x_0$.

The computation of the simulation relation relies heavily on the coupling of the inputs $u, \hat{u}$ and disturbances $w, \hat{w}$ of the two models. The inputs are coupled through an interface function and the disturbances via a coupling kernel. This is illustrated in Figure 3.3 and is based on the method developed in Chapter 2. More specifically, the underlying computation is based on finding an invariant set for the error dynamics $x_{r,t+1} - \hat{x}_{t+1}$. To this end, an optimization problem constrained by parameterized linear matrix inequalities is used to find a value for δ that corresponds with the given value of ϵ (see Chapter 2 for more details). To solve this optimization problem, we use the `multi-parametric toolbox (MPT3)` (Herceg et al. 2013) with `YALMIP` (Lofberg 2004) and with either solver `SeDuMi` (Labit et al. 2002) or `MOSEK` (ApS 2019).

In `SySCoRe`, similarity quantification is implemented in the function `QuantifySim`.

```
% Quantify similarity
[simRel, interface] = QuantifySim(sys, sysAbs, epsilon, options)
```

This function quantifies the similarity between the models `sys` and `sysAbs`, with `sysAbs` either a reduced-order or a finite-state approximation of `sys`, hence in terms

of behavior `sysAbs` $\preceq$ `sys`. `QuantifySim` yields a simulation relation `simRel` of the form (3.9) that is stored in the object `SimRel`. This object includes a method to check whether two states belong to the simulation relation and a method to combine the two simulation relations from (3.9) if necessary. Besides that, the function `QuantifySim` also returns the feedback-matrix of the `interface` function, when interface (3.7b) or (3.7c) is chosen through the `options`.

Example *(continued).* Next, we quantify the similarity between the model of the car stored in `sysLTI` and its finite-state abstraction `sysAbs` constructed in code line 24 by choosing a suitable value for ϵ and using the function `QuantifySim`.

```
25 % Choose a value for epsilon
26 epsilon = 1.005;
27 % Quantify similarity
28 simRel = QuantifySim(sysLTI, sysAbs, epsilon);
```

Piecewise affine systems. The function `QuantifySim` can handle both PWA (3.2) and LTI models (3.3). However, for PWA systems the probability deviation is a PWA function $\boldsymbol{\delta}(\hat{x})$ that depends on the partition of the abstract state as described in Section 2.5.

3.3d Synthesizing a robust controller

We synthesize a robust (finite-state) controller based on the dynamic programming approach described in Haesaert and Soudjani (2020), which is robust in the sense that it takes the deviation bounds ϵ and δ into account to compute a lower bound on the actual satisfaction probability. Furthermore, it is proven in Haesaert and Soudjani (2020, Theorem 4) that the resulting control policy synthesized for the abstract model can always be refined to a control policy for the actual model.

More specifically, we implicitly construct a product composition of the finite-state model $\hat{\mathbf{M}}$ with the DFA such that computing the *satisfaction probability* becomes a reachability problem over this product composition. This can in turn be solved using *dynamic programming* by associating a robust dynamics programming operator that allows for an iterative computation of the lower bound on the satisfaction probability. Denote the state of the DFA by q, then the probability that a trajectory starting at $(\hat{x}, q)$ reaches the set of accepting states by applying policy $\boldsymbol{\mu}$ within horizon $[1, 2, \dots N]$ is denoted as $V_N^{\boldsymbol{\mu}}(\hat{x}, q)$. This is equivalent to the probability of satisfying the specification ϕ over this time horizon. The probability V is computed iteratively by defining the operator

$$\mathbf{T}^{\hat{u}}(V)(\hat{x}, q) := \mathbf{L}\left(\mathbb{E}_{\hat{u}}\left(\min_{q^+ \in Q^+} \max\{\mathbf{1}_{Q_f}(q^+), V(\hat{x}^+, q^+)\}\right) - \delta\right), \tag{3.10}$$

where $\hat{x}^+$and q^+ are resp. the next state of the abstract model and the DFA, $\mathbb{E}$ is expectation with respect to the probabilistic transitions in the abstract model,

$\mathbf{1}_{Q_f}(q)$ is an indicator function that is equal to 1 if q is inside the set of accepting states Q_f of the DFA and is 0 otherwise, $\mathbf{L} : \mathbb{R} \to [0, 1]$ is a truncation function, and with

$$Q^+(q, \hat{y}^+) := \left\{ \tau_{\mathcal{A}_\phi}(q, L(y^+)) \mid ||y^+ - \hat{y}^+|| \leq \epsilon \right\}, \tag{3.11}$$

where $\tau_{\mathcal{A}_\phi}$ is the transition function of the DFA and $L(y^+)$ is the label of the next output. This operator is robust in the sense that the probability gets reduced by δ at every time step and the worst-case transition of the DFA is considered with respect to ϵ. The derivation of this operator for Markov decision processes can be found in Haesaert and Soudjani (2020).

Synthesis of an abstract control strategy `pol` and the computation of the robust *satisfaction probability* `satProb` is performed by the function `SynthesizeRobustController` and it is based on the abstract model `sysAbs`, the specification as a `DFA` and the simulation relation `simRel`.

```
% Compute satisfaction probability and policy
[satProb, pol] = SynthesizeRobustController(...
sysAbs, DFA, simRel, thold, options)
```

We include the possibility to set the threshold `thold` that stops the value iteration when the difference between two iterations is smaller than this threshold. The default value is set to $1 \cdot 10^{-12}$. This choice is justified by the fact that the operator in (3.10) is contractive and will always converge monotonically to a fixed point. Additionally, we include the `options` to compute the value function only for the initial DFA state and to compute an upper bound on the satisfaction probability. Internally, the dynamic programming algorithm computes the product between large-scale matrices (one of which is the probability matrix as mentioned in Section 3.3b on finite-state abstractions). By performing these computations using a tensor product (Nilsson et al. 2018), we gain superior computational efficiency.

The resulting control policy `pol` is a mapping $\boldsymbol{\mu} : \hat{\mathbb{X}} \times Q \to \hat{\mathbb{U}}$ from the pair of abstract and DFA states to the abstract input space. The abstract controller can now be written as $\hat{\mathbf{C}} : \hat{u} = \boldsymbol{\mu}(\hat{x}, q)$.

Example *(continued).* After specifying the desired threshold for convergence `thold`, we synthesize a robust control policy `pol` based on the finite-state abstract model `sysAbs`, the specification as a `DFA` and the simulation relation `simRel` constructed in code line 28. In this case, we are only interested in the satisfaction probability `satProb` of the initial DFA state, hence we set the `options` to `true`.

```
29 % Specify threshold for convergence error
30 thold = 1e-6;
31 % Synthesize an abstract robust controller
32 [satProb, pol] = SynthesizeRobustController(...
33 sysAbs, DFA, simRel, thold, true);
```

The robust satisfaction probability is computed for all $x_0 \in \mathbb{X}$ and is illustrated in Figure 3.4a.

3.3e Control refinement

To refine an abstract finite-state controller to a controller **C** that can be implemented on the original continuous-state system (see step 5 in Figure 3.1), we use one or multiple interface functions from (3.7) as illustrated in Figure 3.3. In `SySCoRe`, control refinement is included in the class `RefineController`, where it is possible to select an interface function using the `options`.

```
% Refine abstract controller
Controller = RefineController(satProb, pol, sysAbs, simRel, sys, ...
    DFA, options);
```

This class not only refines the finite-state input to the actual input but also determines the state of the finite-state model based on the state of the original model.

Example *(continued).* To construct a controller **C** that can be implemented on the original model **M** based on the abstract control policy `pol` computed in code line 32, we use the following.

```
34 % Refine abstract controller
35 Controller = RefineController(satProb, pol, sysAbs, simRel, ...
       sysLTI, DFA);
```

3.3f Deployment

The final step is to deploy the controller on the model and perform simulations using `ImplementController`.

```
% Implement the controller on the model
xsim = ImplementController(x0, N, Controller, option);
```

Here, `N` is the desired time horizon for the simulation and `option` is used to supply the number of trajectories and/or additional model-order reduction inputs.

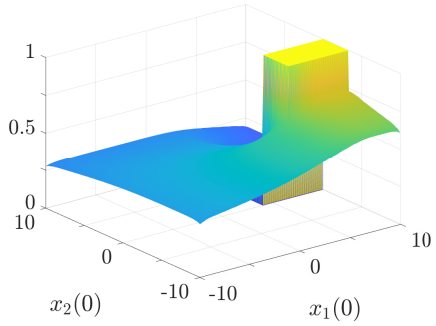
Example *(continued).* To simulate the controlled system with the `Controller` constructed in code line 34, we use `ImplementController` to obtain the state trajectory starting at x_0. Trajectories of the controlled system for three initial states are illustrated in Figure 3.4c.

```
36 x0 = [-4; -5]; % initial state
37 N = 40; % time horizon
38 % Simulate controlled system
39 xsim = ImplementController(x0, N, Controller);
```

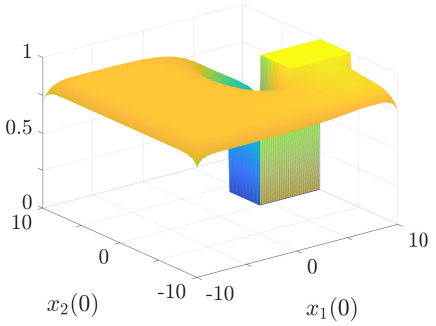
Example *(continued - improved results).* To improve the results, we can select interface function $u_t = \hat{u}_t + K(x_t - \hat{x}_t)$ as in (3.7b) by setting `int_f = 1` and passing this to the functions `GridInputSpace`, `QuantifySim`, and `RefineController`. We update among others line 20 to use respectively 60% and 40% of the input space for actuation and feedback. The new line is as follows.

```
40 % Construct abstract input space uhat
41 [uhat,sysLTI.U] = ...
       GridInputSpace(lu,sysLTI.U,'interface',int_f,0.6,0.4);
```

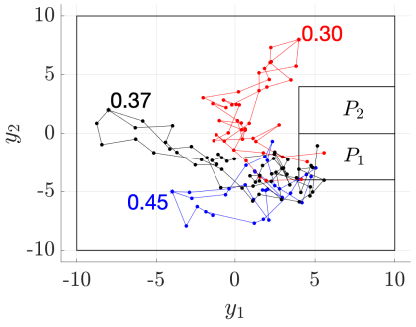
The resulting satisfaction probability and trajectories are given in Figures 3.4b and 3.4d. Comparing Figures 3.4a and 3.4b, we see that the satisfaction probability is increased substantially when using this interface function. In this case, the additional state-feedback term K compensates for the large stochastic disturbance that causes states $\hat{x}_t$ to deviate from x_t. Comparing Figures 3.4c and 3.4d, we see that the controllers are different, especially for initial state $[4, 8]^\top$, the controller seems to be more cautious with respect to avoiding region P_2 when interface function (3.7b) is applied.



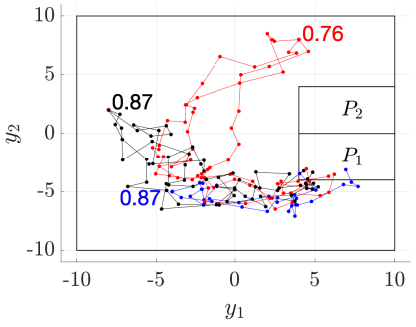
(a) Robust satisfaction probability, when interface function (3.7a) is used.



(b) Robust satisfaction probability, when interface function (3.7b) is used.



(c) Three different trajectories for some initial states with the corresponding satisfaction probability.



(d) Three different trajectories for some initial states with the corresponding satisfaction probability.

Figure 3.4: Results of the running example with interface function $u_t = \hat{u}_t$, that is (3.7a), in **(a)**,**(c)**, and with interface function $u_t = \hat{u}_t + K(x_t - \hat{x}_t)$, that is (3.7b), in **(b)**,**(d)**. The robust satisfaction probability for all initial states $\left[x_1(0), x_2(0)\right]^\top \in \mathbb{X}$, is shown in **(a)** and **(b)**. In **(c)** and **(d)**, we show three trajectories for each initial state: $x_0 = \left[-4, -5\right]^\top$ (blue), $x_0 = \left[-8, 2\right]^\top$ (black), and $x_0 = \left[4, 8\right]^\top$ (red). The corresponding robust satisfaction probability is given at the initial state.

3.4 Benchmarks

To show the capabilities of `SySCoRe`, we included multiple benchmarks, of which some are discussed here. The package delivery has a complex specification with a cyclic DFA, the building automation system includes model-order reduction, and the Van der Pol oscillator is nonlinear. We evaluate the run time and memory usage of the benchmarks and compare `SySCoRe` to some existing tools.

3.4a Package delivery

With the package delivery benchmark (Abate et al. 2022), we show the capability of `SySCoRe` to handle complex scLTL specifications beyond basic reach-avoid scenarios, i.e., cyclic DFAs. Consider an agent traversing in a 2D space, whose dynamics can be described by an LTI system (3.3) with $A := 0.9I_2$, $B := I_2$, $B_w := \sqrt{0.2}I_2$, $C := I_2$, and disturbance $w_k \sim \mathcal{N}(0, I_2)$. We initialize the system using `LinModel`.

Define the state space $\mathbb{X} = [-6, 6]^2$, input space $\mathbb{U} = [-1, 1]$, output space $\mathbb{Y} = \mathbb{X}$, and regions P_1, P_2 and P_3 as follows: $P_1 := [5, 6] \times [-1, 1]$, $P_2 := [0, 1] \times [-5, 1]$ and $P_3 := [-4, -2] \times [-4, -3]$. The agent can pick up a package at P_1 and must deliver it to P_3. If the agent visits P_2 while carrying a package, it loses the package and has to pick up a new package at P_1. This corresponds to the scLTL specification $\Diamond(p_1 \wedge (\neg p_2 \,\mathsf{U}\, p_3))$ implemented as in (3.6a). We generate the corresponding DFA using `TranslateSpec` and obtained the DFA as shown in Figure 3.5a.

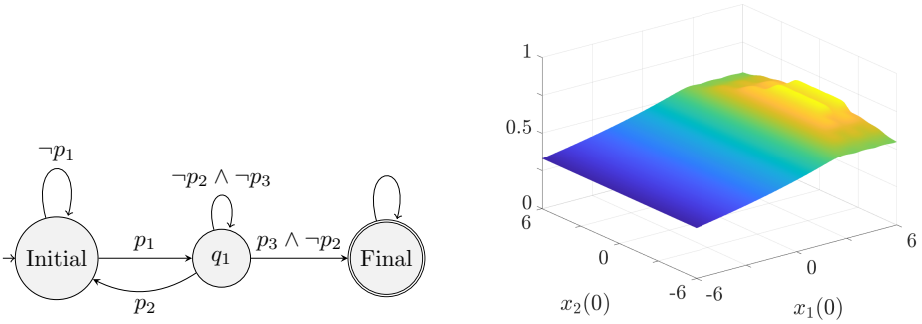
Next, we construct a finite-state abstraction using `GridInputSpace` and `FSabstraction`. For this study, we choose a comparatively fine state abstraction $l = [400, 400]$, which allows us to generate a simulation relation using `QuantifySim` with an epsilon of just 0.075. Note that the partition size l is a tuning parameter that is determined empirically. We synthesize a robust controller for the discrete abstraction using

```
[satProb, pol] = SynthesizeRobustController( ...
sysAbs, DFA, rel, thold, false);
```

Since the resulting control policy is conditional on both the current system state and the DFA state, we set the 5th argument to `false`. By doing so, we synthesize a controller for all DFA states instead of only the initial one. The obtained robust satisfaction probability `satProb` over different initial states x_0 is displayed in Figure 3.5b and can be obtained by running

```
% Plot satisfaction probability
plotSatProb(satProb, sysAbs, 'initial', DFA);
```

The peak satisfaction probability is 0.663.



(a) DFA corresponding to ϕ_{PD}. **(b)** Robust satisfaction probability.

Figure 3.5: Result of the package delivery benchmark. In **(a)**, the DFA $\mathcal{A}_{\phi_{PD}}$. In **(b)**, the robust satisfaction probability for the package delivery benchmark as a function of the initial state x_0. We only show the satisfaction probability for the initial DFA state.

Finally, we refine the controller using `RefineController`. To demonstrate the performance of the obtained controller, we simulate the controlled system using `ImplementController` for $N = 60$ time steps and an initial state of $x_0 = [-5, -5]^T$. Note that N is an empirical parameter and should be set high enough for the DFA to terminate. As expected, the agent moves to region P_1 to pick up a package and delivers it to P_3 whilst avoiding P_2. To plot trajectories we included the function `plotTrajectories`.

3.4b Van der Pol oscillator

In this benchmark, we show how `SySCoRe` can be applied to nonlinear stochastic systems. For this, consider the discrete-time dynamics of the Van der Pol oscillator (Abate et al. 2022), given by

$$\begin{aligned} x_{1,t+1} &= x_{1,t} + x_{2,t}\tau + w_{1,t} \\ x_{2,t+1} &= x_{2,t} + (-x_{1,t} + (1 - x_{1,t}^2)x_{2,t})\tau + u_t + w_{2,t}, \end{aligned} \tag{3.12}$$

where the sampling time τ is set to $0.1s$, $w_t \sim \mathcal{N}(0, 0.04I_2)$, and $y_t = x_t$. We define the state space $\mathbb{X} = [-3, 3]^2$, input space $\mathbb{U} = [-1, 1]$, and output space $\mathbb{Y} = \mathbb{X}$. For the Van der Pol oscillator, we are looking at an unbounded safety specification (cf. (3.6c)), where the objective is to synthesize a controller such that the system remains in the region $P_1 := \mathbb{Y}$ until reaching region $P_2 := [-1.4, -0.7] \times [-2.9, -2]$, corresponding to the scLTL specification $p_1 \,\mathsf{U}\, p_2$. First, we construct a DFA for the formula (3.6c) using `TranslateSpec`.

Since the dynamics of the oscillator (`sysNonLin`) in (3.12) are nonlinear, the abstraction process is split into two parts as outlined in Section 3.3b. First, we construct a PWA approximation as follows:

```
% Number of grid points in each direction
N = [41 41];
% Perform PWA approximation
sysPWA = PWAapproximation(sysNonLin, N);
```

In the second part of the abstraction step, a finite-state abstraction (`sysAbs`) of the PWA approximation (`sysPWA`) is constructed using `GridInputSpace` and `FSAbstraction` with `l=[600,600]` grid cells. To generate a simulation relation between this abstraction and the original model, we set $\epsilon = 0.08$ and compute a suitable weighting matrix D for the simulation relation on $(\hat{x}, x)$, as described in Section 3.3c. To reduce computation time, we only use a finite number of states to compute this weighting matrix. Details on why we need this global weighting matrix can be found in Section 2.5.

```
% Compute weighting matrix D for the simulation relation based on ...
    the following states
States = [1/8*x1l, 6/10*x2u; 5/7*x1u, 5/17*x2u; 2/13*x1u, 5/9*x2l; ...
    3/4*x1l, 1/7*x2l; 0, 0]';
[D, ~] = ComputeD(epsilon, sysPWA, sysAbs, 'interface', int_f, ...
    'states', States);
% Quantify similarity
[rel, sysPWA] = QuantifySim(sysPWA, sysAbs, epsilon, 'interface', ...
    int_f, 'weighting', D);
```

Note that `QuantifySim` returns `sysPWA` instead of the usual `interface` because each piecewise-affine system gets its own interface function and we store this directly in `sysPWA`.

Next, we use `SynthesizeRobustController` to synthesize a robust controller for `sysAbs` and show the satisfaction probability (displayed in Figure 3.6) using `plotSatProb`. Finally, we refine the controller as follows:

```
Controller = RefineController(satProb, pol, sysAbs, rel, sysPWA, ...
    DFA, int_f);
```

As before, `ImplementController` is used to simulate the system.

3.4c Building automation system

In the last benchmark, we address a large-scale system showcasing the model-order reduction capabilities of `SySCoRe`. We consider a 7D *affine* stochastic system of a building automation system, regulating the temperature in two zones influenced by a 6D disturbance. A detailed description including the system dynamics can be found in Abate et al. (2018) and Cauchi and Abate (2018). The goal is to synthesize a controller maintaining the temperature in zone one at $20°C$ with a maximum

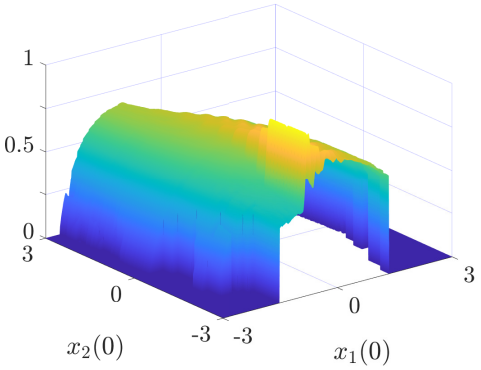


Figure 3.6: Robust satisfaction probability for the Van der Pol oscillator benchmark as a function of the initial state.

permissible deviation of $\pm 0.5°C$ for 6 consecutive time steps. We translate the specification (3.6b) to a DFA using `TranslateSpec`.

The dynamics of this building automation system are not of the form (3.3), since it is influenced by a Gaussian disturbance with mean $\mu \neq 0$ and variance $\Sigma \neq I$. Furthermore, it is not an LTI system but has affine state dynamics, which cannot be handled by our current implementation of model-order reduction. To deal with the disturbance, we first transform the system to a system with Gaussian disturbance $w \sim \mathcal{N}(0, I)$ using the following:

```
% Transform the model
[sysLTI, a] = NormalizeDisturbance(sysLTI,a);
```

To deal with the affine dynamics, we perform a steady-state shift and simulate the steady-state system that has LTI dynamics. After performing the control synthesis steps, we compensate for this steady-state shift again to obtain the dynamics of the actual system.

Now, we can start with the synthesis steps. First, we reduce the 7D model to a 2D reduced-order model (see Eq. (3.8)) using function `ModelReduction` with `f = 0.098` and dimr=2.

```
% Perform model-order reduction
[sysLTIr, ~] = ModelReduction(sysLTI, dimr, f);
```

As mentioned in Section 3.3b, we use an interface function of the form (3.7c), which is selected using `int_f = 1` and compute the matrices P and Q using `ComputeProjection`. Next, we define the state and input spaces, and the output regions and atomic propositions for the reduced-order model as before.

To construct the finite-state abstraction of the reduced-order model, we first grid the input space with `lu = 3`.

```
% Construct abstract input space
[uhat,sysLTIr.U] = GridInputSpace(lu, sysLTIr.U, 'interface', ...
    int_f, 0.6, 0.175);
```

Here, we have chosen to use 60% of the input space for actuation and 17.5% for feedback. This leaves 22.5% for the $Qx_{r,t}$ part of the interface function, which is currently not guaranteed to be satisfied, but can be checked a posteriori.

Before constructing a finite-state abstraction of the reduced-order model, we reduce the state space to increase the computational speed. This step is currently only available for invariance specifications and is performed by `ReduceX`, which performs several backward iterations on the safety region P_1 to determine a good guess of the invariant set. This set is then used as the reduced state space. The construction of the finite-state abstraction of the reduced-order model is as before, except that we give the total number of grid cells as input `l`, instead of the number of grid cells in each direction.

```
% Reduce the state space to speed up computations
[sysLTIr, ~] = ReduceX(sysLTIr, sysLTIr.U{2}, P1, 'invariance', 5);
% Construct finite-state abstraction
l = [3000*3000];  % Total number of grid cells
tol=10^-6;
sysAbs = FSabstraction(sysLTIr, uhat, l, tol, DFA, ...
    'TensorComputation', true);
```

To relate the reduced-order finite-state model `sysAbs` to the original model `sysLTI`, we construct two simulation relations: relation `rel_1` with $\epsilon_1 = 0.2413$ between `sysLTI` and `sysLTIr`, and relation `rel_2` with $\epsilon_2 = 0.1087$ between `sysLTIr` and `sysAbs`:

```
% Compute MOR simulation relation
[rel_1, K, kernel] = QuantifySim(sysLTI, sysLTIr, epsilon_1, ...
    'MOR', sysAbs);
% Compute finite-state simulation relation
[rel_2] = QuantifySim(sysLTIr, sysAbs, epsilon_2);
% Combine simulation relations
rel = CombineSimRel(rel_1, rel_2, sysLTIr, sysAbs);
```

For model-order reduction, we have to explicitly define the coupling `kernel` matrix F, which is later used to compute the disturbance of the reduced-order model as $w_r = w + F(x - Px_r)$. For details see Section 2.4b.

Synthesizing and refining the controller are done as before and the satisfaction probability of the reduced-order model is shown in Figure 3.7 (obtained through `plotSatProb`). We simulate the controlled system $N_s = 6$ times, making sure the output is shifted with respect to the steady-state solution.

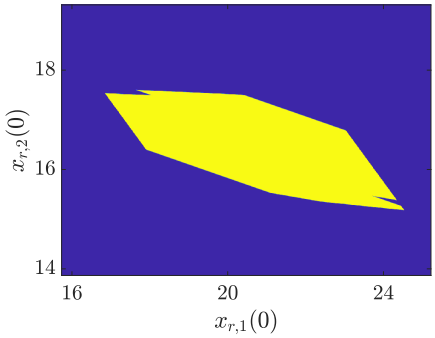


Figure 3.7: Robust satisfaction probability of the reduced-order model of the building automation system benchmark. Yellow and blue correspond to a probability of respectively 0.9195 and 0.

```
% Simulate controlled system Ns times
N_s = 6;
xsim = ImplementController(x0, N, Controller, Ns, 'MOR', sysLTIr, ...
    kernel);
```

The resulting trajectories can be evaluated using `plotTrajectories`. Some arbitrary trajectories are given in Figure 2.6c in Chapter 2 of this thesis.

3.4d Performance evaluation

The performance of `SySCoRe` is evaluated on the benchmarks mentioned above. The details of the benchmarks and their total run time and memory usage are reported in Table 3.2. The computation times per step are reported in Table 3.3. The data has been obtained on a computer with a 2.3 GHz Quad-Core Intel Core i5 processor and 16 GB 2133 MHz memory by taking the average over 5 computations. Here, we observed a maximum 6% standard deviation.

Table 3.2 can be used to compare the different benchmarks with respect to the computations performed by `SySCoRe`. The main difference between the running example and the package delivery benchmark is the DFA. The DFA of the package delivery benchmark requires more memory, however, the increase in computation time is small. Due to the simple DFA of the running example, we only compute the satisfaction probability for the initial DFA state. This will not suffice for the package delivery benchmark, which is the reason that more computation time is spent on steps (4)-(6) compared to the running example (see Table 3.3). The computation time for the nonlinear benchmark is large, however, the memory usage remains reasonable. The increase in computation time is mainly due to the fine gridding. We can also see in Table 3.3 that the similarity quantification takes a considerable amount of time. This is because we perform this step for each partition separately (1600 times in this case). For higher-dimensional systems that require model-order reduction (building automation system benchmark), the computation time and memory usage increase substantially, mainly due to the fact that the similarity quantification has to be performed multiple times.

Table 3.2: An overview of the different benchmarks in **(a)** and their total computation time in seconds (s) and memory usage in megabytes (MB) in **(b)**. The details of the computation times for each step are reported in Table 3.3. *Dim.* and *Comp.* are abbreviations for *Dimension* and *Computation*, respectively. The size of the specification refers to the number of states of the DFA. The benchmark names are abbreviated as follows. Running example (RE), Package Delivery (PD), Van der Pol oscillator (vdPol), and Building automation system (BAS).

(a) Overview of the benchmarks

	System		MOR	Specification		
	Dynamics	Dim.		Type	Time horizon	Size
RE	Linear	2	No	Reach-avoid	Unbounded	3
PD	Linear	2	No	Reach-avoid	Unbounded	3
VdPol	Nonlinear	2	No	Safety, reachability	Unbounded	3
BAS	Linear	7	Yes	Safety	Bounded	8

(b) Total computation time and memory usage

	Comp. time (s)	Memory (MB)
RE	7.94	27.53
PD	11.02	133.40
VdPol	3191.6	178.83
BAS	122.05	5365.6

Table 3.3 shows that the similarity quantification of step (3) requires the most computation time, followed by the finite-state abstraction of step (2). The large computation time of the similarity quantification is due to solving an optimization problem constrained by parameterized matrix inequalities that could be non-convex. For most abstraction-based approaches in the literature, the main bottleneck is the finite-state abstraction and control synthesis. This shows the efficiency of our tensor-based implementations. It should also be noted that the efficiency of the tensor computations is also exploited in the control synthesis step.

3.4e Comparison to existing tools

A comparison of the results on the benchmarks obtained by `SySCoRe` and current tools is given in Table 3.4. The package delivery benchmark has a complex DFA and cannot be handled natively by tools `AMYTISS`, `FAUST`, and `StocHy` (see Table 3.4a). `SReachTools` can only handle safety specifications and is not applicable to this benchmark. The Van der Pol oscillator benchmark poses significant challenges for the tools due to its nonlinear dynamics, as reported in Table 3.4b. Only `AMYTISS` can solve a benchmark that resembles this one as considered in Abate et al. (2021) with multiplicative noise instead of additive noise. `AMYTISS` can only handle systems with a bounded disturbance, hence it cannot directly solve the benchmark as presented here.

Table 3.3: Computation times for the different steps (1)-(6) in seconds and as percentage of the total runtime. Steps (1)-(6) correspond to (1) translating the specification, (2) finite-state abstraction, (3) similarity quantification, (4) synthesizing a controller, (5) control refinement, and (6) deployment. Step (5) is almost instantaneous (≈ 0.001 s), therefore, we have taken the computation times of steps (5) and (6) together. The benchmark names are abbreviated as follows. Running example (RE), Package Delivery (PD), Van der Pol oscillator (vdPol), and Building automation system (BAS).

	Step (1)	Step (2)	Step (3)	Step (4)
RE	0.26s (3.3%)	1.32s (16.6%)	5.59s (70.4%)	0.51s (6.39%)
PD	0.28s (2.6%)	1.66s (15.0%)	6.19s (56.2%)	1.71s (15.5%)
VdPol	0.59s (0.02%)	1440s (45.1%)	1748.6s (54.8%)	2.85s (0.09%)
BAS	0.36s (0.3%)	4.80s (3.9%)	67.9s (55.7%)	37.33s (30.6%)

	Step (5)and (6)	Total comp. time
RE	0.20s (2.57%)	7.94s (100%)
PD	0.70s (6.37 %)	11.02s (100%)
VdPol	1.42s (0.04%)	3191.6s (100%)
BAS	9.19s (7.53%)	122.05s (100%)

Table 3.4: Results of the benchmarks for different tools. Here, *n.a.* means that a tool is *not applicable* and *n.s.* means that the current version of the tool does *not natively support* the computations on the benchmark, but that we do not see fundamental limitations hindering such an extension. To compare the tools we exclude the deployment of the controller (step (6)) since this step is not performed by the other tools.

(a) Package delivery benchmark.

Tool	Run time (s)
`AMYTISS`	n.s.
`FAUST`	n.s.
`SReachTools`	n.a.
`StocHy`	n.s.
`SySCoRe`	10.32

(b) Van der Pol benchmark.

Tool	Run time (s)
`AMYTISS`	n.s.
`FAUST`	n.a.
`SReachTools`	n.a.
`StocHy`	n.a.
`SySCoRe`	3190.2

(c) Building automation benchmark.

Tool	Run time (s)	Max. reach probability
`AMYTISS`	312.14	≈ 0.8
`FAUST`	n.s.	n.s.
`SReachTools`	4.59	≥ 0.99
`StocHy`	≥ 335.87	$\geq 0.80 \pm 0.23$
`SySCoRe`	102.86	≥ 0.92

The benchmark on the building automation system can be solved by AMYTISS, SReachTools, and StocHy without being able to use model-order reduction. This benchmark considers a stochastic safety problem and the performance of multiple tools is compared in Abate et al. (2020) and Abate et al. (2021). Table 3.4c reports the results of SySCoRe together with the results from running the repeatability packages of Abate et al. (2020) and Abate et al. (2021) on a computer with a 2.3 GHz Quad-Core Intel Core i5 processor and 16 GB 2133 MHz memory. For StocHy, there was no repeatability package available, however, since the computational power of the CPU used for the results in Abate et al. (2021) was more than our computer, we included the results of Abate et al. (2021) as a lower bound on the computation time required by StocHy. Note that this benchmark belongs to the class of partially degenerate systems (Soudjani and Abate 2013b). The formulation of the abstraction error for this class is available but the current version of FAUST does not natively support partially degenerate systems. With respect to the computation time, SReachTools performs best, and AMYTISS and StocHy require a longer computation time. Though from the results in Abate et al. (2021), we see that AMYTISS could be faster than the current implementation of SySCoRe when parallel execution within CPUs is available (this parallel computation will be exploited in future versions of SySCoRe). With respect to accuracy, both AMYTISS and StocHy obtain a maximum reachability probability smaller than SySCoRe, while SReachTools still outperforms SySCoRe. This shows that SReachTools is the best option for this benchmark, which is expected since it is developed exactly for linear systems and stochastic reach-avoid problems with small disturbances.

3.5 Summary and extensions

This chapter described the first release of SySCoRe, a tool that excels at control synthesis problems for systems with large (unbounded) stochastic disturbances and temporal specifications with possibly unbounded time horizon. It combines reduced-order models and finite abstractions with formal guarantees obtained by coupled stochastic simulation relations. SySCoRe substantially extends the class of models and temporal specifications that current tools can handle for control synthesis. Furthermore, the modular development of SySCoRe allows ease of use and facilitates future extensions. The efficient implementation of tensor computations in SySCoRe allows for fast computations, which can be exploited further by including more parallel computations as done in AMYTISS.

An important direction for future releases is to extend the current implementation of model-order reduction to piecewise-affine systems, such that it can also be applied to nonlinear systems. Currently, only Gaussian disturbances are implemented in SySCoRe, however, extensions to other distributions are underway and require deriving new inequality constraints for the optimization problem solved in the similarity quantification. The computation time of the similarity quantification is large due to solving optimization problems constrained by parameterized matrix inequalities that could be non-convex. We are working on improving the efficiency of solving this optimization.

The modular implementation of `SySCoRe` can be utilized to integrate model-order reduction with discretization-free approaches such as the kernel method of `SReachTools` (Vinod et al. 2019; Thorpe et al. 2021) and the barrier certificates (Jagtap et al. 2020), or to perform synthesis for stochastic systems with parametric uncertainty (as in Chapter 5). To get non-trivial lower bounds, `SySCoRe` currently requires fine-tuning the hyperparameters (e.g., the grid size and the output deviation). It is of interest to automatically design these parameters or to provide guidelines to the user on the appropriate values depending on the case study.

4

Specification-guided temporal logic control for stochastic systems: a multi-layered approach

The design of provably correct controllers for continuous-state stochastic systems with respect to temporal logic specifications can sometimes be done directly but generally requires an approximate solution. Hence, the control design often relies on an approximation step, referred to as *abstraction*. Different circumstances, such as different models and/or specifications, lead to different approaches that are either based on a discretization of the state space (*discretization-based* approach) or not (*discretization-free* approach). Besides the choice of an approach, the accuracy of the approximated probability of satisfying a specification can depend on parameter choices in the approach, such as required precision.

In this chapter, we develop a method that naturally has the flexibility to switch between parameters or abstraction techniques to reduce the conservatism of the approximated satisfaction probability. First, we develop a discretization-based approach with variable precision by switching between layers with different precision parameters. By also extending this *multi-layered approach* to switching between layers with respectively discretization-based models and discretization-free models, we achieve an efficient implementation that is both accurate and performs well concerning computation time and memory usage. At the end of this chapter, we illustrate the benefit of the multi-layered approach in several case studies.

4.1 Introduction

Stochastic difference equations are often used to model the behavior of complex systems that are operating in an uncertain environment, such as autonomous vehicles, airplanes, and drones. In this work, we are interested in automatically designing controllers for which we can give guarantees on the functionality of stochastic systems with respect to temporal logic specifications. Such automatic control synthesis is often referred to as correct-by-design control synthesis.

There are multiple techniques within this area, where we distinguish between methods that construct a finite-state abstraction of the original continuous-state model by discretizing the state space (Belta et al. 2017; Lavaei et al. 2022a), and methods that do not rely on a space discretization (Vinod et al. 2019; Jagtap et al. 2020). The former is referred to as *discretization-based* techniques, and the latter as *discretization-free* techniques. For certain models and specifications, the most suitable approach is discretization-based while for a different situation, the most suitable approach is discretization-free (Abate et al. 2020; Abate et al. 2021). Even for the same model, a different specification could lead to a different approach performing best. Furthermore, within a method, the accuracy of the computed probability of satisfying a specification can depend on parameter choices in the synthesis approach, such as precision.

As an example, consider a reach-avoid specification and a system whose labeled output space is illustrated in Figure 4.1, where we want to avoid the red area and reach the green area from an arbitrary initial point in the configuration space. We are interested in synthesizing a controller, or even simpler, planning a path that satisfies this specification subject to the dynamics of a given system, while also computing the probability that the specification is satisfied, referred to as the *satisfaction probability*. In general, we can reason that certain circumstances make it possible to accurately compute the satisfaction probability by constructing a nominal path surrounded by a tube containing possible paths with a high probability, as is common with discretization-free approaches. Such circumstances are for example 1) a large state space, with a small avoid region and a large goal region that is far away from the avoid region, or 2) the distribution associated with the disturbance has a small variance. However, the usage of a *single* nominal path reduces the achievable accuracy of the approximated satisfaction probability.

Discretization-based approaches, such as Soudjani et al. (2015), Cauchi and Abate (2019), and Cauchi et al. (2019) overcome this limitation by (often manually) refining the grid until the accuracy of the satisfaction probability is sufficient. However, in practice, they are limited by the available memory and computation time. Put differently, when the available memory is limited, large-scale systems pose enormous challenges for discretization-based approaches, and discretization-free approaches are the only option. In this chapter, we are interested in the boundary cases, where it is not immediately obvious what the most suitable approach is. For those cases, we develop an approach that naturally switches between different techniques. More specifically, in situations where a discretization-free technique gives acceptable accuracy, we use those techniques, and only when necessary we use the discretization-based technique. Thereby effectively combining the approaches

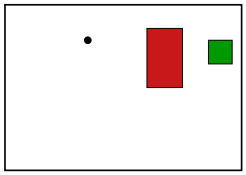


Figure 4.1: Example of an output space for a reach-avoid specification. The red area is the area to avoid, the green area is the goal region, and the black dot is an initial state.

to compute a single approximate satisfaction probability. In our approach, the specification determines which technique is necessary for which situation, hence we refer to our approach as *specification-guided*.

Combining both abstraction techniques while maintaining the guarantees is challenging and involves multiple changes. One of them is the fact that they have a different precision. Therefore, we initially leave the different abstraction techniques to the side and first develop a discretization-based approach, where we switch between different simulation relations. More specifically, we build upon Chapter 2 of this thesis and switch between different layers each containing its own simulation relation with specific precision parameters. Next, we extend the multi-layered methodology to feature layers with different abstraction techniques, that are discretization-based or discretization-free.

Literature. In the area of discretization-based techniques, multiple methods are related to achieving a variable precision, namely through non-uniform partitioning of the state space. For deterministic systems there exist methods that construct a non-uniform discretization grid (Tazaki and Imura 2010; Ren and Dimarogonas 2019). More specifically, they give an approximate bisimulation relation for variable precision (or dynamic) quantization and develop a method to locally refine a coarse finite-state abstraction based on the system dynamics. Furthermore, for deterministic systems, there also exist methods known as multi-layered discretization-based control synthesis. They focus on maintaining multiple finite-state abstraction layers with different precision, where they use the coarsest finite-state abstraction when possible (Cámara et al. 2011a; Cámara et al. 2011b; Hsu et al. 2018; Girard and Gössler 2020). For stochastic models, non-uniform partitioning of the state space has been introduced for the purpose of verification (Soudjani and Abate 2013a) and for verification and control synthesis in the software tools FAUST^2 (Soudjani et al. 2015) and StocHy (Cauchi and Abate 2019; Cauchi et al. 2019). The latter builds on interval Markov decision processes. In this chapter we consider different simulation relations that are used to quantify the similarity between a continuous-state model and its finite-state abstraction. Hence, we go a step beyond non-uniform partitioning and achieve a variable precision differently.

We start with allowing variable precision by presenting a simulation relation that contains multiple precision layers. For such relations, we develop an algorithm to determine a switching strategy and a robust dynamic programming approach such that we can compute a lower bound on the satisfaction probability of complex specifications. Besides that, we extend our multi-layered methodology to allow for layers with a discretization-free technique and layers with a discretization-based technique, therefore, improving the efficiency and scalability of our method.

In the next section, we discuss the considered model and specifications, and formulate the problem statement for a general class of nonlinear stochastic difference equations. In Section 4.3, we discuss the approach and structure of this chapter in a detailed manner. In Section 4.4, we discuss the multi-layered approach and define the multi-layered simulation relation for variable precision. In Section 4.5, we extend the multi-layered method to allow discretization-free layers and discretization-based layers. Finally, in Section 4.6, we apply our method to multiple case studies and analyze its benefit. We end this chapter with a conclusion.

4.2 Problem formulation

Notation. The Borel measurable space of a set $\mathbb{X} \subset \mathbb{R}^n$ is denoted by $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$, with $\mathcal{B}(\mathbb{X})$ the Borel sigma-algebra on $\mathbb{X}$. A probability measure $\mathbb{P}$ over this space has realization $x \sim \mathbb{P}$, with $x \in \mathbb{X}$. Furthermore, a time update of a state variable x is interchangeably denoted by $x(t+1), x_{t+1}$ or x^+.

Model. We consider general Markov decision processes (gMDP) as defined in Haesaert et al. (2017b) and Haesaert and Soudjani (2020) as follows.

Definition 4.1 (general Markov decision process (gMDP))**.** *A gMDP is a tuple* $\mathbf{M} = (\mathbb{X}, x_0, \mathbb{U}, \mathbf{t}, \mathbb{Y}, h)$ *with state space* $\mathbb{X}$*, initial state* $x_0 \in \mathbb{X}$*, input space* $\mathbb{U}$*, and with output space* $\mathbb{Y}$ *and measurable output map* $h : \mathbb{X} \to \mathbb{Y}$*. The transitions kernel* $\mathbf{t} : \mathbb{X} \times \mathbb{U} \times \mathcal{B}(\mathbb{X}) \to [0, 1]$ *assigns to each state* $x \in \mathbb{X}$ *and input* $u \in \mathbb{U}$ *a probability measure* $\mathbf{t}(\cdot \mid x, u)$ *over* $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$*.*

We consider output space $\mathbb{Y}$ to be a metric space and denote the class of all models with the same metric output space $(\mathbb{Y}, \mathbf{d}_{\mathbb{Y}})$ as $\mathcal{M}_{\mathbb{Y}}$. The behavior of a gMDP with n_x-dimensional state space $\mathbb{X} \subset \mathbb{R}^{n_x}$ can equivalently be described by a stochastic difference equation. We consider discrete-time systems whose dynamics are described by a stochastic difference equation with Gaussian disturbance

$$\mathbf{M} : \begin{cases} x(t+1) = f(x(t), u(t), w(t)) \\ y(t) = h(x(t)), \quad \forall t \in \{0, 1, 2, \dots\}, \end{cases} \tag{4.1}$$

with state $x(t) \in \mathbb{X}$, input $u(t) \in \mathbb{U}$, disturbance $w(t) \in \mathbb{W} \subseteq \mathbb{R}^{n_w}$, output $y(t) \in \mathbb{Y}$, and with measurable functions $f : \mathbb{X} \times \mathbb{U} \times \mathbb{W} \to \mathbb{X}$ and $h : \mathbb{X} \to \mathbb{Y}$. The system is initialized with $x(0) = x_0 \in \mathbb{X}$, and $w(t)$ is an independently and identically distributed sequence with realizations $w \sim \mathbb{P}_w = \mathcal{N}(\mu, \Sigma)$.

A finite path of the model (4.1) is a sequence $\boldsymbol{\omega}_{[0,t]} := x_0, u_0, x_1, u_1, \dots, x_t$. An infinite path is a sequence $\boldsymbol{\omega} := x_0, u_0, \dots$. The paths start at $x_0 = x(0)$ and are built up from realizations $x_{t+1} = x(t+1)$ based on (4.1) given a state $x(t) = x_t$, input $u(t) = u_t$ and disturbance $w(t)$ for each time step t. We denote the state trajectories as $\mathbf{x} = x_0, x_1, \dots$, with associated suffix $\mathbf{x}_t = x_t, x_{t+1}, \dots$. The output y_t contains the variables of interest for the performance of the system and for each state trajectory, there exists a corresponding output trajectory $\mathbf{y} = y_0, y_1, \dots$, with associated suffix $\mathbf{y}_t = y_t, y_{t+1}, \dots$.

A control strategy is a sequence $\boldsymbol{\mu} = (\mu_0, \mu_1, \mu_2, \dots)$ of maps $\mu_t(\boldsymbol{\omega}_{[0,t]}) \in \mathbb{U}$ that assigns for each finite path $\boldsymbol{\omega}_{[0,t]}$ an input $u(t) = u_t$. The control strategy is a Markov policy if μ_t only depends on x_t, that is $\mu_t : \mathbb{X} \to \mathbb{U}$. We refer to a Markov policy as *stationary* if μ_t does not depend on the time index t, that is $\boldsymbol{\mu} = (\mu, \mu, \dots)$ for some μ. In this work, we are interested in control strategies denoted as $\mathbf{C}$ that can be represented with finite memory, that is, policies that are either time-stationary Markov policies or have finite internal memory. A policy with finite internal memory first maps the finite state execution of the system to a finite set (memory), followed by computing the input as a function of the system state and the memory state. By doing so, we can satisfy temporal specifications on the system trajectories.

Specifications. To express (unbounded time-horizon) temporal logic specifications, we use the syntactically co-safe linear temporal logic language (scLTL) (Kupferman and Vardi 2001; Belta et al. 2017). This language consists of atomic propositions $p_1, p_2, \dots p_N$ that are true or false. The set of atomic propositions and the corresponding alphabet are denoted by $\text{AP} = \{p_1, \dots, p_N\}$ and 2^{AP}, respectively. Each letter $\pi \in 2^{\text{AP}}$ contains the set of atomic propositions that are true. A (possibly infinite) string of letters forms a word $\boldsymbol{\pi} = \pi_0, \pi_1, \dots$, with associated suffix $\boldsymbol{\pi}_t = \pi_t, \pi_{t+1}, \dots$. The output trajectory $\mathbf{y} = y_0, y_1, \dots$ of a system (4.1) is mapped to the word $\boldsymbol{\pi} = L(y_0), L(y_1), \dots$ using labeling function $L : \mathbb{Y} \to 2^{\text{AP}}$ that translates each output to a specific letter $\pi_t = L(y_t)$. Similarly, suffixes $\mathbf{y}_t$ are translated to suffix words $\boldsymbol{\pi}_t$. By combining atomic propositions with logical operators, the language of scLTL can be defined as follows.

Definition 4.2 (scLTL syntax)**.** *An scLTL formula ϕ is recursively defined over a set of atomic propositions as*

$$\phi ::= p \,|\, \neg p \,|\, \phi_1 \wedge \phi_2 \,|\, \phi_1 \vee \phi_2 \,|\, \bigcirc \phi \,|\, \phi_1 \,\mathsf{U}\, \phi_2, \tag{4.2}$$

with atomic proposition $p \in \text{AP}$.

The semantics of this syntax can be given for the suffixes $\boldsymbol{\pi}_t$. An atomic proposition $\boldsymbol{\pi}_t \models p$ holds if $p \in \pi_t$, while a negation $\boldsymbol{\pi}_t \models \neg\phi$ holds if $\boldsymbol{\pi}_t \not\models \phi$. Furthermore, a conjunction $\boldsymbol{\pi}_t \models \phi_1 \wedge \phi_2$ holds if both $\boldsymbol{\pi}_t \models \phi_1$ and $\boldsymbol{\pi}_t \models \phi_2$ are true, while a disjunction $\boldsymbol{\pi}_t \models \phi_1 \vee \phi_2$ holds if either $\boldsymbol{\pi}_t \models \phi_1$ or $\boldsymbol{\pi}_t \models \phi_2$ is true. Also, a next statement $\boldsymbol{\pi}_t \models \bigcirc\phi$ holds if $\boldsymbol{\pi}_{t+1} \models \phi$. Finally, an until statement $\boldsymbol{\pi}_t \models \phi_1 \,\mathsf{U}\, \phi_2$ holds if there exists an $i \in \mathbb{N}$ such that $\boldsymbol{\pi}_{t+i} \models \phi_2$ and for all $j \in \mathbb{N}, 0 \leq j < i$, we have $\boldsymbol{\pi}_{t+j} \models \phi_1$. A trajectory satisfies a specification if the generated word $\boldsymbol{\pi}_0 = \boldsymbol{\pi} = L(\mathbf{y})$ satisfies the specification, i.e., $\boldsymbol{\pi}_0 \models \phi$.

Correct-by-design control synthesis focuses on designing a controller $\mathbf{C}$, for model $\mathbf{M}$ and specification ϕ, such that the controlled system $\mathbf{M} \times \mathbf{C}$ satisfies the specification, denoted as $\mathbf{M} \times \mathbf{C} \models \phi$. For stochastic systems, we are interested in the satisfaction probability, denoted as $\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi)$. The general problem considered in this chapter is the following.

Problem. Given model $\mathbf{M}$ as in (4.1), an scLTL specification ϕ, and a probability $p_\phi \in [0, 1]$, find a controller $\mathbf{C}$, such that

$$\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi) \geq p_\phi. \tag{4.3}$$

4.3 Multi-layered approach

In discretization-based methods, a finite-state approximation of the original model called the abstract model, is used to both synthesize a controller and to compute a lower bound on the satisfaction probability. By refining the obtained controller in a suitable manner the lower bound on the satisfaction probability is maintained for the original model. Crucial steps in this approach are the *similarity quantification* between the original model and its abstraction, and the *dynamic programming* approach to compute the lower bound on the satisfaction probability and the associated controller.

In this chapter, we build upon the similarity quantification and robust dynamic programming approach as in Chapter 2 of this thesis. More specifically, this chapter is structured as follows.

- **Standard approach.** We first recap the standard approach in which *one finite-state model* and *one similarity quantification* are used.
- **Homogeneous layers.** Next, we introduce a multi-layered approach with variable precision. Here, we use multiple homogeneous layers, in which *one finite-state model* with multiple layers and *multiple similarity quantifications* is used.
- **Heterogeneous layers.** As a final contribution, we combine the multi-layered approach with a model that is constructed using a discretization-free technique. Hence, we use *different abstract models*.

Let us start with a brief recap of the standard approach based on Chapter 2.

4.3a Standard approach

Similarity quantification. Suppose that we have approximated the continuous-state model as given in (4.1), with the following finite-state abstract model

$$\hat{\mathbf{M}} : \begin{cases} \hat{x}(t+1) = \hat{f}(\hat{x}(t), \hat{u}(t), \hat{w}(t)) \\ \hat{y}(t) = \hat{h}(\hat{x}(t)), \end{cases} \tag{4.4}$$

with state $\hat{x} \in \hat{\mathbb{X}}$, initialized at $\hat{x}(0) = \hat{x}_0$ and with input $\hat{u} \in \hat{\mathbb{U}}$, output $\hat{y} \in \mathbb{Y}$ and disturbance $\hat{w} \in \hat{W}$. Furthermore, we have functions $\hat{f} : \hat{\mathbb{X}} \times \hat{\mathbb{U}} \times \hat{\mathbb{W}} \to \hat{\mathbb{X}}$ and $\hat{h} : \hat{\mathbb{X}} \to \mathbb{Y}$, and $\hat{w}(t)$ is an independently and identically distributed sequence with realizations $\hat{w} \sim \mathbb{P}_{\hat{w}}$.

We use the abstract model in (4.4) to compute a lower bound on the probability of satisfying specification ϕ. To preserve this lower bound, we need to quantify the similarity between the models (4.1) and (4.4), graphically represented as in Figure 4.2.

We consider an approximate simulation relation as in Definition 2.4 in Chapter 2 of this thesis, since this is suitable for specifications that are unbounded in time.

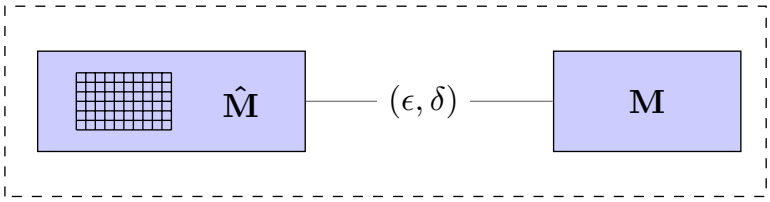


Figure 4.2: Graphical representation of the standard approach.

This method for quantifying the similarity is based on an explicit coupling between the models (4.1) and (4.4), which allows for analyzing how close the probability transitions are. Definition 2.4 in Chapter 2 is based on the following observation. Given a simulation relation $\mathcal{R} \subset \hat{\mathbb{X}} \times \mathbb{X}$, for all pairs of states inside this relation $(\hat{x}, x) \in \mathcal{R}$, and for all inputs $\hat{u} \in \hat{\mathbb{U}}$ we can quantify a lower bound on the probability that the next pair of states is also inside this simulation relation, i.e. $(\hat{x}^+, x^+) \in \mathcal{R}$. Hence, for all states $(\hat{x}, x) \in \mathcal{R}$, $\forall \hat{u} \in \hat{\mathbb{U}}$, we require that

$$(\hat{x}^+, x^+) \in \mathcal{R} \tag{4.5}$$

has a lower bound on its probability denoted by $1 - \delta$ under the transitions in the coupled model.

The approximate simulation relation in Definition 2.4 does not only quantify the similarity based on the probability deviation δ but also on output deviation ϵ. Furthermore, if $\hat{\mathbf{M}}$ is (ϵ, δ)-stochastically simulated by $\mathbf{M}$, then this is denoted as $\hat{\mathbf{M}} \preceq^{\delta}_{\epsilon} \mathbf{M}$. In Chapter 2 of this thesis, it has been shown that ϵ and δ have a trade-off. Increasing ϵ decreases the achievable δ and vice versa. To compute the deviation bounds (ϵ, δ), an optimization problem constrained by a set of parameterized matrix inequalities is given in Chapter 2 of this thesis.

Robust dynamic programming. After quantifying the similarity, we can synthesize a robust controller and compute a lower bound on the satisfaction probability based on the deviation bounds ϵ and δ. Since we consider specifications written using scLTL, we can formulate this problem as a reachability problem that can be solved as a dynamic programming problem.

For control synthesis purposes an scLTL specification (4.2) can be written as a deterministic finite-state automaton (DFA), defined by the tuple $\mathcal{A} = (Q, q_0, \Sigma_{\mathcal{A}}, \tau_{\mathcal{A}}, Q_f)$. Here, Q, q_0, and Q_f denote the set of states, initial state, and set of accepting states, respectively. Furthermore, $\Sigma_{\mathcal{A}} = 2^{\mathrm{AP}}$ denotes the input alphabet and $\tau_{\mathcal{A}} : Q \times \Sigma_{\mathcal{A}} \to Q$ is a transition function. For any scLTL specification ϕ, there exists a corresponding DFA $\mathcal{A}_\phi$ such that the word $\boldsymbol{\pi}$ satisfies this specification $\boldsymbol{\pi} \models \phi$, iff $\boldsymbol{\pi}$ is accepted by $\mathcal{A}_\phi$. Here, acceptance by a DFA means that there exists a trajectory $q_0 q_1 q_2 \dots q_f$, that starts with q_0, evolves according to $q_{t+1} = \tau_{\mathcal{A}}(q_t, \pi_t)$, and ends at $q_f \in Q_f$. We can therefore reason about the satisfaction of specifications over $\mathbf{M}$ by analyzing its product composition with $\mathcal{A}_\phi$ (Tkachev et al. 2013) denoted as $\mathbf{M} \otimes \mathcal{A}_\phi$. This composition yields a stochastic system with states $(x_t, q_t) \in \mathbb{X} \times Q$ and input u_t. Given input u_t, the stochastic transition from x_t to x_{t+1} of $\mathbf{M}$ is extended to the transition from (x_t, q_t) to (x_{t+1}, q_{t+1}) with $q_{t+1} = \tau_{\mathcal{A}_\phi}(q_t, L(h(x_t)))$.

Hence, by computing a Markov policy over $\mathbf{M} \otimes \mathcal{A}_\phi$ that solves this reachability problem, we obtain a controller with respect to specification ϕ, that has a finite memory (Belta et al. 2017). In Section 2.6, we have shown that for any gMDP (and other equivalent representations), this reachability problem can be rewritten as a dynamic programming (DP) problem.

Given Markov policy $\boldsymbol{\mu} = (\mu_0, \mu_1, \ldots, \mu_N)$ with time horizon N for $\mathbf{M} \otimes \mathcal{A}_\phi$, define the time-dependent value function $V_N^{\boldsymbol{\mu}}$ as

$$V_N^{\boldsymbol{\mu}}(x,q) = \mathbb{E}_{\boldsymbol{\mu}}\left[\sum_{k=1}^{N} \mathbf{1}_{Q_f}(q_k) \prod_{j=1}^{k-1} \mathbf{1}_{Q \setminus Q_f}(q_j) \middle| (x_0, q_0) = (x,q) \right] \tag{4.6}$$

with indicator function $\mathbf{1}_E(q)$ equal to 1 if $q \in E$ and 0 otherwise. Since $V_N^{\boldsymbol{\mu}}(x,q)$ expresses the probability that a trajectory generated under $\boldsymbol{\mu} : \mathbb{X} \times Q \to \mathbb{U}$ starting from (x,q) will reach the target set Q_f within the time horizon $[1, \ldots, N]$, it also expresses the probability that specification ϕ will be satisfied in this time horizon. Next, express the associated DP operator

$$\mathbf{T}^u(V)(x,q) := \mathbb{E}_u\big(\max\left\{ \mathbf{1}_{Q_f}(q^+), V(x^+, q^+) \right\} \big), \tag{4.7}$$

with $u = \mu(x,q)$, $x^+ := x(t+1)$ evolving according to (4.1), and with the implicit transitions $q^+ = \tau_{\mathcal{A}_\phi}(q, L(h(x^+)))$. Consider a policy $\boldsymbol{\mu}_k = (\mu_{k+1}, \ldots \mu_N)$ with time horizon $N-k$, then it follows that $V_{N-k+1}^{\boldsymbol{\mu}_{k-1}}(x,q) = \mathbf{T}^{\mu_k}(V_{N-k}^{\boldsymbol{\mu}_k})(x,q)$, $\forall (x,q) \in \mathbf{M} \otimes \mathcal{A}_\phi$. Thus if $V_{N-k}^{\boldsymbol{\mu}_k}$ expresses the probability of reaching Q_f within $N-k$ steps, then $\mathbf{T}^{\mu_k}(V_{N-k}^{\boldsymbol{\mu}_k})$ expresses the probability of reaching Q_f within $N-k+1$ steps with policy $\boldsymbol{\mu}_{k-1}$. It follows that for a stationary policy $\boldsymbol{\mu}$, the infinite-horizon value function can be computed as $V_\infty^{\boldsymbol{\mu}} = \lim_{N\to\infty} (\mathbf{T}^{\boldsymbol{\mu}})^N V_0$ with $V_0 \equiv 0$. Furthermore, the optimal DP operator $\mathbf{T}^*(\cdot) := \sup_\mu \mathbf{T}^\mu(\cdot)$ can be used to compute the optimal converged value function V_∞^*. The corresponding satisfaction probability can now be computed as

$$\mathbb{P}^{\boldsymbol{\mu}^*} := \max(\mathbf{1}_{Q_f}(\bar{q}_0), V_\infty^*(x_0, \bar{q}_0)),$$

with $\bar{q}_0 = \tau_{\mathcal{A}_\phi}(q_0, L(h(x_0))$ and is also denoted as $\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi)$. When the policy $\boldsymbol{\mu}^*$, or equivalently the controller $\mathbf{C}$, yields a satisfaction probability higher than p_ϕ, then (4.3) is satisfied and the synthesis problem is solved.

Due to the continuous states of $\mathbf{M}$, the DP formulation above cannot be computed for the original model $\mathbf{M}$, instead, we use abstract model $\hat{\mathbf{M}}$. As a consequence, we have to take deviation bounds ϵ and δ into account. For a fixed precision, this robust DP operator is introduced by Haesaert and Soudjani (2020). We repeat it here for completeness:

$$\mathbf{T}^{\hat{u}}(V)(\hat{x},q) = \mathbf{L}\big(\mathbb{E}_{\hat{u}}\big(\min_{q^+ \in Q_\epsilon^+} \max\left\{ \mathbf{1}_{Q_f}(q^+), V(\hat{x}^+, q^+) \right\} \big) - \delta\big), \tag{4.8}$$

with $\hat{u} = \mu(\hat{x}, q)$ and $\mathbf{L} : \mathbb{R} \to [0,1]$ being a truncation function defined as $\mathbf{L}(\cdot) := \min(1, \max(0, \cdot))$, and with

$$Q_\epsilon^+(q, \hat{y}^+) := \left\{ \tau_{\mathcal{A}}(q, L(y^+)) \mid ||y^+ - \hat{y}^+|| \le \epsilon \right\}. \tag{4.9}$$

Note that compared to (4.7), in (4.8) we subtract δ from the expected value to take the deviation in probability into account. Similarly, the deviation in the outputs

ϵ can give different labels for the outputs y and $\hat{y}$. Therefore, we minimize the probability with respect to the possible DFA states associated with these labels. This is done through Q_ϵ^+ as in (4.9). A graphical representation of the value iteration using this robust DP operator is given in Figure 4.3.

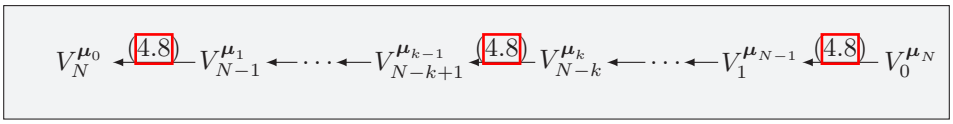


Figure 4.3: Graphical representation of the value iteration with a standard robust DP operator. The arguments $(\hat{x}, q)$ of the value functions are omitted for simplicity.

The value function gives the probability of satisfying the specification in 1 to ∞ time step, by including the satisfaction at x_0, we can compute the robust satisfaction probability, that is

$$\mathbb{R}^{\boldsymbol{\mu}} := \max(\mathbf{1}_{Q_f}(\bar{q}_0), V_\infty^{\boldsymbol{\mu}}(\hat{x}_0, \bar{q}_0)), \tag{4.10}$$

with $\bar{q}_0 = \tau_{\mathcal{A}_\phi}(q_0, L(h(x_0)))$ and $\hat{x}_0 \in \mathcal{R}^{-1}(x_0)$. The robust satisfaction probability as in (4.10) is also denoted as $\mathbb{R}^{\boldsymbol{\mu}}_{\epsilon,\delta}(\hat{\mathbf{M}} \times \hat{\mathbf{C}} \models \phi)$ to emphasize that it is computed on $\hat{\mathbf{M}} \times \hat{\mathbf{C}}$ by taking (ϵ, δ) into account.

In Haesaert and Soudjani (2020) they also show that the robust DP operator in (4.8) satisfies some important properties. The first property is that it is monotonically increasing. Note that a Bellman-operator $\mathbf{T}(\cdot)$ is *monotonically increasing* if for any two functions V and W for which we have that $\forall(\hat{x}, q) \in \hat{\mathbb{X}} \times Q$: $V(\hat{x}, q) \geq W(\hat{x}, q)$, it holds that

$$\mathbf{T}^{\hat{u}}(V)(\hat{x}, q) \geq \mathbf{T}^{\hat{u}}(W)(\hat{x}, q).$$

Following Lemma 1 in Haesaert and Soudjani (2020), we can now conclude the following.

Proposition 4.1. *The robust DP operator in* (4.8) *is monotonically increasing, and the series* $\left\{(\mathbf{T}^{\hat{u}})^k(V_0)\right\}_{k\geq 0}$ *with* $V_0 \equiv 0$ *is monotonically increasing and* point-wise converging. *Furthermore, the fixed point equation* $V_\infty^{\hat{u}} = \mathbf{T}^{\hat{u}}(V_\infty^{\hat{u}})$ *has a unique solution for* $\delta > 0$, *which is* $V_\infty^{\hat{u}} = \lim_{N\to\infty}(\mathbf{T}^{\hat{u}})^N V_0$ *with* $V_0 \equiv 0$.

Based on these properties and the robust satisfaction probability as in (4.10), following Theorem 4 in Haesaert and Soudjani (2020), we can now conclude the following.

Proposition 4.2. *Let* $\hat{\mathbf{M}} \otimes \mathcal{A}_\phi$ *and a controller* $\hat{\mathbf{C}}$ *be given. Then for any* $\mathbf{M}$ *with* $\hat{\mathbf{M}} \preceq_\epsilon^\delta \mathbf{M}$, *we can construct* $\mathbf{C}$, *such that* $\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi) \geq \mathbb{R}_{\epsilon,\delta}(\hat{\mathbf{M}} \times \hat{\mathbf{C}} \models \phi)$.

This concludes the recap of the most crucial concepts in the standard approach. Next, we give some details on the multi-layered approach developed in this chapter.

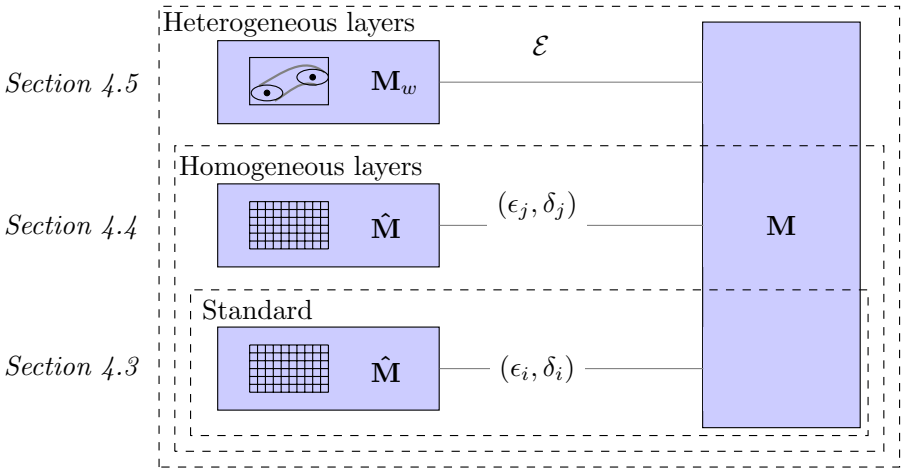


Figure 4.4: Overview of the structure of this chapter.

4.3b Multi-layered approach

The different approaches discussed in this chapter are graphically illustrated in Figure 4.4.

Homogeneous layers. First, we develop a multi-layered approach with homogeneous layers. To this end, we construct multiple layers containing the same abstract model $\hat{\mathbf{M}}$ as in (4.4), and quantify the similarity compared to the original model with different deviation bounds ϵ and δ. By doing so, we achieve a multi-layered simulation relation with variable precision. To perform the value iteration, we adjust the standard DP operator, such that it suits this multi-layered approach.

Heterogeneous layers. Next, we extend the multi-layered approach to heterogeneous layers. Here, one of the layers contains a discretization-free model constructed by computing reachability tubes, instead of a discretization-based abstract model $\hat{\mathbf{M}}$ as in (4.4). Since this model is substantially different than the abstract model $\hat{\mathbf{M}}$, the value iteration is performed differently as well. To perform the value iteration with heterogeneous layers, we have to combine the DP operator of the discretization-free layer with the DP operator of the multi-layered approach with homogeneous layers.

Overview. The main part of this chapter is structured as illustrated in Figure 4.4. In Section 4.4, we define a multi-layered simulation relation, derive the multi-layered DP operator, and describe a practical approach to efficiently implement the value iteration. Furthermore, we discuss how to obtain a controller and give an efficient implementation for the similarity quantification of linear systems. In Section 4.5, we extend the multi-layered approach to heterogeneous layers. We start with describing the discretization-free layer and how to perform the dynamic programming for this layer on its own. Next, we discuss the multi-layered approach with heterogeneous layers and describe suitable DP operators.

4.4 Homogeneous layers with variable precision

In this section, we define a multi-layered simulation relation which allows variable precision using one abstract model, but different deviation bounds ϵ and δ. This is graphically represented in Figure 4.5. Furthermore, we detail an appropriate DP approach, discuss multiple computational improvements, prove the control refinement, and give an efficient implementation for linear systems.

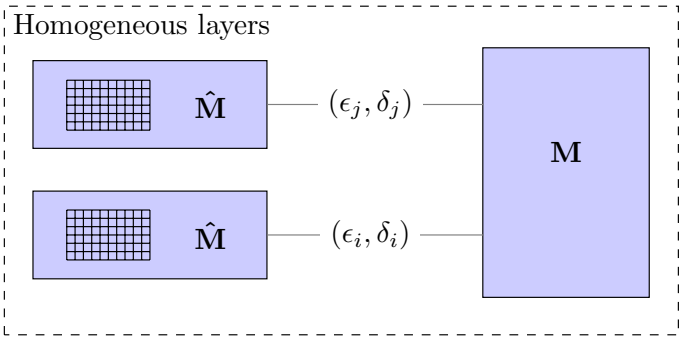


Figure 4.5: Graphical representation of the multi-layered approach with homogeneous layers.

4.4a Multi-layered simulation relation

Current methods define one simulation relation for the whole state space, while we desire a multi-layered simulation relation $\boldsymbol{\mathcal{R}}$ that switches between multiple simulation relations to allow for multiple (ϵ_i, δ_i) pairs, with $i \in \{1, 2, \ldots, N_R\}$ where N_R denotes the number of simulation relations. Each simulation relation is denoted as $\mathcal{R}_i$ and the corresponding precision as ϵ_i.

An example of a multi-layered simulation relation with two simulation relations is given in Figure 4.6.

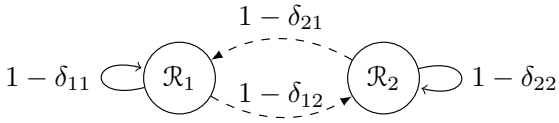


Figure 4.6: Multi-layered simulation relation $\boldsymbol{\mathcal{R}}$ consisting of two simulation relations $\mathcal{R}_1$ and $\mathcal{R}_2$. The edges are labeled with a lower bound on the probability that the transition occurs.

Here, the self-loops represent remaining in the same simulation relation, and a switch is indicated by the dashed arrows. Similar to the invariance requirement in (4.5), we associate a lower bound on the probability of each transition from $\mathcal{R}_i$ to $\mathcal{R}_j$

as $1-\delta_{ij}$, with $i,j \in \{1,2,\ldots,N_R\}$. Furthermore, we define $\boldsymbol{\epsilon}$, and $\boldsymbol{\delta}$ with vector $\boldsymbol{\epsilon} = \begin{bmatrix}\epsilon_1, \epsilon_2, \ldots, \epsilon_{N_R}\end{bmatrix}$, and matrix $\boldsymbol{\delta} = \begin{bmatrix}\delta_{ij}\end{bmatrix}_{ij}$ In the remainder of this chapter, a switch from simulation relation $\mathcal{R}_i$ to $\mathcal{R}_j$ is denoted by the action s_{ij}, and we define the set of all possible switching actions by $\mathbb{S} := \{s_{ij} \text{ with } i,j \in \{1,2,\ldots,N_R\}\}$.

The similarity quantification between the two models is performed by coupling the transitions of the models. First, the control inputs u and $\hat{u}$ are coupled through an interface function denoted by

$$\mathcal{U}_v : \hat{\mathbb{U}} \times \hat{\mathbb{X}} \times \mathbb{X} \to \mathbb{U}. \tag{4.11}$$

Next, the disturbances w and $\hat{w}$ are coupled through a stochastic kernel $\bar{\mathcal{W}}_{ij}(\cdot|\hat{x},x,\hat{u})$ in a similar fashion as in Chapter 2 of this thesis. However, in this multi-layered setting, we compute multiple stochastic kernels $\bar{\mathcal{W}}_{ij}(\cdot|\hat{x},x,\hat{u})$ with $i,j \in \{1,2,\ldots,N_R\}$, and use the switching action s_{ij} to select one of them based on the state pair $(\hat{x},q)$ and the current layer i.

Based on these notions, we can now formally define the multi-layered simulation relation as follows.

Definition 4.3 (Multi-layered simulation relation)**.** *Let the models $\mathbf{M}, \hat{\mathbf{M}} \in \mathcal{M}_{\mathbb{Y}}$, and the interface function $\mathcal{U}_v$ (4.11) be given. If there exists measurable relations $\mathcal{R}_i \subseteq \hat{\mathbb{X}} \times \mathbb{X}$ and Borel measurable stochastic kernels $\bar{\mathcal{W}}_{ij}$ that couple $\mathbb{P}_w$ and $\mathbb{P}_{\hat{w}}$ for $i,j \in \{1,2,\ldots,N_R\}$ such that*

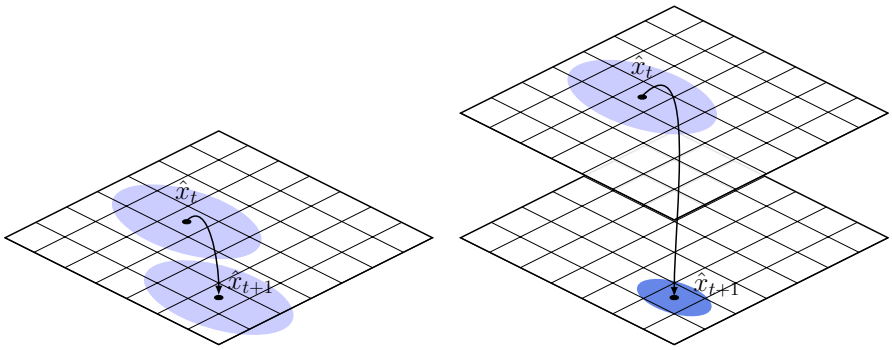
1. $\exists i \in \{1,2,\ldots,N_R\} : (\hat{x}_0, x_0) \in \mathcal{R}_i$,
2. $\forall i \in \{1,2,\ldots,N_R\}, \forall (\hat{x},x) \in \mathcal{R}_i : \boldsymbol{d}_{\mathbb{Y}}(\hat{y},y) \leq \epsilon_i$,
3. $\forall i \in \{1,2,\ldots,N_R\}, \forall (\hat{x},x) \in \mathcal{R}_i, \forall \hat{u} \in \hat{\mathbb{U}} : (\hat{x}^+, x^+) \in \mathcal{R}_j$ *holds with probability at least $1-\delta_{ij}$ with respect to $\bar{\mathcal{W}}_{ij}$,*

then $\hat{\mathbf{M}}$ is stochastically simulated by $\mathbf{M}$ in a multi-layered fashion, denoted as $\hat{\mathbf{M}} \preceq^{\boldsymbol{\delta}}_{\boldsymbol{\epsilon}} \mathbf{M}$, with precision $\boldsymbol{\epsilon} = [\epsilon_i]_i$ and $\boldsymbol{\delta} = [\delta_{ij}]_{ij}$ for $i,j \in \{1,2,\ldots,N_R\}$. □

Remark 4.1. This simulation relation differs from the original one in Definition 2.4, since it contains multiple simulation relations $\mathcal{R}_i$ with different output precision and, therefore, allows for variable precision. Note that for $N_R = 1$, this multi-layered simulation relation becomes equivalent to the simulation relation from Definition 2.4. The difference between a simulation relation with a fixed precision and a multi-layered simulation relation with a variable precision is illustrated in Figure 4.7. Here, we assume ellipsoidal simulation relations, however, the method described in this chapter is not restricted to relations with this specific shape.

4.4b Multi-layered dynamic programming

Consider a switching strategy that associates a switching action s_{ij} to state pairs $(\hat{x},q) \in \hat{\mathbb{X}} \times Q$ depending on the layer i. More specifically, we extend the input



(a) Fixed precision **(b)** Variable precision

Figure 4.7: Graphical representation of a probabilistic transition for a fixed precision in **(a)** and a variable precision with two discretization-based layers in **(b)**. In the left figure, the light blue ellipsoids contain all states x_t resp. x_{t+1} for which it holds that $(\hat{x}_t, x_t) \in \mathcal{R}_1$ resp. $(\hat{x}_{t+1}, x_{t+1}) \in \mathcal{R}_1$. In the right figure, the light blue ellipsoid contains all states x_t for which it holds that $(\hat{x}_t, x_t) \in \mathcal{R}_1$, and the dark blue ellipsoid contains all states x_{t+1} for which it holds that $(\hat{x}_{t+1}, x_{t+1}) \in \mathcal{R}_2$. In both figures, the space discretization or grid size is the same, while the simulation relations $\mathcal{R}_1$ and $\mathcal{R}_2$ are ellipsoids with different parameters.

action $\hat{u}$ to $(\hat{u}, s_{ij})$ and split up the corresponding policy as $\boldsymbol{\mu} = (\boldsymbol{\mu}^u, \boldsymbol{\mu}^s)$, with $\boldsymbol{\mu}^u = (\mu_0^u, \mu_1^u, \dots, \mu_N^u)$ and $\boldsymbol{\mu}^s = (\mu_0^s, \mu_1^s, \dots, \mu_N^s)$ determining respectively the input action and switching action, with respectively the mappings $\mu_k^u : \hat{\mathbb{X}} \times Q \times \{1, 2, \dots, N_R\} \to \hat{\mathbb{U}}$, and $\mu_k^s : \hat{\mathbb{X}} \times Q \times \{1, 2, \dots, N_R\} \to \mathbb{S}$, $\forall k \in [0, \dots, N]$. Note that for $\boldsymbol{\mu}$, we still have $\boldsymbol{\mu} = (\mu_0, \mu_1, \dots, \mu_N)$, with N the time horizon. For the DP, we are now interested in computing both $\hat{u} = \mu_k^u(\hat{x}, q, i)$ and $s_{ij} = \mu_k^s(\hat{x}, q, i)$ for all $k \in [0, \dots, N]$.

To implement a layered DP approach, we extend the value function to include the different relations $\mathcal{R}_i$ as different layers. Hence, to each layer, we assign a value function $V(\hat{x}, q, i)$. This value function defines a lower bound on the probability that specification ϕ will be satisfied in the time horizon $[1, \dots, N]$. We define a new robust operator $\mathbf{T}_{s_{ij}}^{\hat{u}}$ as

$$\mathbf{T}_{s_{ij}}^{\hat{u}}(V)(\hat{x}, q, i) = \mathbf{L}\big(\mathbb{E}_{\hat{u}}\big(\min_{q^+ \in Q_{\epsilon_j}^+} \max\left\{\mathbf{1}_{Q_f}(q^+), V(\hat{x}^+, q^+, j)\right\}\big) - \delta_{ij}\big), \tag{4.12}$$

with

$$Q_{\epsilon_j}^+(q, \hat{y}^+) := \left\{\tau_{\mathcal{A}}(q, L(y^+)) \mid ||y^+ - \hat{y}^+|| \leq \epsilon_j\right\}. \tag{4.13}$$

For given policy $\mu = (\mu^u, \mu^s)$, we define $\mathbf{T}^{\mu}(V)(\hat{x}, q, i) = \mathbf{T}_{s_{ij}}^{\hat{u}}(V)(\hat{x}, q, i)$ with $\hat{u} = \mu^u(\hat{x}, q, i)$, and $s_{ij} = \mu^s(\hat{x}, q, i)$. Consider a policy $\boldsymbol{\mu}_k = (\mu_{k+1}, \dots, \mu_N)$, then for all $(\hat{x}, q, i)$ we have that $V_{N-k+1}^{\boldsymbol{\mu}_{k-1}} = \mathbf{T}^{\mu_k}(V_{N-k}^{\boldsymbol{\mu}_k})$, initialized with $V_0 \equiv 0$. Figure 4.8 gives a graphical representation of the multi-layered value iteration for two layers and all possible switching actions. As before, for a stationary policy $\boldsymbol{\mu}$, the infinite-

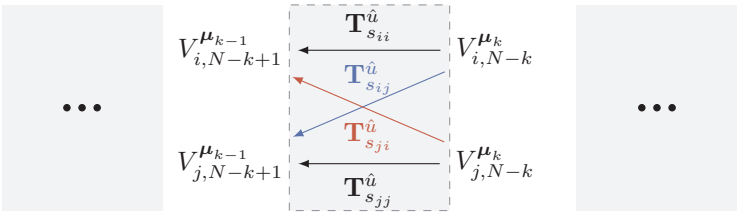


Figure 4.8: Graphical representation of the value iteration for homogeneous layers, where all possible switching actions $s_{ij} \in \mathbb{S}$ are considered. All operations in the gray box can be performed using (4.12). The arguments $(\hat{x}, q, i)$ and $(\hat{x}, q, j)$ of the value functions in respectively the top and bottom rows are omitted for simplicity. Note that the layer is indicated as a subscript instead. The dots indicate that the full value iteration follows the sequence as in Figure 4.3.

horizon value function for all layers can be computed as $V^{\boldsymbol{\mu}}_{\infty} = \lim_{N\to\infty}(\mathbf{T}^{\boldsymbol{\mu}})^N V_0$ with $V_0 \equiv 0$.

Remark 4.2. The DP operator in (4.12) is an adaptation of the operator in (4.8) to a multi-layered setting. For a fixed precision, that is one value for ϵ and one for δ (which could be a function of $\hat{x}$ as in Section 2.6), the DP operators in (4.12) and (4.8) become equivalent, hence we retrieve the DP operator for a fixed precision.

Since the DP operator in (4.12) is a simple adaptation of the standard DP operator in (4.8), we can follow Proposition 4.1 to conclude that it satisfies the important properties mentioned in this proposition.

The value function gives the probability of satisfying the specification in 1 to ∞ time step, by including the first time instance based on x_0, we can compute the robust satisfaction probability, that is

$$\mathbb{R}^{\boldsymbol{\mu}} := \max(\mathbf{1}_{Q_f}(\bar{q}_0), \max_{i\in\{1,2,\dots,N_R\}} V^{\boldsymbol{\mu}}_{\infty}(\hat{x}_0, \bar{q}_0, i)), \tag{4.14}$$

with $\bar{q}_0 = \tau_{\mathcal{A}_\phi}(q_0, L(h(x_0)))$ and $\hat{x}_0 \in \mathcal{R}^{-1}(x_0)$. The robust satisfaction probability as in (4.14) is also denoted as $\mathbb{R}^{\boldsymbol{\mu}}_{\boldsymbol{\epsilon},\boldsymbol{\delta}}(\hat{\mathbf{M}} \times \hat{\mathbf{C}} \models \phi)$.

To compute the optimal converged value function V^*_{∞}, we use the optimal robust operator

$$\mathbf{T}^*(\cdot) := \sup_{\mu} \mathbf{T}^{\mu}(\cdot). \tag{4.15}$$

Note that the operator $\sup_{\mu}(\cdot)$ implicitly optimizes both over $\hat{u} \in \hat{\mathbb{U}}$ and $j \in \{1, 2, \dots, N_R\}$ which in practice is a very expensive operation with respect to computation time and memory usage. This is especially the case when the number of abstract inputs or the number of layers is large. Next, we present a way to alleviate this issue.

4.4c Efficient implementation of multi-layered dynamic programming

Optimize the switching strategy using a surrogate model. To reduce the computation time and memory usage, we split the computation of the optimal input policy and switching strategy. To this end, we use the DP operator in (4.12) in two different ways. First to compute a switching strategy, then to compute the abstract input and the robust satisfaction probability. In this section, we detail an approach to determine the switching strategy that is close to optimal, while also being efficient with respect to computation time and memory usage.

In (4.15), we determine the optimal action pair $(\hat{u}, s_{ij})$, by considering all possible input actions $\hat{u} \in \hat{\mathbb{U}}$ and all possible switching actions $s_{ij} \in \mathbb{S}$ at all states and layers. This computation requires a lot of memory and computation power, therefore we pre-compute an approximately optimal switching strategy.To save computational power, we only compute the switching strategy for an abstract model with an extra coarsely partitioned state space, referred to as the *surrogate model* $\hat{\mathbf{M}}_s$ with state space $\hat{\mathbb{X}}_s$. Note that this surrogate model is a finite-state abstraction of the original model $\mathbf{M}$ that is constructed in the exact same way as finite-state abstraction $\hat{\mathbf{M}}$, however, the grid is a lot coarser for $\hat{\mathbf{M}}_s$ compared to $\hat{\mathbf{M}}$.

The approach to compute the switching strategy using this surrogate model is summarized in Algorithm 4.1. After constructing the surrogate model by partitioning the state space with a finite number of grid cells, we compute a suitable switching action using the $(\boldsymbol{\epsilon}, \boldsymbol{\delta})$ values corresponding to the actual abstract model $\hat{\mathbf{M}}$. More specifically, we compute the switching action at $(\hat{x}, q, i)$ as

$$(\hat{u}^*, s_{ij}^*) = \arg\max_{(\hat{u}, s_{ij})} \mathbf{T}_{s_{ij}}^{\hat{u}}(V)(\hat{x}, q, i), \tag{4.16}$$

with operator $\mathbf{T}_{s_{ij}}^{\hat{u}}(\cdot)$ as in (4.12), and with $j \in \{1, 2, \ldots, N_R\}$. This optimal switching action $s_{ij}^* = \mu_k^s(\hat{x}, q, i)$ is computed at each time step $k \in [0, \ldots, N]$, with N the time horizon. Hence, it is used to build up the optimal switching strategy $\boldsymbol{\mu}^s = (\mu_0^s, \mu_1^s, \ldots, \mu_N^s)$.

Algorithm 4.1 Optimize switching strategy

1: **Input:** $\mathbf{M}$, $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}$, $\mathcal{A}_\phi$
2: $\hat{\mathbf{M}}_s \leftarrow$ Construct surrogate model $\hat{\mathbf{M}}_s$ using a coarse grid.
3: $\boldsymbol{\mu}^s(\hat{x}_s, q, i) \leftarrow$ Perform dynamic programming through (4.12) over all possible switching actions in $\mathbb{S}$ based on model $\hat{\mathbf{M}}_s$ from step 2 and inputs $(\boldsymbol{\epsilon}, \boldsymbol{\delta})$. After convergence is reached, compute optimal switching action using (4.16).
4: $\boldsymbol{\mu}^s(\hat{x}, q, i)$ $\leftarrow$Translate switching strategy for $\hat{\mathbf{M}}_s$ from step 3 to switching strategy for $\hat{\mathbf{M}}$.
5: **Output:** $\boldsymbol{\mu}^s(\hat{x}, q, i)$

Since the DP algorithm determines the best switching action s_{ij} for each abstract state of the surrogate model, it is optimal with respect to this surrogate model. Finally, in step 4, we translate the switching strategy from the surrogate model to the actual abstract model $\hat{\mathbf{M}}$. More specifically, for each $\hat{x} \in \hat{\mathbb{X}}$ we determine the closest (with respect to the L_2-norm) state $\hat{x}_s \in \hat{\mathbb{X}}_s$ and associate the same switching action s_{ij} to it.

After computing the switching strategy $\boldsymbol{\mu}^s$, the control policy $\boldsymbol{\mu}^u$ is determined for the actual abstract model $\hat{\mathbf{M}}$ together with the robust satisfaction probability. The complete approach is summarized in Algorithm 4.2.

Algorithm 4.2 Control synthesis

1: **Input:** $\mathbf{M}$, $\hat{\mathbf{M}}$, $\mathcal{A}_\phi$
2: $(\boldsymbol{\epsilon}, \boldsymbol{\delta}) \leftarrow$ Compute $(\boldsymbol{\epsilon}, \boldsymbol{\delta})$ such that $\hat{\mathbf{M}} \preceq^{\boldsymbol{\delta}}_{\boldsymbol{\epsilon}} \mathbf{M}$ holds with relation $\mathcal{R}$.
3: $\boldsymbol{\mu}^s(\hat{x}, q, i) \leftarrow$ Algorithm 4.1.
4: $\boldsymbol{\mu}^u(\hat{x}, q, i), \mathbb{R}^{\boldsymbol{\mu}}_{\boldsymbol{\epsilon},\boldsymbol{\delta}}(\hat{\mathbf{M}} \times \hat{\mathbf{C}} \models \phi) \leftarrow$ Compute robust controller and satisfaction probability given the switching strategy from step 3.
5: **Output:** $\boldsymbol{\mu} = (\boldsymbol{\mu}^u, \boldsymbol{\mu}^s)$, and $\mathbb{R}^{\boldsymbol{\mu}}_{\boldsymbol{\epsilon},\boldsymbol{\delta}}(\hat{\mathbf{M}} \times \hat{\mathbf{C}} \models \phi)$

Partial value iteration. To improve the computational efficiency of our approach even further, we consider layers where we only partially compute the value function. An example is illustrated in Figure 4.9, where for the gray part of the state space in the high-precision layer, we fix the value function to zero and do not update it.Therefore, the total computation time and memory usage is lower than when performing the value iteration fully. This is especially the case for high-dimensional systems, models with a large number of layers, and models with a large state space.

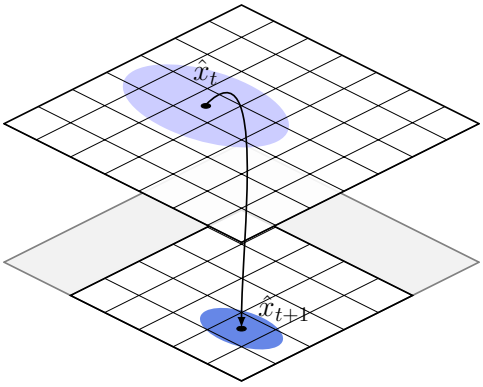


Figure 4.9: Homogeneous layers with a variable precision, where the value function in the high-precision layer is not fully computed. The part of the state space of the high-precision layer (bottom), where the value function is not computed, is given in gray.

4.4d Control refinement

The abstract controller $\hat{\mathbf{C}}$ is designed based on the abstract model $\hat{\mathbf{M}}$ in (4.4). Since we take the similarity quantification into account during the control design, we can refine the abstract controller $\hat{\mathbf{C}}$ to a controller $\mathbf{C}$ that can be deployed on the original model $\mathbf{M}$. In this section, we show that with the given DP approach, such a control refinement is indeed possible and the computed satisfaction probability is valid.

We design the abstract control $\hat{\mathbf{C}}$ based on the abstract model $\hat{\mathbf{M}}$ that is coupled through an interface function $\mathcal{U}_v$ as in (4.11) and a stochastic kernel $\bar{\mathcal{W}}_{ij}$ as in (4.17). Due to this coupling, we can define the combined stochastic difference equation, with finite mode σ_t that keeps track of the layer as

$$\hat{\mathbf{M}}\|\mathbf{M}: \begin{cases} \begin{pmatrix} \hat{x}_{t+1} \\ x_{t+1} \end{pmatrix} & = \begin{pmatrix} \hat{f}(\hat{x}_t, \hat{u}_t, \hat{w}_t) \\ f(x_t, \mathcal{U}_v(\hat{u}_t, \hat{x}_t, x_t), w_t) \end{pmatrix} \\ \hat{y}_t & = \hat{h}(\hat{x}_t) \\ y_t & = h(x_t) \\ \sigma_{t+1} & = j \text{ if } \sigma_t = i, \text{ and } s_t = s_{ij} \\ (\hat{w}_t, w_t) & \sim \bar{\mathcal{W}}_{ij}(\cdot|\hat{x}, x, \hat{u}) \text{ if } s_t = s_{ij}, \end{cases} \tag{4.17}$$

with pair of states $(\hat{x}, x) \in \hat{\mathbb{X}} \times \mathbb{X}$, control input $\hat{u} \in \hat{\mathbb{U}}$, outputs $\hat{y}, y \in \mathbb{Y}$, and coupled disturbance $(\hat{w}, w)$. The input space of this combined system has been extended; that is, next to control input $\hat{u}_t$ we also have a switching input s_t. More specifically, since the disturbances of the combined transitions (4.17) are generated from the stochastic kernel $\bar{\mathcal{W}}_{ij}$, (4.17) holds, with $(\hat{w}, w) \sim \bar{\mathcal{W}}_{ij}$ if $s_t = s_{ij}$. The combined system is illustrated in the gray box in Figure 4.10a. Given the coupled stochastic difference equation, we can analyze how close the transitions are and therefore, quantify the similarity between models (4.1) and (4.4). Furthermore, the additional switching action allows us to do this in a multi-layered manner, as defined in Definition 4.3.

Consider a control strategy $\boldsymbol{\mu}$ for $\hat{\mathbf{M}}$, indicated by $\hat{\mathbf{C}}$ in Figure 4.10a. This strategy can also be implemented on the combined model $\hat{\mathbf{M}}\|\mathbf{M}$ and we denote the value function of the combined model as $V_c(\hat{x}, x, q, i)$. The control strategy for the combined model can be refined to a control strategy of the original model $\mathbf{M}$ (4.1), as depicted in Figure 4.10b. Although $V_c(\hat{x}, x, q, i)$ expresses the probability of satisfaction, it cannot be computed directly, instead we can compute $V(\hat{x}, q, i)$ over the abstract model $\hat{\mathbf{M}}$ using (4.12).

To prove that for any control strategy $\boldsymbol{\mu}$ (or controller $\hat{\mathbf{C}}$) for $\hat{\mathbf{M}}\|\mathbf{M}$ there still trivially exists a controller $\mathbf{C}$ for $\mathbf{M}$ that preserves the lower-bound on the satisfaction probability, it is sufficient to formulate the following lemma.

Lemma 4.1. *Suppose* $\hat{\mathbf{M}} \preceq^{\boldsymbol{\delta}}_{\boldsymbol{\epsilon}} \mathbf{M}$ *with a multi-layered simulation relation* $\mathcal{R}$ *is given. Let* $V(\hat{x}, q, j) \leq V_c(\hat{x}, x, q, j)$ *for all* $(\hat{x}, x) \in \mathcal{R}_j$*, then*

$$\mathbf{T}^{\hat{u}}_{s_{ij}}(V)(\hat{x}, q, i) \leq \mathbf{T}^{\hat{u}}_{s_{ij}}(V_c)(\hat{x}, x, q, i) \quad \forall (\hat{x}, x) \in \mathcal{R}_i, \tag{4.18}$$

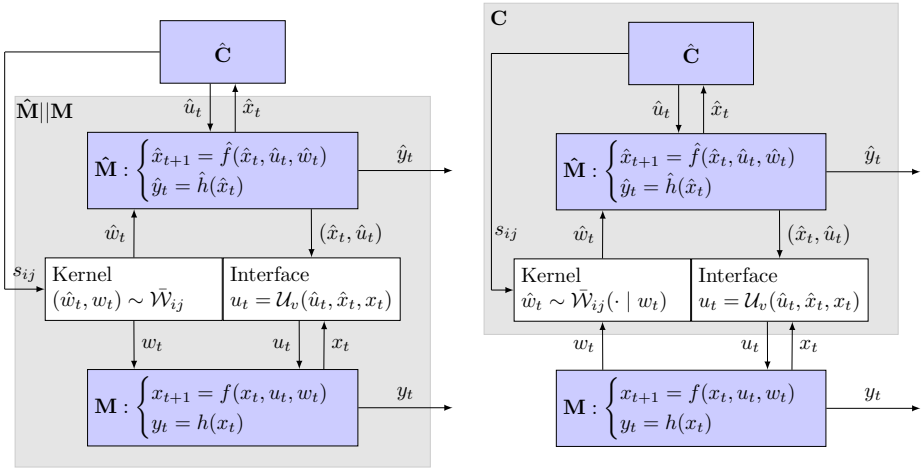


Figure 4.10: In **(a)**, designing an abstract controller $\hat{\mathbf{C}}$ for coupled transitions of $\hat{\mathbf{M}}||\mathbf{M}$, where the switching action s_{ij} determines the stochastic kernel $\bar{\mathcal{W}}_{ij}$. In **(b)**, refined controller $\mathbf{C}$ deployed on the model $\mathbf{M}$ with *conditional* kernel $\bar{\mathcal{W}}_{ij}$.

where $\mathbf{T}^{\hat{u}}_{s_{ij}}(V)(\hat{x}, q, i)$ *is the* $(\boldsymbol{\epsilon}, \boldsymbol{\delta})$*-robust operator* (4.12) *with respect to stochastic transitions of* $\hat{\mathbf{M}}$ *and* $\mathbf{T}^{\hat{u}}_{s_{ij}}(V_c)(\hat{x}, x, q, i)$ *is the exact recursion* (4.7) *extended to the combined stochastic transitions* (4.17)*.*

The proof of Lemma 4.1 follows along the same lines as the proof of Lemma 3 in the extended version of Haesaert and Soudjani (2020). It is given here for completeness.

Proof. The expected-value part of the operator $\mathbf{T}^{\hat{u}}_{s_{ij}}(V)(\hat{x}, q, i)$ in (4.12) equals

$$\mathbb{E}_{\hat{u}}\Big(\min_{q^+ \in Q^+_{\epsilon_j}} \max\left\{ \mathbf{1}_{Q_f}(q^+), V(\hat{x}^+, q^+, j) \right\} \Big) - \delta_{ij} \tag{4.19}$$

This expected value can be rewritten to an integral and multiplication with the transition kernel. Hence, (4.19) is equivalent to

$$\int_{\hat{x}^+ \in \hat{\mathbb{X}}} \min_{q^+ \in Q_{\epsilon_j}} \max\left\{ \mathbf{1}_{Q_f}(q^+), V(\hat{x}^+, q^+, j) \right\} \mathbf{t}(\mathrm{d}\hat{x}^+ \mid \hat{x}, \hat{u}) - \delta_{ij}. \tag{4.20}$$

Denote the transition kernel of the combined model $\hat{\mathbf{M}}||\mathbf{M}$ as $\bar{\mathcal{W}}^x_{ij}(\cdot \mid \hat{x}, x, \hat{u}) : \mathcal{B}(\hat{\mathbb{X}} \times \mathbb{X}) \to [0, 1]$, which is equivalent to the evolution of the state pair of $\hat{\mathbf{M}}||\mathbf{M}$ as in (4.17), that is

$$\begin{pmatrix} \hat{x}_{t+1} \\ x_{t+1} \end{pmatrix} = \begin{pmatrix} \hat{f}(\hat{x}_t, \hat{u}_t, \hat{w}_t) \\ f(x_t, \mathcal{U}_v(\hat{u}_t, \hat{x}_t, x_t), w_t), \end{pmatrix}$$

with $(\hat{w}_t, w_t) \sim \bar{\mathcal{W}}_{ij}(\cdot \mid \hat{x}, x, \hat{u}) : \mathcal{B}(\mathbb{W} \times \mathbb{W}) \to [0,1]$, written as a transition kernel. For $(\hat{x}, x) \in \mathcal{R}_i$ and input $\hat{u}$ applied to the combined model $\hat{\mathbf{M}}||\mathbf{M}$, the integral in (4.20) is equivalent to the integral

$$\int_{(\hat{x}^+, x^+) \in \hat{\mathbb{X}} \times \mathbb{X}} \min_{q^+ \in Q_{\epsilon_j}} \max\left\{\mathbf{1}_{Q_f}(q^+), V(\hat{x}^+, q^+, j)\right\} \bar{\mathcal{W}}_{ij}^{x}(\mathrm{d}\hat{x}^+ \times \mathrm{d}x^+ \mid \hat{x}, x, \hat{u}) - \delta_{ij}.$$

For $(\hat{x}, x) \in \mathcal{R}_i$, we can split this integral up using the simulation relation $\mathcal{R}_i \subseteq \hat{\mathbb{X}} \times \mathbb{X}$ and find the following upper bound

$$\begin{aligned}
&\int_{(\hat{x}^+, x^+) \in \mathcal{R}_i} \min_{q^+ \in Q_{\epsilon_j}} \max\left\{\mathbf{1}_{Q_f}(q^+), V(\hat{x}^+, q^+, j)\right\} \bar{\mathcal{W}}_{ij}^{x}(\mathrm{d}\hat{x}^+ \times \mathrm{d}x^+ \mid \hat{x}, x, \hat{u}) \\
&+ \int_{(\hat{x}^+, x^+) \in (\hat{\mathbb{X}} \times \mathbb{X}) \setminus \mathcal{R}_i} \min_{q^+ \in Q_{\epsilon_j}} \max\left\{\mathbf{1}_{Q_f}(q^+), V(\hat{x}^+, q^+, j)\right\} \bar{\mathcal{W}}_{ij}^{x}(\mathrm{d}\hat{x}^+ \times \mathrm{d}x^+ \mid \hat{x}, x, \hat{u}) - \delta_{ij} \\
&\leq \int_{(\hat{x}^+, x^+) \in \mathcal{R}_i} \min_{q^+ \in Q_{\epsilon_j}} \max\left\{\mathbf{1}_{Q_f}(q^+), V(\hat{x}^+, q^+, j)\right\} \bar{\mathcal{W}}_{ij}^{x}(\mathrm{d}\hat{x}^+ \times \mathrm{d}x^+ \mid \hat{x}, x, \hat{u}).
\end{aligned}$$

Here, we used that $\bar{\mathcal{W}}_{ij}^{x}((\hat{\mathbb{X}} \times \mathbb{X}) \setminus \mathcal{R}_i \mid \hat{x}, x, \hat{u}) \leq \delta_{ij}$.

Following the assumption of the lemma, we have that for all $(\hat{x}^+, x^+) \in \mathcal{R}_j$, it holds that $V(\hat{x}^+, q^+, j) \leq V_c(\hat{x}^+, x^+, q^+, j)$. Furthermore, for all $(\hat{x}^+, x^+) \in \mathcal{R}_j$, we have $q^+ = \tau_{\mathcal{A}}(q, L(y^+)) \in Q_{\epsilon_j}^+$. Hence, we can rewrite the last integral over $\mathcal{R}_i$ and find the upper bound

$$\begin{aligned}
&\int_{(\hat{x}^+, x^+) \in \mathcal{R}_i} \max\left\{\mathbf{1}_{Q_f}(q^+), V_c(\hat{x}^+, x^+, q^+, j)\right\} \bar{\mathcal{W}}_{ij}^{x}(\mathrm{d}\hat{x}^+ \times \mathrm{d}x^+ \mid \hat{x}, x, \hat{u}) \\
&\leq \int_{(\hat{x}^+, x^+) \in \hat{\mathbb{X}} \times \mathbb{X}} \max\left\{\mathbf{1}_{Q_f}(q^+), V_c(\hat{x}^+, x^+, q^+, j)\right\} \bar{\mathcal{W}}_{ij}^{x}(\mathrm{d}\hat{x}^+ \times \mathrm{d}x^+ \mid \hat{x}, x, \hat{u})
\end{aligned}$$

where the last integral is equal to $\mathbf{T}_{s_{ij}}^{\hat{u}}(V_c)(\hat{x}, x, q, i)$.

By taking the truncation $\mathbf{L}$ to the $[0,1]$ interval over (4.19), we obtain operator $\mathbf{T}_{s_{ij}}^{\hat{u}}(V)(\hat{x}, q, i)$ as in (4.12). This truncation operation does not alter the steps in the proof, and since $\mathbf{T}_{s_{ij}}^{\hat{u}}(V_c)(\hat{x}, x, q, i)$ naturally falls within the interval $[0,1]$, Lemma 4.1 is proven. □

As the combined model represents the extension of $\hat{\mathbf{C}}$ to $\mathbf{C}$, as depicted in Figure 4.10b, we can guarantee that the robust satisfaction probability computed using (4.14), gives a lower bound on the actual satisfaction probability of $\mathbf{M} \times \mathbf{C}$. To recap, we quantify their similarity between models $\mathbf{M}$ in (4.1) and $\hat{\mathbf{M}}$ in (4.4) through a multi-layered simulation relation as in Definition 4.3. We use the abstract model $\hat{\mathbf{M}}$ to compute 1) a policy $\boldsymbol{\mu}$ or equivalently an abstract controller $\hat{\mathbf{C}} : (\hat{u}, s_{ij}) = \boldsymbol{\mu}(\hat{x}, q, i)$, and 2) the robust satisfaction $\mathbb{R}_{\boldsymbol{\epsilon},\boldsymbol{\delta}}(\hat{\mathbf{M}} \times \hat{\mathbf{C}} \models \phi)$ as in (4.14). In this computation, we take the deviation bounds $(\boldsymbol{\epsilon}, \boldsymbol{\delta})$, such that we have

$\hat{\mathbf{M}} \preceq_{\boldsymbol{\epsilon}}^{\boldsymbol{\delta}} \mathbf{M}$, into account. Based on Lemma 4.1, we can now follow Theorem 4 in Haesaert and Soudjani (2020) to conclude that this robust satisfaction probability gives a lower bound on the actual satisfaction probability.

Proposition 4.3. *Given models* $\mathbf{M}$ *in* (4.1), $\hat{\mathbf{M}}$ *in* (4.4), *DFA* $\mathcal{A}_\phi$, *and stationary Markov policy* $\boldsymbol{\mu}$. *If* $\hat{\mathbf{M}} \preceq_{\boldsymbol{\epsilon}}^{\boldsymbol{\delta}} \mathbf{M}$ *holds, then there exists* $\mathbf{C}$, *such that* $\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi) \geq \mathbb{R}_{\boldsymbol{\epsilon},\boldsymbol{\delta}}^{\boldsymbol{\mu}}(\hat{\mathbf{M}} \times \hat{\mathbf{C}} \models \phi)$.

4.4e Implementation of similarity quantification for LTI systems

In this section, we elaborate on how to compute the deviation bounds $(\boldsymbol{\epsilon}, \boldsymbol{\delta})$ such that $\hat{\mathbf{M}} \preceq_{\boldsymbol{\epsilon}}^{\boldsymbol{\delta}} \mathbf{M}$ holds (step 2 of Algorithm 4.2). The derivation is given for linear time-invariant (LTI) systems but can be extended to nonlinear stochastic systems following the techniques described in Section 2.5 of this thesis. The approach detailed next, follows the same reasoning as in Section 2.4a of this thesis, but we made adaptations such that it suits the multi-layered setting.

Let the models $\mathbf{M}$ (4.1) and $\hat{\mathbf{M}}$ (4.4) be linear time-invariant (LTI) systems whose behavior is described by the following stochastic difference equations

$$\mathbf{M}: \begin{cases} x(t+1) = Ax(t) + Bu(t) + B_w w(t) \\ y(t) = Cx(t), \text{ and} \end{cases} \tag{4.21}$$

$$\hat{\mathbf{M}}: \begin{cases} \hat{x}(t+1) = \Pi\left(A\hat{x}(t) + B\hat{u}(t) + B_w \hat{w}(t)\right) \\ \hat{y}(t) = C\hat{x}(t), \end{cases} \tag{4.22}$$

with matrices A, B, B_w and C of appropriate sizes and with the disturbances $w(t), \hat{w}(t)$ having the standard Gaussian distribution, i.e., $w(t) \sim \mathcal{N}(0, I) = \mathbb{P}_w$ and $\hat{w}(t) \sim \mathcal{N}(0, I) = \mathbb{P}_{\hat{w}}$. The abstract model is constructed by partitioning the state space $\mathbb{X}$ in a finite number of regions $\mathbb{A}_j \subset \mathbb{X}$ and operator $\Pi(\cdot) : \mathbb{X} \to \hat{\mathbb{X}}$ maps states $x \in \mathbb{A}_j$ to their representative points[1] $\hat{X}_j \in \hat{\mathbb{X}}$. We assume that the regions $\mathbb{A}_j$ are designed in such a way that the set $\mathcal{B} := \bigcup_j \{\hat{X}_j - x_j | x_j \in \mathbb{A}_j\}$ is a bounded polytope and has vertices vert($\mathcal{B}$). Details on constructing such an abstract LTI system can be found in Section 2.4a of this thesis.

To compute the multi-layered simulation relations in Definition 4.3, we choose the interface function $u(t) = \mathcal{U}_v(\hat{u}_t, \hat{x}_t, x_t) = \hat{u}(t)$ and consider simulation relations $\mathcal{R}_i$

$$\mathcal{R}_i := \left\{ (\hat{x}, x) \in \hat{\mathbb{X}} \times \mathbb{X} \mid ||x - \hat{x}||_D \leq \epsilon_i \right\}, \tag{4.23}$$

where $||x||_D = \sqrt{x^T D x}$ with D a symmetric positive definite matrix $D = D^T \succ 0$. We use the same weighting matrix D for all simulation relations $\mathcal{R}_i$, with $i \in \{1, 2, \dots N_R\}$. The simulation relations in (4.23) have an ellipsoidal shape and for

[1] In general, any point in the region $\mathbb{A}_j$ can be its representative point, but in practice the center has computational benefits.

fixed precision such ellipsoids are illustrated in Figure 4.7a. A switch from one simulation relation to the other one is similarly illustrated in Figure 4.7b.

For these relations (4.23), condition 1 in Definition 4.3 is satisfied by choosing weighting matrix $D \succ 0$, such that

$$C^T C \preceq D. \tag{4.24}$$

We can now construct kernels $\bar{\mathcal{W}}_{ij}$ using a coupling compensator as introduced in Chapter 2 of this thesis. By doing so, condition 2 of Definition 4.3 can be quantified via contractive sets for the error dynamics $x_{t+1} - \hat{x}_{t+1}$ based on the combined transitions (4.17). We assume that there exist factors $\alpha_{ij} \in [0,1]$ with $\epsilon_j = \alpha_{ij}\epsilon_i$ that represent the set contraction between the different simulation relations. Now, we can describe the satisfaction of condition 2 as a function of δ_{ij}, α_{ij} and ϵ_i.

Lemma 4.2. *Consider models* $\mathbf{M}$ *in* (4.21) *and* $\hat{\mathbf{M}}$ *in* (4.22) *for which simulation relations* $\mathcal{R}_i$ *as in* (4.23) *are given with weighting matrix* D *satisfying* (4.24). *Given* δ_{ij}, α_{ij}, *and* ϵ_i, *if there exist parameters* λ_{ij} *and matrices* F_{ij} *such that the matrix inequalities*

$$\begin{bmatrix} \frac{1}{\epsilon_i^2} D & F_{ij}^T \\ F_{ij} & r_{ij}^2 I \end{bmatrix} \succeq 0, \qquad \textit{(input bound)} \tag{4.25a}$$

$$\begin{bmatrix} \lambda_{ij} D & * & * \\ 0 & (\alpha_{ij}^2 - \lambda_{ij})\epsilon_i^2 & * \\ D(A + B_w F_{ij}) & D\beta_l & D \end{bmatrix} \succeq 0 \quad \textit{(contraction)} \tag{4.25b}$$

are satisfied, then there exists a $\bar{\mathcal{W}}_{ij}$ *such that condition 2 in Definition 4.3 is satisfied. The matrix inequalities in* (4.25) *are parameterized with* $\lambda_{ij} > 0$ *and should hold for all* $\beta_l \in \mathit{vert}(\mathcal{B})$. *Furthermore,* r_{ij} *is computed as a function of* δ_{ij}, *that is* $r_{ij} = |2 \operatorname{idf}\left(\frac{1-\delta_{ij}}{2}\right)|$, *with* idf *denoting the inverse distribution function of the standard Gaussian distribution.* □

This lemma allows us to conclude the following.

Theorem 4.1. *Consider models* $\mathbf{M}$ *in* (4.21) *and* $\hat{\mathbf{M}}$ *in* (4.22) *for which simulation relations* $\mathcal{R}_i$ *as in* (4.23) *are given with weighting matrix* D *satisfying* (4.24). *If the inequalities* (4.25) *hold for all* $i, j \in \{1, \ldots, N_R\}^2$ *and there exists* $i \in \{1, \ldots, N_R\}$ *such that* $(\hat{x}_0, x_0) \in \mathcal{R}_i$ *then* $\hat{\mathbf{M}}$ *is stochastically simulated by* $\mathbf{M}$ *in a multi-layered fashion as in Definition 4.3, denoted as* $\hat{\mathbf{M}} \preceq_{\boldsymbol{\epsilon}}^{\boldsymbol{\delta}} \mathbf{M}$.

Proof. The proof of both Lemma 4.2 and Theorem 4.1 build on top of the proofs of Theorem 2.1 and Theorem 2.2 for controlled invariant sets. Instead of invariant sets, the proof uses contractive sets to handle the multi-layered simulation relation.

For the construction of the matrix inequalities in (4.25), we follow Chapter 2 and model the state dynamics of the abstract model (4.22) as $\hat{x}(t+1) = A\hat{x}(t) + B\hat{u}(t) + B_w(\hat{w}_\gamma(t) - \gamma(t)) + \beta(t)$ with disturbance $\hat{w}_\gamma \in \mathbb{W} \subseteq \mathbb{R}^{n_w}$, shift $\gamma \in \Gamma$ and deviation

$\beta \in \mathcal{B}$. The disturbance is generated by a Gaussian distribution with a shifted mean, $\hat{w}_\gamma \sim \mathcal{N}(\gamma, I)$. The β-term pushes the next state towards the representative point of the grid cell. Based on Chapter 2, we choose stochastic kernels $\bar{\mathcal{W}}_{ij}$ such that the probability of event $w - \hat{w}_\gamma = 0$ is large. The error dynamics conditioned on this event equal $x_\Delta^+ = Ax_\Delta(t) + B_w\gamma_{ij}(t) - \beta(t)$, where state x_Δ and state update x_Δ^+ are the abbreviations of $x_\Delta(k) := x(t) - \hat{x}(t)$ and $x_\Delta(t+1)$, respectively. This can be seen as a system with state x_Δ, constrained input γ_{ij}, and bounded disturbance β.

For a given deviation δ_{ij}, we compute a bound on the allowable shift as $\gamma_{ij} \in \Gamma_{ij} := \{\gamma_{ij} \in \mathbb{R}^{n_w} \mid ||\gamma_{ij}|| \leq r_{ij}\}$ and we parameterize the shift $\gamma_{ij} = F_{ij}x_\Delta$ with the matrix F_{ij}. In the exact same fashion as the proof of Theorem 2.2, we can show that if there exists λ_{ij} and F_{ij} such that the matrix inequalities in (4.25) are satisfied, then the following implications also hold

$$x_\Delta^\top D x_\Delta \leq \epsilon_i^2 \implies x_\Delta^\top F_{ij}^\top F_{ij} x_\Delta \leq r_{ij}^2 \qquad \textit{(input bound)}$$
$$x_\Delta^\top D x_\Delta \leq \epsilon_i^2 \implies (x_\Delta^+)^\top D x_\Delta^+ \leq \alpha_{ij}^2 \epsilon_i^2. \qquad \textit{(contraction)}$$

Therefore, we satisfy the bound $\gamma_{ij} \in \Gamma_{ij}$ and the simulation relation $\mathcal{R}_i$ describes an α_{ij}-contractive set. Hence, using Lemma 2.1 in Chapter 2, we can conclude that there exists a kernel $\bar{\mathcal{W}}_{ij}$, such that condition 2 in Definition 4.3 is satisfied. Since condition 1 in Definition 4.3 is already satisfied by choosing D appropriately, $\hat{\mathbf{M}} \preceq_{\boldsymbol{\epsilon}}^{\boldsymbol{\delta}} \mathbf{M}$ holds as long as the conditions in Theorem 4.1 are satisfied.

Concluding, since (4.24) holds, condition 1 in Definition 4.3 is satisfied for all i, j. If in addition λ_{ij} and F_{ij} satisfy (4.25), then there exists a kernel $\bar{\mathcal{W}}_{ij}$ such that condition 2 in Definition 4.3 holds (Lemma 4.2). Once this does not only hold for a specific i, j, but for all $i, j \in [1, \ldots, N_R]$ and there exists $i \in \{1, 2, \ldots, N_R\}$ with $(\hat{x}_0, x_0) \in \mathcal{R}_i$, then we have $\hat{\mathbf{M}} \preceq_{\boldsymbol{\epsilon}}^{\boldsymbol{\delta}} \mathbf{M}$. □

4.5 Heterogeneous layers

To alleviate the curse of dimensionality induced by discretization-based abstraction, and to improve the computational efficiency of our approach, we consider the possibility of an additional discretization-free layer. More specifically, in this section we change the multi-layered approach from the previous section to have layers with both discretization-based models and discretization-free models, hence we consider *heterogeneous* layers. We start by introducing the discretization-free (DF) layer, its model, and the dynamic programming approach for this layer on its own. Next, we consider the combination of a discretization-free layer and a discretization-based (DB) layer, referred to as *heterogeneous layers*, for which we define conditions on switching between those different layers. Finally, we describe the adjusted dynamic programming method.

4.5a Discretization-free layer

In the discretization-free layer, we consider a generic representation of dynamic programming-based planning that leverages a discretization-free approach. We introduce the important concepts next.

Model. Consider a finite set of states taken from the state space of the original model $\mathbf{M}$ as in (4.1), that is $x_w \in \mathbb{X}_w \subset \mathbb{X}$, referred to as waypoints. We associate to each waypoint an ellipsoidal set

$$\mathcal{E}(x_w) := \{x \in \mathbb{X} \mid ||x - x_w||_{D_w} \leq \epsilon_w\} \tag{4.26}$$

containing states $x \in \mathbb{X}$ of the original model. Here, D_w is a symmetric positive definite matrix. Note that the center of $\mathcal{E}$ equals x_w, but without loss of generality, we consider its shape the same for all $x_w \in \mathbb{X}_w$. We now define a state transition function $\Delta_w : \mathbb{X}_w \times \mathbb{X}_w \to [0, 1]$ that gives a lower bound on the probability of reaching waypoint x'_w in a finite number of steps n_s. More specifically, there exists a sequence of control strategies, such that with a probability of at least $\Delta_w(x_w, x'_w)$, $\exists n_s \in \mathbb{N}$, such that $x_k \in \mathcal{E}(x_w)$ and $x_{k+n_s} \in \mathcal{E}(x'_w)$. Together, this allows us to define the following waypoint model

$$\mathbf{M}_w = (\mathbb{X}_w, x_{w,0}, \Delta_w, \mathbb{Y}, h_w), \tag{4.27}$$

with initial state $x_{w,0} \in \mathbb{X}_w$, outputs $y_w \in \mathbb{Y}$, and output map $h_w : \mathbb{X}_w \to \mathbb{Y}$. An example of a waypoint model is given in Figure 4.11.

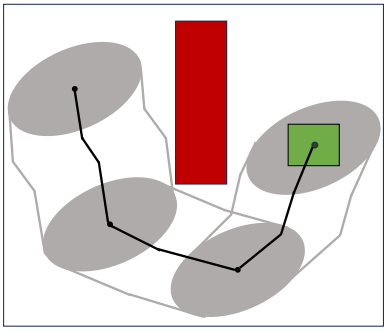


Figure 4.11: Example of a state trajectory (black line) of a waypoint model with the corresponding tube in gray. The red and green regions are respectively an avoid and a goal region projected from the output space to the state space. The black dots are the waypoints x_w and the gray ellipsoids are the sets $\mathcal{E}$. Note that this waypoint model is not well-posed since not all of the outputs corresponding to the top right ellipsoid have the same label.

Specification. To handle temporal logic specifications based on the model $\mathbf{M}_w$, we require that $\mathbf{M}_w$ is well-posed with respect to labeling $L : \mathbb{Y} \to 2^{\mathrm{AP}}$, if $\forall x_w \in \mathbb{X}_w$:

1. all outputs y corresponding to states $x \in \mathcal{E}(x_w)$ have the same label,
2. the outputs corresponding to paths from x_w to x'_w either never change label or only once.

These two assumptions allow us to keep track of the DFA state in a simple manner, which is necessary when considering temporal logic specifications.

The construction of the waypoint model, allows us to handle specifications given using scLTL$\backslash\bigcirc$, where the $\bigcirc$-operator is excluded since the number of time steps between waypoints is usually larger than 1.

Available methods. The construction of paths in the waypoint model is in essence the same as solving a (deterministic) reach-avoid problem, for which there are a lot of methods available (Althoff et al. 2010; Girard 2012; Bogomolov et al. 2019). For some situations, the transition probability Δ_w can even be computed directly via a continuous-state (stochastic) model using for example the tool `SReachTools` (Vinod et al. 2019). By defining the layer in this manner, we can use dynamic programming (as defined next) to make a connection between DB and DF methods.

Dynamic programming. Denote the value function of the DF layer as $V_{x_w} : \mathbb{X}_w \times Q \to [0,1]$. Consider time horizon $[1,\dots,N]$ and a policy $\boldsymbol{\mu}_k = (\mu_{k+1},\dots\mu_N)$ with time horizon $N-k$, and with $\mu_k : \mathbb{X}_w \to \mathbb{X}_w$ that chooses a waypoint x'_w that is reachable from waypoint x_w. We compute the value function at the next iteration as $V^{\text{next}}_{x_w} = V^{\boldsymbol{\mu}_{k-1}}_{x_w,N-k+1} = \mathbf{T}^{\mu_k}_{x_w}(V^{\boldsymbol{\mu}_k}_{x_w,N-k})$, with the Bellman operator defined as follows.

$$\mathbf{T}^{x'_w}_{x_w}(V_{x_w})(x_w,q) = \mathbf{L}\big(\max\big\{\mathbf{1}_{Q_f}(q'), V_{x_w}(x'_w,q')\big\} - (1-\Delta_w(x_w,x'_w))\big), \quad (4.28)$$

with q' the DFA state after n_s time steps, that is $q' = q(t+n_s)$, and with $\Delta_w(x_w,x'_w)$ the probability of reaching x'_w from x_w. Note that for this layer, the Bellman operator does not contain an expected value operator anymore, so the value function is computed in a deterministic fashion.

4.5b Heterogeneous layers

In this section, we are interested in switching between DF layers and DB layers. Denote the switching actions from a DF layer to a DB layer and vice versa as respectively s_{fb} and s_{bf}. For ease of notation, we explain everything for only one DF layer and one DB layer, and we remark on how to deal with multiple DF and DB layers. This is graphically represented in Figure 4.12.

In the previous section on the multi-layered approach, we use the inherent contraction of the state dynamics to switch to a layer with higher precision. For heterogeneous layers, this is not possible, since the states of the original model that are associated with the DF layer might not cover the complete state space. More specifically, for the DB layers, all states $x \in \mathbb{X}$ can be mapped to a representative state $\hat{x} \in \hat{\mathbb{X}}$ through the operator $\Pi : \mathbb{X} \to \hat{\mathbb{X}}$. This does not necessarily hold for the DF layer, due to the sparsity of the sample states x_w. Hence, we have to define when a switch between heterogeneous layers is allowed.

A switch (assuming it exists) from the DB layer to the DF layer is always from one state $\hat{x} \in \hat{\mathbb{X}}$ to one state $x_w \in \mathbb{X}_w$, but a switch from the DF layer to the DB layer is from one state x_w to a set of states in $\hat{\mathbb{X}}$, denoted as $\hat{A}(x_w) \subseteq \hat{\mathbb{X}}$. We define the conditions of switching between the DF layer and the DB layer as follows.

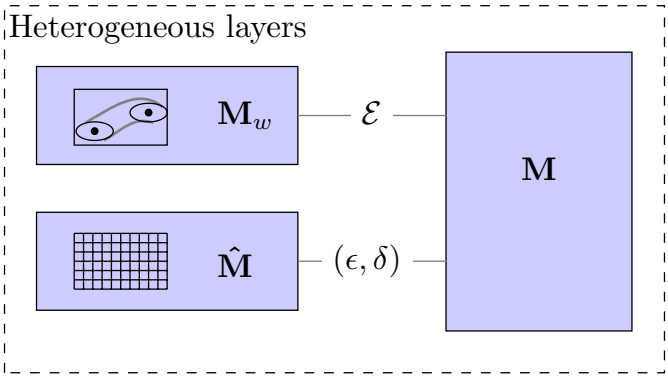


Figure 4.12: Graphical representation of the multi-layered approach with heterogeneous layers.

Definition 4.4 (Conditions for switching between heterogeneous layers)**.** *Given models* $\mathbf{M}$, $\mathbf{M}_w$ (4.27) *and* $\hat{\mathbf{M}}$, *ellipsoidal sets* $\mathcal{E}$, *and simulation relation* $\mathcal{R} \subset \hat{\mathbb{X}} \times \mathbb{X}$, *such that we have* $\hat{\mathbf{M}} \preceq_\epsilon^\delta \mathbf{M}$.

1. *A switch* s_{bf} *from state* $\hat{x}$ *to state* x_w *is possible, if* $\forall x \in \mathcal{R}(\hat{x}) : (x_w, x) \in \mathcal{E}$ *holds.*

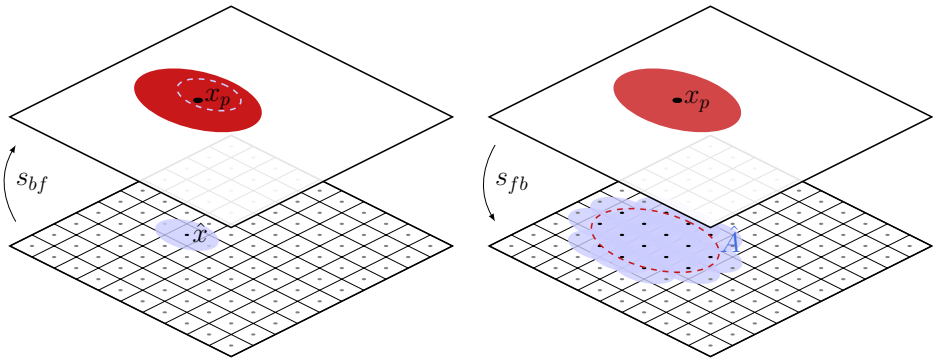
2. *A switch* s_{fb} *from state* x_w *to subset* $\hat{A}(x_w)$ *is possible, if* $\forall x \in \mathcal{E}(x_w), \exists \hat{x} \in \hat{A}(x_w) : (\hat{x}, x) \in \mathcal{R}$ *holds.*

Remark 4.3. When there are multiple DB layers with different precision, the conditions in Definition 4.4 are determined for each DB layer.

The conditions of Definition 4.4 can equivalently be written as 1. $\mathcal{R}(\hat{x}) \subseteq \mathcal{E}(x_w)$ and 2. $\mathcal{E}(x_w) \subseteq \mathcal{R}(\hat{A}(x_w))$. In Figure 4.13 we illustrate these conditions for a specific state and subset $\hat{A}(x_w)$.

4.5c Heterogeneous dynamic programming

To distinguish between the value function of the different layers, we denote the value function of the DF layer and DB layer respectively as $V_{x_w} : \mathbb{X}_w \times Q \to [0, 1]$ and $V_{\hat{x}} : \hat{\mathbb{X}} \times Q \to [0, 1]$. Consider time horizon $[1, \ldots, N]$, then we initially update the value functions of both layers *separately*. To this end, define the updated value functions as $V_{x_w}^{\text{next}} := \mathbf{T}_{x_w}^{\mu_k}(V_{x_w,N-k}^{\boldsymbol{\mu}_k})$ and $V_{\hat{x}}^{\text{next}} := \mathbf{T}_{\hat{x}}^{\mu_k}(V_{\hat{x},N-k}^{\boldsymbol{\mu}_k})$ with the operators defined in respectively (4.28) and (4.8). After computing the next iteration of the value function for both the DF layer and DB layers, we optimize the switching strategy for each layer. More specifically, we take the maximum of either staying in the same layer or switching to the other layer to implicitly determine the switching strategy. Furthermore, we take the conditions as defined in Definition 4.4 into account.



(a) For a specific state $\hat{x}$ the states $x \in \mathcal{R}(\hat{x})$ are indicated by the light blue ellipsoid, which projected on the PRM layer yields the dashed ellipsoid. The states $(x_w, x) \in \mathcal{E}$ are shown in red.

(b) For a specific state x_w the states $x \in \mathcal{E}(x_w)$ are indicated by the red ellipsoid, which projected on the DB layer yields the dashed ellipsoid). The states $(\hat{x}, x) \in \mathcal{R}$ for $\hat{x} \in \hat{A}$ are shown in light blue. The set $\hat{A}$ consists of the states represented by black dots.

Figure 4.13: Following respectively conditions 1 and 2 of Definition 4.4, a possibility of switching action s_{bf} from state $\hat{x}$ towards the state x_w is shown in **(a)**. A possibility of switching action s_{fb} from state x_w towards a subset $\hat{A}(x_w)$ is shown in **(b)**.

For the DF layer, we compute the value function based on $V^{\text{next}}_{x_w}$ and $V^{\text{next}}_{\hat{x}}$ Since, it is not known towards which specific state $\hat{x} \in \hat{A}(x_w)$ we switch, to preserve a lower bound on the satisfaction probability, we consider the worst-case possibility. The value function that takes a possible switch into account can now be given as

$$V^{\boldsymbol{\mu}_{k-1}}_{x_w, N-k+1}(x_w, q) = \max\left\{ V^{\text{next}}_{x_w}(x_w, q), \min_{\hat{x} \in \hat{A}(x_w)} \{V^{\text{next}}_{\hat{x}}(\hat{x}, q)\} \right\}, \tag{4.29}$$

with $\hat{A}(x_w) \subset \hat{\mathbb{X}}$ being the set as defined in point 2 of Definition 4.4 if a switch is possible and the empty set, denoted as $\emptyset$ otherwise. If a switch is not possible, hence $\hat{A}(x_w) = \emptyset$, then $\min_{\hat{x} \in \hat{A}(x_w)} \{V^{\text{next}}_{\hat{x}}(\hat{x}, q)\}$ equals zero.

Remark 4.4. When it is allowed to switch from the DF layer towards multiple DB layers according to Definition 4.4, an additional max-operator is required to determine the optimal DB layer to switch towards.

For the DB layer, we compute the value function as

$$V^{\boldsymbol{\mu}_{k-1}}_{\hat{x}, N-k+1}(\hat{x}, q) = \max\{V^{\text{next}}_{\hat{x}}(\hat{x}, q), V^{\text{next}}_{x_w}(\mathcal{S}_{bf}(\hat{x}), q)\}. \tag{4.30}$$

Here, $\mathcal{S}_{bf} : \hat{\mathbb{X}} \to \mathbb{X}_w \cup x_\emptyset$, with $x_\emptyset$ an auxiliary state, for which we get $V^{\text{next}}_{x_w}(x_w, q) = V^{\text{next}}_{x_w}(x_\emptyset, q) \equiv 0$, is a function defined as follows

$$\mathcal{S}_{bf}(\hat{x}) = \begin{cases} x_w \text{ if } s_{bf} \text{ is possible from } \hat{x} \text{ to } x_w \text{ according to condition 2 in Def. 4.4,} \\ x_\emptyset \text{ otherwise.} \end{cases}$$

It is trivial to prove that the operations in (4.29) and (4.30) are *monotonically increasing* and *upper bounded by one*. First of all, we can see that (4.29) and (4.30) consist of max- and min-operators and the operators defined in (4.28) and (4.8). Since (4.28) is a minor adaptation of the standard DP operator in (4.8), we can follow Proposition 4.1 to conclude that it is monotonically increasing. Next, we can follow the proof of Lemma 1 in Haesaert and Soudjani (2020) and show that all of the operators that build up these operations preserve an inequality, such as $V(x,q) \geq W(x,q)$.

The overall value function, that is the value function of the total model with multiple heterogeneous layers is denoted as $\bar{V} : (\mathbb{X}_w \sqcup \hat{\mathbb{X}}) \times Q \to [0,1]$, with $\sqcup$ the disjoint union. This operator differs from the normal union operator $\bigcup$, by keeping the original set membership as a distinguishing characteristic of the union set. Given policy $\bar{\boldsymbol{\mu}}_k = (\bar{\mu}_{k+1}, \ldots, \bar{\mu}_N)$, with $\bar{\mu} : (\mathbb{X}_w \sqcup \hat{\mathbb{X}}) \times Q \to \mathbb{X}_w \times \hat{\mathbb{U}}$, the overall value function is computed iteratively as $\bar{V}^{\bar{\boldsymbol{\mu}}_{k-1}}_{N-k+1} = \bar{\mathbf{T}}^{\bar{\mu}_k}(\bar{V}^{\bar{\boldsymbol{\mu}}_k}_{N-k})$. Here, the overall Bellman-operator $\bar{\mathbf{T}}^{\bar{\mu}_k}(\bar{V}^{\bar{\boldsymbol{\mu}}_k}_{N-k})$ is defined as

$$\bar{\mathbf{T}}^{\bar{\mu}_k}(\bar{V}^{\bar{\boldsymbol{\mu}}_k}_{N-k}) = \begin{cases} (4.29) \text{ for } x_w \in \mathbb{X}_w \\ (4.30) \text{ for } \hat{x} \in \hat{\mathbb{X}}. \end{cases} \tag{4.31}$$

The complete procedure of the value iteration is illustrated in Figure 4.14.

Remark 4.5. When there are multiple DB layers with different precision, Figure 4.14 becomes more evolved, and among other changes, equation (4.12) is used instead of (4.8) to take into account switching between the DB layers.

We can now conclude the following about the overall Bellman-operator in (4.31).

Theorem 4.2. *Given any policy $\bar{\boldsymbol{\mu}}$. Suppose that the operations in (4.29) and (4.30) are* monotonically increasing *and* upper bounded by one*, then also the overall Bellman-operator $\bar{\mathbf{T}}^{\bar{\mu}_k}(\bar{V}^{\bar{\boldsymbol{\mu}}_k}_{N-k})$ as in (4.31) is* monotonically increasing *and* upper bounded by one.

Proof. The overall Bellman-operator $\bar{\mathbf{T}}^{\bar{\mu}_k}(\bar{V}^{\bar{\boldsymbol{\mu}}_k}_{N-k})$ as in (4.31) is either computed as (4.29) or (4.30) depending on the considered state. Since both operations in (4.29), and (4.30) are monotonically increasing and upper bounded by one, this also holds for the overall Bellman-operator $\bar{\mathbf{T}}^{\bar{\mu}_k}(\bar{V}^{\bar{\boldsymbol{\mu}}_k}_{N-k})$. □

As before, the value function gives the probability of satisfying the specification in 1 to ∞ time step, by including the first time instance based on x_0, we can compute the robust satisfaction probability, that is

$$\bar{\mathbb{R}}^{\bar{\boldsymbol{\mu}}} := \max\left(\mathbf{1}_{Q_f}(\bar{q}_0), \bar{V}^{\bar{\mu}}_{\infty}\big((x_{w,0}, \hat{x}_0), \bar{q}_0\big)\right), \tag{4.32}$$

with $\bar{q}_0 = \tau_{\mathcal{A}_\phi}(q_0, L(h(x_0)))$, and with $x_{w,0} \in \mathcal{E}^{-1}(x_0)$ and $\hat{x}_0 \in \mathcal{R}^{-1}(x_0)$.

Following the same reasoning as in Section 4.4d, we can now conclude that the robust satisfaction probability in (4.32) computed through the overall Bellman-operator $\bar{\mathbf{T}}^{\bar{\mu}_k}(\bar{V}^{\bar{\boldsymbol{\mu}}_k}_{N-k})$ in (4.31) provides a lower bound on the actual satisfaction probability $\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi)$.

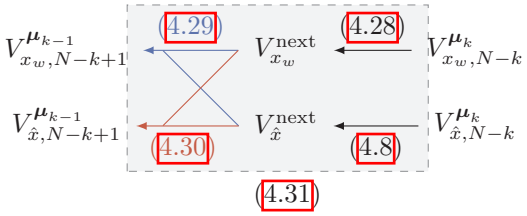


Figure 4.14: Graphical representation of the value iteration for heterogeneous layers. The operator in (4.31) is equivalent to the complete operation in the gray box. The arguments (x_w, q) and $(\hat{x}, q)$ of the value functions in respectively the top and bottom rows are omitted for simplicity.

4.6 Results

To show the benefits of the multi-layered approach (with homogeneous layers), we consider several case studies with increasing complexity. Next, we show that the multi-layered approach with heterogeneous layers is promising by implementing it in a simple example. All simulations are performed on a computer with a 2.3 GHz Quad-Core Intel Core i5 processor and 16 GB 2133 MHz memory, and use the toolbox `SySCoRe` described in Chapter 3 as a basis. For each case study, we mention the computation time and memory usage. Here, the memory usage is computed as the size of the matrices stored in the workspace.

4.6a Simple reach-avoid specification

Consider a simple reach-avoid specification to park a car in a one-dimensional (1D) and two-dimensional (2D) space.

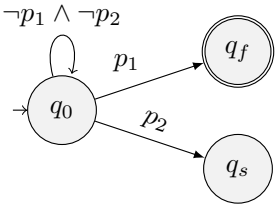
Case 1: parking a car in a 1D space
We consider parking a car in a one-dimensional space. The goal of the controller is to guarantee that the car parks in area P_1, without going through area P_2. This specification can be written as

$$\phi_{park} = \neg p_2 \,\mathsf{U}\, p_1, \tag{4.33}$$

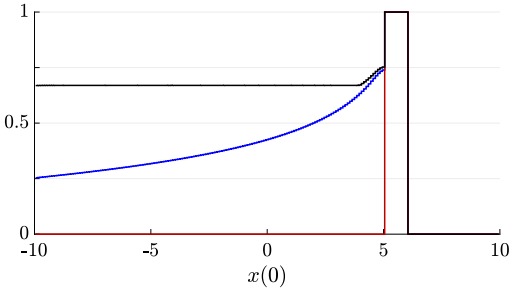
where the labels p_1, and p_2 correspond to respectively regions P_1 and P_2, and can be represented by the DFA in Figure 4.15a.

The dynamics of the car are modeled using an LTI stochastic difference equation as in (4.21) with $A = 0.9, B = 0.5$ and $B_w = C = 1$. We used states $x \in \mathbb{X} = [-10, 10]$, inputs $u \in \mathbb{U} = [-1, 1]$, outputs $y \in \mathbb{Y} = \mathbb{X}$ and Gaussian disturbance $w \sim \mathcal{N}(0, 0.5)$. Furthermore, we have regions $P_1 = [5, 6\rangle, P_2 = [6, 10]$ defined on the output space $\mathbb{Y}$ and labeled as respectively p_1 and p_2.

For this case study, we decided to have two layers and followed Algorithm 4.1 to determine a switching strategy. To this end, we set the first layer with $\mathcal{R}_1$ with



(a) DFA $\mathcal{A}_{\phi_{park}}$. The initial, final, and sink state are denoted by respectively q_0, q_f, and q_s.



(b) Robust satisfaction probability, where the red and blue lines are obtained with respectively only using $\mathcal{R}_1$, $(\epsilon, \delta) = (0.5, 0)$ and $\mathcal{R}_2$, $(\epsilon, \delta) = (0.2, 0.01)$. Switching between these two simulation relations with switching strategy (4.34) yields the black line.

Figure 4.15: DFA associated with specification $\phi_{park} = \neg p_2 \,\mathsf{U}\, p_1$ in **(a)**, and in **(b)** the robust satisfaction probability of the 1D car park example obtained for different simulation relations.

bounds $(\epsilon_1, \delta_{11}) = (0.5, 0)$ and the second layer with $\mathcal{R}_2$ with deviation $\delta_{22} = 0.0168$. We chose $\delta_{12} = 0.168$ and precision $\epsilon_2 = 0.2$ to satisfy Lemma 4.2. Next, we obtained a surrogate model by partitioning with regions of size 0.4 and found the optimal switching strategy[2] corresponding to this grid as

$$\mu^s(\hat{x}, q, i) = \begin{cases} s_{11} & \text{if } -10 \leq \hat{x} \leq 4.8 \text{ and } i = 1 \\ s_{12} & \text{if } 4.8 < \hat{x} \leq 10 \text{ and } i = 1 \\ s_{21} & \text{if } -10 \leq \hat{x} \leq 2.8 \text{ and } i = 2 \\ s_{22} & \text{if } 2.8 < \hat{x} \leq 10 \text{ and } i = 2, \end{cases} \tag{4.34}$$

with i denoting the layer.

Next, we constructed abstract model $\hat{\mathbf{M}}$ in the form of (4.22) by partitioning with regions of size 0.1 with $\mathcal{B} = [-0.05, 0.05]$ and $\hat{u} \in \hat{\mathbb{U}} = \left[-1, -\frac{2}{3}, -\frac{1}{3}, \ldots, 1\right]$. We quantified the accuracy of $\hat{\mathbf{M}}$ with a bi-layered simulation relation with $\mathcal{R}_1$ and $\mathcal{R}_2$ and obtained the satisfaction probability in Figure 4.15b. The average[3] computation time is 22.6 seconds while using a memory of 21.2 MB.

A fixed precision with either simulation relation $\mathcal{R}_1$ or $\mathcal{R}_2$ yields the conservative satisfaction probability indicated by the respective red and blue lines in Fig 4.15b. The bi-layered method (black line) takes advantage of both simulation relations. Close to the parking areas simulation relation $\mathcal{R}_2$ is generally active, which compared to simulation relation $\mathcal{R}_1$ gives us a non-zero satisfaction probability. Switching to layer 1 limits the rapid decrease of the satisfaction probability further from the parking areas, which is normally caused by the relatively high value of δ_{22}.

[2] In this case the switching strategy is the same for all DFA states $q \in Q$.
[3] we took the average over 5 computations and observed an average standard deviation of 2.1%.

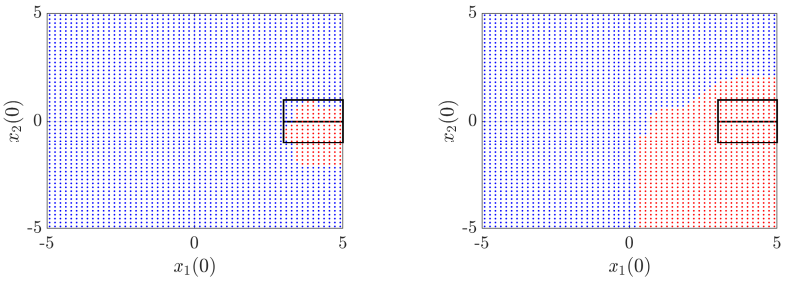
Case 2: parking a car in 2D space

For this case study the specification is the same, that is ϕ_{park} as in (4.33). The dynamics of the car are modeled using an LTI stochastic difference equation as in (4.21) with $A = 0.9I_2, B = 0.5I_2$ and $B_w = C = 1$. We used states $x \in \mathbb{X} = [-5, 5]^2$, inputs $u \in \mathbb{U} = [-1, 1]^2$, outputs $y \in \mathbb{Y} = \mathbb{X}$ and Gaussian disturbance $w \sim \mathcal{N}(0, 0.5)$. Furthermore, we have regions $P_1 = [3, 5] \times [-1, 0]$and $P_2 = [3, 5] \times [0, 1]$ defined on the output space $\mathbb{Y}$ and labeled as respectively p_1 and p_2.

We computed deviation bounds $(\boldsymbol{\epsilon}, \boldsymbol{\delta})$ that satisfy Lemma 4.2 as

$$\boldsymbol{\epsilon} = \begin{bmatrix} 0.5 & 0.2 \end{bmatrix}, \quad \boldsymbol{\delta} = \begin{bmatrix} 0 & 0.168 \\ 0 & 0.0169 \end{bmatrix}.$$

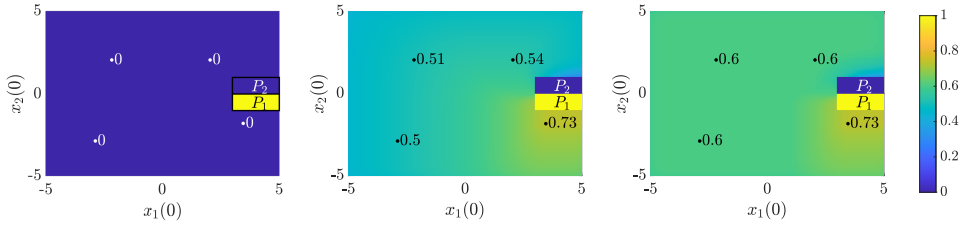
Next, we obtained a surrogate model by partitioning with 55×55 grid cells and found the optimal switching strategy[1] corresponding to this grid as illustrated in Figure 4.16.



(a) Switching strategy for layer 1. The blue and red dots correspond to s_{11} and s_{12} respectively.

(b) Switching strategy for layer 2. The blue and red dots correspond to s_{21} and s_{22} respectively.

Figure 4.16: Switching strategy for the states of the surrogate model of the 2D car park case study. Here, a blue resp. red dot represents switching to layer 1 resp. layer 2. The black boxes indicate regions P_1 (bottom) and P_2 (top).



(a) Single layer, with $(\epsilon_1, \delta_1) = (0.5, 0)$.

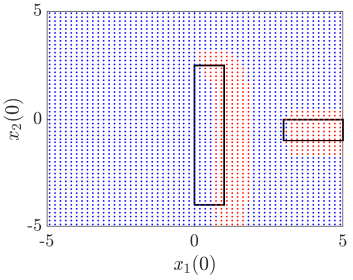
(b) Single layer, with $(\epsilon_2, \delta_2) = (0.2, 0.0169)$.

(c) Multiple layers with simulation relation $\mathcal{R}$.

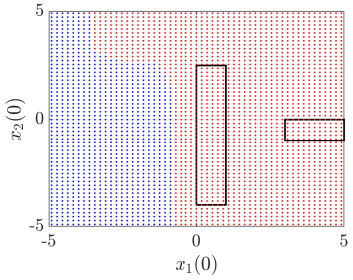
Figure 4.17: Robust satisfaction probability of the 2D car park case study where a single layer is used in **(a)** and **(b)**. A multi-layered simulation relation, with switching strategy as in Figure 4.16 is used in **(c)**.

Then, we constructed abstract model $\hat{\mathbf{M}}$ in the form of (4.22) by partitioning with 283×283 regions leading to $\mathcal{B} = [-0.0353, 0.0353]^2$. We quantified the accuracy of $\hat{\mathbf{M}}$ with a bi-layered simulation relation $\mathfrak{R}$ and obtained the robust satisfaction probability in Figure 4.17c. The average[4] computation time is 78.8 seconds while using a memory of 243 MB. The robust satisfaction probability of the multi-layered approach is higher everywhere compared to only using a single layer, with either $\mathcal{R}_1$, with $(\epsilon_1, \delta_1) = (0.5, 0)$ or $\mathcal{R}_2$, with $(\epsilon_2, \delta_2) = (0.2, 0.0169)$ as can be seen in Figure 4.17.

Similar results are obtained when moving the avoid region to $P_2 = [0, 1] \times [-4, 2.5]$ as can be seen in Figure 4.19. However, close to region P_1 the robust satisfaction probability is not increased by using a multi-layered approach. The obtained switching strategy is shown for the surrogate model in Figure 4.18.

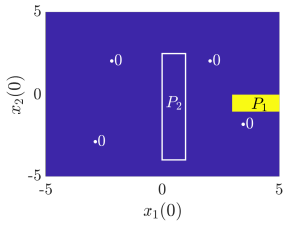


(a) Switching strategy for layer 1. The blue and red dots correspond to s_{11} and s_{12} respectively.

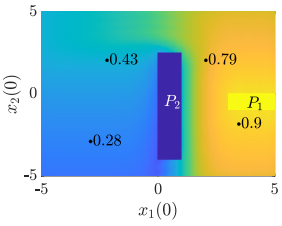


(b) Switching strategy for layer 2. The blue and red dots correspond to s_{21} and s_{22} respectively.

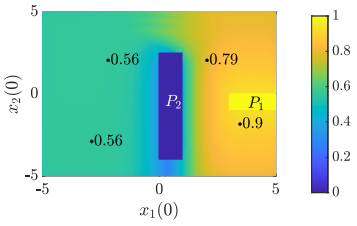
Figure 4.18: Switching strategy for the states of the surrogate model of the 2D car park case study with $P_2 = [0, 1] \times [-4, 2.5]$. Here, a blue resp. red dot represents switching to layer 1 resp. layer 2. The black boxes indicate regions P_1 (right) and P_2 (left).



(a) Single layer, with $(\epsilon_1, \delta_1) = (0.5, 0)$.



(b) Single layer, with $(\epsilon_2, \delta_2) = (0.2, 0.0169)$.



(c) Multiple layers with simulation relation $\mathfrak{R}$.

Figure 4.19: Robust satisfaction probability of the 2D car park case study with $P_2 = [0, 1] \times [-4, 2.5]$where a single layer is used in **(a)** and **(b)**. A multi-layered simulation relation, with switching strategy as in Figure 4.18 is used in **(c)**.

[4] we took the average over 5 computations and observed an average standard deviation of 2.2%.

4.6b Complex reach-avoid specification

Next, we consider a complex reach-avoid specification that is cyclic.

Case 3: Package delivery
In this case, we consider multiple regions. A pick-up region P_1, a delivery region P_3, a strict avoid region P_4, and a region where you lose a package P_2. The goal of the controller is to make sure that a package is picked-up at P_1 and delivered to P_3 while avoiding P_4. Region P_2 is fine to visit without a package, but once crossed while carrying a package, the package is lost and a new package has to be picked up from P_1. This specification is written in scLTL as $\phi_{PD} = \neg p_4 \,\mathsf{U}\, (p_1 \wedge (\neg(p_4 \vee p_2) \,\mathsf{U}\, p_3))$, with its corresponding DFA given in Figure 4.20. The regions are defined on the output space $\mathbb{Y}$ as $P_1 = [-4,-3]^2, P_2 = [0,1] \times [0,2.5], P_3 = [3,5] \times [-2,-0.5]$ and $P_4 = [0,1] \times [-4,0]$, and are labeled as respectively p_1, p_2, p_3 and p_4. The dynamics of the car are similar as before, except for the slightly enlarged input space $\mathbb{U} = [-1.25, 1.25]^2$ and a smaller stochastic disturbance $w \sim \mathcal{N}(0.0.25)$.

We decided to have two layers and followed Algorithm 4.1 to determine a switching strategy. To this end, we set the first layer with $\mathcal{R}_1$ with bounds $(\epsilon_1, \delta_{11}) = (0.3, 0)$ and the second layer with $\mathcal{R}_2$ with deviation $\delta_{22} = 0.0160$. We chose $\delta_{12} = 0.1586$ and precision $\epsilon_2 = 0.3$ to satisfy Lemma 4.2. Hence, we have total deviation bounds

$$\boldsymbol{\epsilon} = \begin{bmatrix} 0.5 & 0.3 \end{bmatrix}, \quad \boldsymbol{\delta} = \begin{bmatrix} 0 & 0.1586 \\ 0 & 0.0160 \end{bmatrix}.$$

Next, we obtained a surrogate model by partitioning with 55×55 grid cells and found the optimal switching strategy corresponding to this grid as illustrated in Figure 4.21. Here, the blue and red dots correspond to switching to layer 1 and layer 2 respectively. From this figure, we can see that this approach is indeed guided by the specification, since depending on the DFA state, the *role* of region P_2 (either *not relevant* or *try to avoid*) is different and the switching strategy changes accordingly.

Then, we constructed abstract model $\hat{\mathbf{M}}$ in the form of (4.22) by partitioning the state space with 283×283 regions and the input space with 3×3 regions. We quantified the accuracy of $\hat{\mathbf{M}}$ with a bi-layered simulation relation with $\mathcal{R}_1$ and $\mathcal{R}_2$ as before and obtained the robust satisfaction probability in Figure 4.22c and 4.22e. The average[5] computation time is 123 seconds while using a memory of 340 MB. The robust satisfaction probability of the multi-layered approach is higher everywhere compared to only using a single layer, as can be seen in Figure 4.22.

[5] we took the average over 5 computations and observed an average standard deviation of 1.1%.

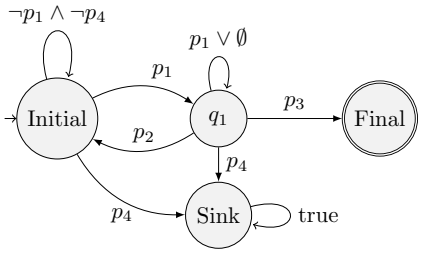
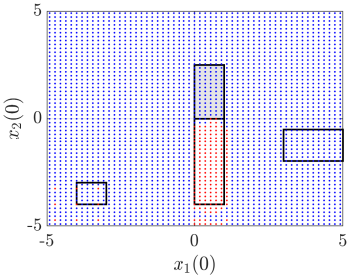
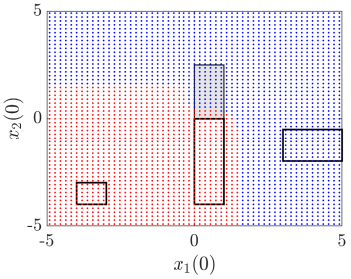


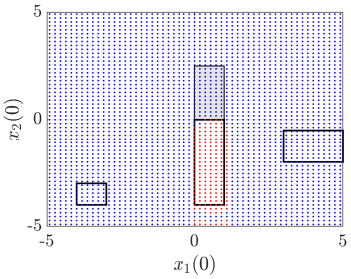
Figure 4.20: Cyclic DFA corresponding to the specification of case 3, package delivery. Here, $\emptyset$ denotes the *empty set*, which means that all atomic propositions $p \in \text{AP}$ are false.



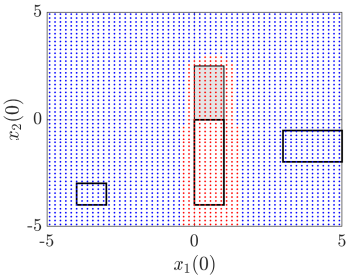
(a) Switching strategy for DFA state q_0 and layer 1. The blue and red dots corresponds to s_{11} and s_{12} respectively.



(b) Switching strategy for DFA state q_0 and layer 2. The blue and red dots corresponds to s_{21} and s_{22} respectively.

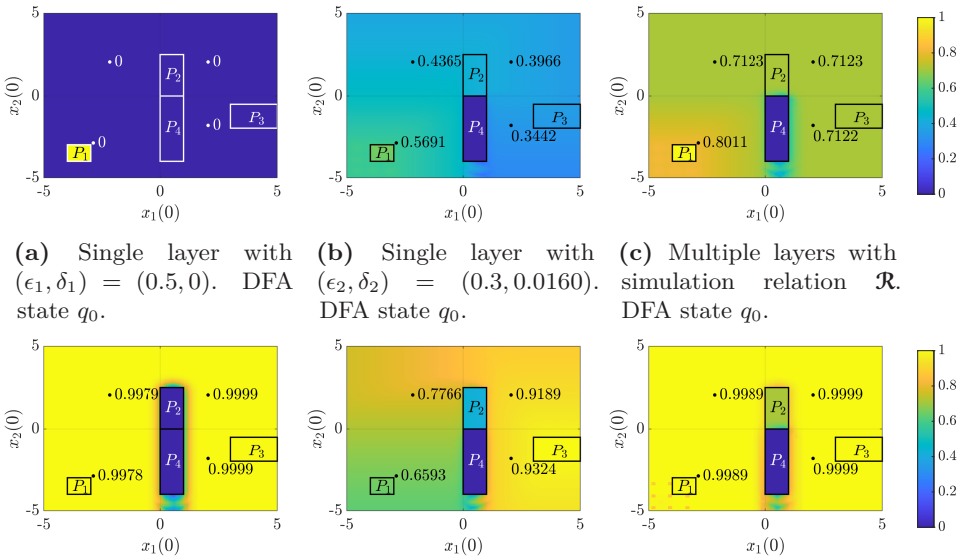


(c) Switching strategy for DFA state q_1 and layer 1. The blue and red dots corresponds to s_{11} and s_{12} respectively.



(d) Switching strategy for DFA state q_1 and layer 2. The blue and red dots corresponds to s_{21} and s_{22} respectively.

Figure 4.21: Switching strategy for the states of the surrogate model of the package delivery case study for DFA states q_0 (Initial) in **(a)**, **(b)** and q_1 in **(c)**, **(d)**. Here, a blue resp. red dot represents switching to layer 1 resp. layer 2. The black boxes indicate regions P_1, P_2, P_3, and P_4. Region P_2 is colored gray.



(a) Single layer with $(\epsilon_1, \delta_1) = (0.5, 0)$. DFA state q_0.

(b) Single layer with $(\epsilon_2, \delta_2) = (0.3, 0.0160)$. DFA state q_0.

(c) Multiple layers with simulation relation $\mathcal{R}$. DFA state q_0.

(d) Single layer with $(\epsilon_1, \delta_1) = (0.5, 0)$. DFA state q_1.

(e) Single layer with $(\epsilon_2, \delta_2) = (0.3, 0.0160)$. DFA state q_1.

(f) Multiple layers with simulation relation $\mathcal{R}$. DFA state q_1.

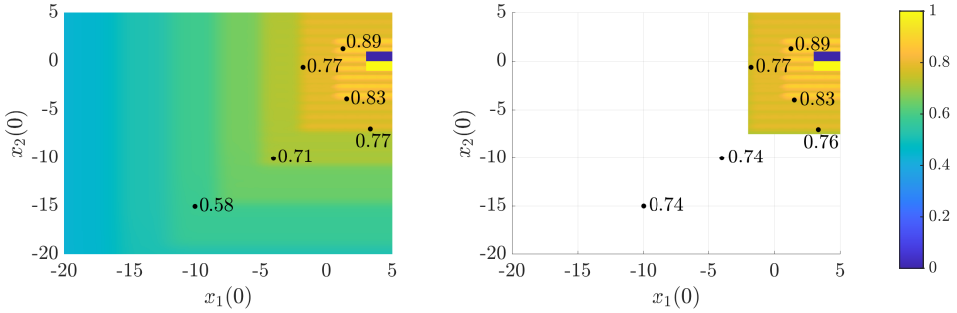
Figure 4.22: Robust satisfaction probability of package delivery case for DFA state q_0 (Initial) in **(a)**-**(c)** and for DFA state q_1 in **(d)**-**(f)**. Here, a single layer is used in **(a)**,**(b)**, and **(d)**,**(e)**. A multi-layered simulation relation, with switching strategy as in Figure 4.21 is used in **(c)** and **(f)**.

4.6c Case study with heterogeneous layers

To show that the multi-layered approach with heterogeneous layers is promising, we apply it to a toy case. We consider a case study similar to Case 2 of parking a car in a 2D space. We consider the same specification ϕ_{park} as in (4.33) with DFA as in Figure 4.15a.

The dynamics of the car are modeled using an LTI stochastic difference equation as in (4.21) with $A = 0.9I_2, B = 0.5I_2, B_w = \sqrt{0.25}I_2$ and $C = I_2$. Note that B_w is used to decrease the influence of the stochastic disturbance substantially. We used a larger state space than before, namely $x \in \mathbb{X} = [-20, 5]$, inputs $u \in \mathbb{U} = [-5, 5]$, outputs $y \in \mathbb{Y} = \mathbb{X}$ and Gaussian disturbance $w \sim \mathcal{N}(0, I)$. Furthermore, we have initial state $x_0 = [-10, -15]^\top$ and regions $P_1 = [3, 5] \times [-1, 0]$, and $P_2 = [3, 5] \times [0, 1]$ defined on the output space $\mathbb{Y}$.

For this case study, we chose two layers, one discretization-free, and one discretization-based layer. For the discretization-free layer, we chose $\Delta_w = 1 - n_s \cdot 10^{-6}$, with the number of steps between each waypoint fixed to $n_s = 3$ steps. We have matrix $D_w = I_2$ for the set $\mathcal{E}$ as in (4.26). To decrease the size of the set $\mathcal{E}$, we used an interface function (4.11) equal to $u_t = u_{w,t} + K(x_t - x_{w,t})$, with $K = -1.4I_2$. Following the derivation in the appendix, we computed $\epsilon_w = 3.28$ for set $\mathcal{E}$ in (4.26).



(a) Single-layered approach with only discretization-based layers

(b) Multi-layered approach with heterogeneous layers

Figure 4.23: Robust satisfaction probability of the toy case. Single-layered with only discretization-based approach in **(a)** and multi-layered approach with heterogeneous layers in **(b)**.

The considered waypoints of the model equal $x_{w,0} = [-10, -15]^\top, x_{w1} = [-4, -10]^\top$, and $x_{w2} = [1.5, -4]^\top$ and are all labeled with the empty label, which is equivalent to $\neg p_1 \wedge \neg p_2$.

For the discretization-based layer, we computed a finite-state abstraction as in (4.22) by gridding part of the state space $[-2, 5] \times [-7.5, 5]$ in both directions with 150 grid cells. We computed the similarity quantification between the original model $\mathbf{M}$ (4.21) and its finite-state abstraction (4.22) for interface function $u = \hat{u}$, and obtained $(\epsilon, \delta) = (0.1, 0.068)$.

For this model with heterogeneous layers, we performed a value iteration and synthesized a controller using the technique described in Section 4.5c. The robust satisfaction probability is shown in Figure 4.23b. The satisfaction probability for the sample states is shown in black. Some points of the discretization-based layer with the associated probability are indicated in red. The average[6] computation time equals 14.5 seconds, while using a memory of 49.2 MB. Figure 4.23a shows the robust satisfaction probability obtained by using only a discretization-based layer with a similar grid resolution as for the discretization-based part when heterogeneous layers are used (536×300 grid cells). The average[7] computation time equals 20.2 seconds, while using a memory of 418 MB. Comparing both figures in Figure 4.23, we can see that the accuracy of the discretization-based layer is slightly lower than when only using a discretization-based approach. This is due to the smaller state space that is gridded. Transitions going out of this smaller state space have a 0 satisfaction probability, hence, on the boundaries of the state space, the robust satisfaction probability is slightly lower compared to covering the complete state space. However, the robust satisfaction probability of the discretization-free layer in Figure 4.23b is higher than when using only a discretization-based layer. Only using a discretization-free layer over the complete state space gives a robust probability

[6] we took the average over 5 computations and observed an average standard deviation of 2.9%.
[7] we took the average over 5 computations and observed an average standard deviation of 2.7%.

of 0 for all samples $x_w \in \mathbb{X}_w$, due to the large set $\mathcal{E}$.

From these results, we conclude that the multi-layered approach with heterogeneous layers is promising. It outperforms the discretization-based techniques with respect to computation time and memory usage and the discretization-free technique as explained in this paper with respect to accuracy. It should be noted that the discretization-free technique employed in this paper is very basic and conservative. Hence, using more advanced methods the results with heterogeneous layers are expected to improve even further.

4.7 Conclusion

In this chapter, we derive a multi-layered approach that allows switching between multiple simulation relations (homogeneous layers) and different abstraction techniques (heterogeneous layers). This approach makes it possible to use the advantages of each individual layer, hence leading to a potentially accurate and efficient computational approach for temporal logic control. We have illustrated the benefit of the multi-layered method with homogeneous or heterogeneous layers by applying it to multiple case studies. They all show good results with respect to accuracy and the case study with heterogeneous layers indicates a significant increase in computational efficiency.

An interesting topic for future work is to allow multiple abstract (discretization-based) models with a different grid size, hence multiple grid resolutions. This should improve the efficiency of the approach even further, without reducing the accuracy.

The theory on combining heterogeneous layers and how to switch between them has been fully developed, however, the technique used in the case study with a discretization-free layer is very conservative and there exist more efficient and more accurate methods that work directly on stochastic systems (Vinod et al. 2019; Jagtap et al. 2020; Engelaar et al. 2023). In this chapter, we illustrated that the multi-layered approach with heterogeneous layers is promising through a simple case study, but to achieve its full potential, a more suitable technique should be used in the discretization-free layer. This is part of ongoing research.

Appendix

4.A Derivation for the case study in Section 4.6c

We want to find ϵ_w, or equivalently the ellipsoidal set $\mathcal{E}$ as in (4.26), such that for all states corresponding to the ellipsoid around the waypoint, the next states remain inside the ellipsoid of the next waypoint, with a probability of $\Delta_w(x_w, x'_w)$. Mathematically, this can be formulated as follows:

$$\forall x \in \mathcal{E}(x_w), \forall u_w \in \mathbb{U}_w : x' \in \mathcal{E}(x'_w) \text{ with probability } 1 - \Delta_w(x_w, x'_w)$$

This is equivalent to

$$\forall x \in \mathcal{E}(x_w), \forall u_w \in \mathbb{U}_w : ||x' - x'_w||_{D_w} = ||\bar{A}(x - x_w) + B_w w||_{D_w} \leq \epsilon_w, \quad (4.35)$$

with $\bar{A} = A + BK$. Here, the interface function $u = u_w + K(x - x_w)$ has been substituted. Using standard properties of the vector norm (Belitskii and Lyubich 1988, Chapter 1), we find the following inequalities

$$\begin{aligned} ||\bar{A}(x - x_w) + B_w w||_{D_w} &\leq ||\bar{A}(x - x_w)||_{D_w} + ||B_w w||_{D_w} \\ &\leq ||\bar{A}||_{D_w} ||(x - x_w)||_{D_w} + ||B_w||_{D_w} ||w||_{D_w}. \end{aligned}$$

For all states $x_w \in \mathcal{E}(x_w)$ this implies that

$$||x' - x'_w||_{D_w} \leq ||\bar{A}||_{D_w} \epsilon_p + ||B_w||_{D_w} ||w||_{D_w}. \quad (4.36)$$

For a given probability $\Delta_w(x_w, x'_w)$ and weighting matrix $D_w = d_w I_{n_x}$, with scalar d_w, we can compute a bound on w using the Chi-squared distribution with n_w degrees of freedom (Chew 1966). Hence, we obtain

$$||w||_{D_w} \leq \frac{1}{d_w} \sqrt{r_w}, \text{ with } r_w = \chi^{-1}(\Delta_w(x_w, x'_w) | n_w), \quad (4.37)$$

with $\chi^{-1}(\cdot | n_w)$ denoting the inverse cumulative distribution function of the Chi-squared distribution with n_w degrees of freedom. Intuitively, this means that for a random variable $w \sim \mathcal{N}(0, I)$, we compute the ellipsoidal set (4.37) containing w with a confidence equal to $\Delta_w(x_w, x'_w)$. Substituting in (4.36), we get

$$||x' - x'_w||_{D_w} \leq ||\bar{A}||_{D_w} \epsilon_p + ||B_w||_{D_w} \frac{1}{d_p} \sqrt{r_w}.$$

Hence, to achieve (4.35) ϵ_w should equal

$$\epsilon_w = | - (||\bar{A}||_{D_w} - 1)^{-1} ||B_w||_{D_w} \frac{1}{d_w} \sqrt{r_w}|. \quad (4.38)$$

Part II

Data-driven approaches

5

A Bayesian approach to temporal logic control of uncertain systems

This chapter addresses the problem of data-driven computation of controllers that are correct by design for safety-critical systems and can provably satisfy (complex) functional requirements. Safety-critical systems are best described by stochastic models with a continuous-state space. In this chapter, we focus on continuous-state stochastic systems, with additional uncertainty about the parameters. To handle this additional uncertainty, in the context of control design for these systems, we propose a two-stage approach that decomposes the problem into a learning stage and a robust formal controller synthesis stage. The first stage utilizes available Bayesian regression results to compute robust credible sets for the true parameters of the system. For the second stage, we introduce methods for systems subject to both stochastic and parametric uncertainties. We provide simulation relations for enabling correct-by-design control refinement that are founded on coupling uncertainties of stochastic systems via sub-probability measures. The presented relations are essential for constructing abstract models that are related to not only one model but to a set of parameterized models. The results are demonstrated on a linear and nonlinear model.

5.1 Introduction

The rapid development of Artificial Intelligence (AI) and learning-based methods has changed the face of modern technology. Autonomous cars, smart grids, robotic systems, and medical devices are just a few examples of engineered systems powered by this technology. Most of these systems operate in safety-critical environments, with operational scenarios being uncertain. Despite the undisputed impact of data-driven methods, their premature usage can lead to severe incidents (Axelrod

2013). Ensuring safe operation, or more generally, designing safety-critical systems that behave in some desired manner even if the environment is uncertain, entails synthesizing robust controllers such that the controlled system exhibits the desired behavior with the satisfaction being formally verifiable. Therefore, there is an ever-growing demand for so-called *correct-by-design* approaches, giving formal guarantees on the absence of any undesired behavior of the controlled system.

However, designing control software with robust satisfaction guarantees proves to be very challenging. Most safety-critical systems are large in scale, operate in an uncertain environment (i.e., their state evolution is subject to uncertainty), and comprise both continuous and discrete state variables. Designing controllers for such systems does not grant analytical or closed-form solutions even when an exact model of the system is known. The survey paper Lavaei et al. (2022a) provides an overview of the current state of the art in formal controller synthesis for stochastic systems. Abstraction methods represent a promising solution and enable formal control synthesis w.r.t. high-level requirements (Tabuada 2009; Belta et al. 2017). Existing approaches, however, are limited to small-scale systems and require prior knowledge of the exact stochastic model of the system. These two major shortcomings still prevent the implementation of correct-by-design controllers into real-world systems.

In the pursuit of improving the scalability of formal synthesis approaches, abstraction-based techniques such as model order reduction, adaptive, and compositional methods are exploited (Lavaei et al. 2022a). All of these necessitate formally quantifying the similarity between an *abstract* model and its latent true counterpart. One means of relating the systems is using *simulation relations*. Whilst there exist different definitions (Baier and Katoen 2008; Haesaert and Soudjani 2020; Lavaei et al. 2022a), in essence, simulation relations allow for quantifying the behavioral similarity of two systems and refining controllers designed on the (finite-state) abstractions to the respective original model whilst transferring any guarantees obtained on the abstract model to the original system.

Requiring knowledge of the exact stochastic model of the system implies that any guarantees on the correctness of the closed-loop system only hold for that specific model. Unfortunately, obtaining an exact model of the system of interest is either not possible or expensive and time-consuming. The type of uncertainty arising from incomplete knowledge of the system is called *epistemic uncertainty*. Data-driven identification methods for learning stochastic systems are well studied (Van Den Hof and Schrama 1995; Keesman 2011). These system identification approaches try to find the best parameters that minimize an appropriate distance (either in the time or frequency domain) between the output data and the output trajectories of the identified model. For stochastic systems, there is no result available to relate the satisfaction of temporal requirements by the identified model to that of the original unknown model with respect to the size of the dataset.

In this chapter, we focus on the following.

Problem 5.1. *Design a controller using data from the unknown true system such that the controlled system satisfies a given temporal logic specification with at least probability* $p_\phi \in [0,1]$ *and confidence* $(1-\alpha) \in [0,1]$.

The main contribution of this chapter is to provide an abstraction-based scheme

for answering this question for the class of parameterized discrete-time stochastic systems and the class of syntactically co-safe linear temporal logic (scLTL) specifications (Tkachev et al. 2013). The first stage of our scheme is to utilize available system identification techniques to learn the parameter set that contains the true parameters with a given confidence level from a finite amount of data. In the second stage, we provide closeness guarantees over the whole set of associated models by defining a sub-simulation relation between a set of models and an abstract model, which is founded on coupling uncertainties in stochastic systems via sub-probability measures. We provide theoretical results for establishing this new relation and the associated closeness guarantees for nonlinear parametric systems with additive Gaussian uncertainty. This allows us to design a controller alongside quantified guarantees such that the controlled system satisfies a given probabilistic temporal specification uniformly on this parameter set. Whilst being generally compatible with any robust estimation method providing parameter credible sets (Jaynes and Kempthorne 1976), we demonstrate our approach using Bayesian linear regression (Bishop 2006). This approach makes this chapter the first work that integrates system identification with formal abstraction-based methods.

The remainder of the chapter is organized as follows. After reviewing the related work, we introduce the necessary notions to deal with the stochasticity and uncertainty in Section 5.2. We also give the class of models, the class of specifications, and the problem statement. The system identification framework is presented in Section 5.3. In Section 5.4, we introduce our new notion of sub-simulation relations and control refinement that is based on partial coupling. We also show how to design a controller and use this new relation to give lower bounds on the satisfaction probability of the specification. In Section 5.5, we utilize two parameter estimation methods to obtain credible sets and establish the relation between parametric linear and nonlinear models and their simplified abstract models. Finally, we demonstrate the application of the proposed approach on a linear system and the nonlinear Van der Pol Oscillator in Section 5.6. We conclude the chapter with a discussion in Section 5.7.

Related work: Data-driven formal approaches for systems with no stochastic state transitions are studied in Haesaert et al. (2017c), Makdesi et al. (2021), and Kazemi et al. (2022). For stochastic systems, one approach for dealing with epistemic uncertainty is to model it as a stochastic two-player game, where the objective of the first player is to create the best performance considering the worst-case epistemic uncertainty. The literature on solving stochastic two-player games is relatively mature for finite state systems (Chatterjee and Henzinger 2012; Chatterjee and Doyen 2016). There is a limited number of papers addressing this problem for continuous-state systems. The papers Majumdar et al. (2020) and Majumdar et al. (2021) look at satisfying temporal logic specifications on nonlinear systems utilizing mu-calculus and space discretization. The results rely on direct access to the full state and are hence incompatible with model order reduction techniques, which is often required to tackle the curse of dimensionality introduced by space discretization. The work Badings et al. (2023) addresses epistemic uncertainty in stochastic systems by abstracting the system to an interval Markov decision process (iMDP) and is limited to finite horizons, reach-avoid specifications, and

linear systems. Similarly, Lavaei et al. (2022b) extends upon this by constructing an interval MDP for uncertain nonlinear systems by solving a scenario optimization problem. This is helpful when only data from the system is available or the dynamics are very complex. However, capturing epistemic uncertainty using interval MDPs relies on the assumption that the epistemic uncertainty is state-wise independent, leading to inconsistent and overly conservative results. In contrast, we relate epistemic uncertainty explicitly using parametric MDPs.

The latest work in the area of abstraction-free approaches develops control barrier certificates (CBC) that are based on constructing a set of feasible models (Cohen et al. 2022). Similarly, adaptive control for continuous-time control-affine parametric systems using so-called unmatched control barrier functions is studied in Lopez and Slotine (2023). Whilst CBCs are potentially more scalable than abstraction-based solutions, finding a valid CBC for systems with non-control-affine and non-polynomial dynamics is generally hard. Even more, CBCs for specifications beyond simple reachability yields, e.g., sequential reachability problems (Anand et al. 2021), greatly impede their applicability. Further work on model-free reinforcement learning studies synthesizing robust controllers for temporal logic specifications without constructing a model of the system (Kazemi and Soudjani 2020; Lavaei et al. 2020b).

Our approach utilizes the techniques from the system identification literature to compute parameter sets with respect to a given confidence for stochastic systems. We then build a theoretical foundation on the concepts presented in Chapter 2 of this thesis and in Haesaert et al. (2017b) and Haesaert and Soudjani (2020) to design robust controllers using abstraction methods that are compatible with both model order reduction and space discretization.

5.2 Preliminaries and problem statement

5.2a Preliminaries

The following notions are used. The transpose of a matrix A is indicated by $A^\top$. A measurable space is a pair $(\mathbb{X}, \mathcal{F})$ with sample space $\mathbb{X}$ and σ-algebra $\mathcal{F}$ defined over $\mathbb{X}$, which is equipped with a topology. In this work, we restrict our attention to Polish sample spaces (Bogachev 2007). As a specific instance of $\mathcal{F}$, consider Borel measurable spaces, i.e., $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$, where $\mathcal{B}(\mathbb{X})$ is the Borel σ-algebra on $\mathbb{X}$, that is the smallest σ-algebra containing open subsets of $\mathbb{X}$. A positive *measure* $\mathcal{W}$ on $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$ is a non-negative map $\mathcal{W} : \mathcal{B}(\mathbb{X}) \to \mathbb{R}_{\geq 0}$ such that for all countable collections $\{A_i\}_{i=1}^{\infty}$ of pairwise disjoint sets in $\mathcal{B}(\mathbb{X})$ it holds that $\mathcal{W}(\bigcup_i A_i) = \sum_i \mathcal{W}(A_i)$. A positive measure $\mathcal{W}$ is called a *probability measure* if $\mathcal{W}(\mathbb{X}) = 1$, and is called a *sub-probability measure* if $\mathcal{W}(\mathbb{X}) \leq 1$.

A probability measure $\mathbb{P}_x$ together with the measurable space $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$ define a *probability space* denoted by $(\mathbb{X}, \mathcal{B}(\mathbb{X}), \mathbb{P}_x)$ and has realizations $x \sim \mathbb{P}_x$. We denote the set of all probability measures for a given measurable space $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$ as $\mathcal{P}(\mathbb{X})$. For two measurable spaces $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$ and $(\mathbb{Y}, \mathcal{B}(\mathbb{Y}))$, a *kernel* is a mapping

$\bar{\mathcal{W}} : \mathbb{X} \times \mathcal{B}(\mathbb{Y}) \to \mathbb{R}_{\geq 0}$ such that $\bar{\mathcal{W}}(x, \cdot) : \mathcal{B}(\mathbb{Y}) \to \mathbb{R}_{\geq 0}$ is a measure for all $x \in \mathbb{X}$, and $\bar{\mathcal{W}}(\cdot, B) : \mathbb{X} \to \mathbb{R}_{\geq 0}$ is measurable for all $B \in \mathcal{B}(\mathbb{Y})$. A kernel associates to each point $x \in \mathbb{X}$ a measure denoted by $\bar{\mathcal{W}}(\cdot|x)$. We refer to $\bar{\mathcal{W}}$ as a (sub-)probability kernel if in addition $\bar{\mathcal{W}}(\cdot|x) : \mathcal{B}(\mathbb{Y}) \to [0, 1]$ is a (sub-)probability measure. The normal stochastic kernel with mean $\mu \in \mathbb{R}^n$ and covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$ is defined by

$$\mathcal{N}(\mathrm{d}x|\mu, \Sigma) := \frac{\mathrm{d}x}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp\left[-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right],$$

with $\det(\Sigma)$ denoting the determinant of Σ. The *Dirac delta* measure concentrated at a point $a \in \mathbb{X}$ is denoted as $\delta_a : \mathcal{B}(\mathbb{X}) \to \{0, 1\}$ and is defined on a set $\mathbb{X}$ and for any measurable set $A \subseteq \mathbb{X}$ as

$$\delta_a(A) = \begin{cases} 1 & \text{if } a \in A \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

For given sets A and B, a relation $\mathcal{R} \subset A \times B$ is a subset of the Cartesian product $A \times B$. The relation $\mathcal{R}$ relates $x \in A$ with $y \in B$ if $(x, y) \in \mathcal{R}$, written equivalently as $x\mathcal{R}y$. For a given set $\mathbb{Y}$, a metric or distance function $\mathbf{d}_{\mathbb{Y}}$ is a function $\mathbf{d}_{\mathbb{Y}} : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}_{\geq 0}$ satisfying the following conditions for all $y_1, y_2, y_3 \in \mathbb{Y}$: $\mathbf{d}_{\mathbb{Y}}(y_1, y_2) = 0$ iff $y_1 = y_2$ (reflexive); $\mathbf{d}_{\mathbb{Y}}(y_1, y_2) = \mathbf{d}_{\mathbb{Y}}(y_2, y_1)$ (symmetric); and $\mathbf{d}_{\mathbb{Y}}(y_1, y_3) \leq \mathbf{d}_{\mathbb{Y}}(y_1, y_2) + \mathbf{d}_{\mathbb{Y}}(y_2, y_3)$ (transitive).

5.2b Discrete-time uncertain stochastic systems and control policies

We consider discrete-time nonlinear systems perturbed by additive stochastic noise under model-parametric uncertainty. Consider the model $\mathbf{M}(\theta)$ parametrized with θ:

$$\mathbf{M}(\theta) : \left\{ \begin{array}{ll} x_{t+1} & = f(x_t, u_t; \theta) + w_t, \\ y_t & = h(x_t), \end{array} \right. \tag{5.2}$$

where the system state, input, and output at the t^{th} time-step are denoted by $x_t \in \mathbb{X}, u_t \in \mathbb{U}, y_t \in \mathbb{Y}$, respectively. The functions f and h specify, respectively, the parameterized state evolution of the system and the output map. The additive noise is denoted by $w_t \in \mathbb{R}^{n_x}$, which is an independent, identically distributed (i.i.d.) noise sequence with distribution $w_t \sim \mathbb{P}_w(\cdot)$.

We indicate the input sequence of a model $\mathbf{M}$ by $\mathbf{u} := u_0, u_1, u_2, \ldots$ (respectively, $\mathbf{u}_{[0,N]} := u_0, u_1, u_2, \ldots, u_N$) and we define its (finite) *executions* as sequences of states $\mathbf{x} := x_0, x_1, x_2, \ldots$ (respectively, $\mathbf{x}_{[0,N]} := x_0, x_1, x_2, \ldots, x_N$) initialized with the initial state x_0 of $\mathbf{M}$ at $t = 0$. In each execution, the consecutive state $x_{t+1} \in \mathbb{X}$ is obtained via (5.2).

The execution history $(x_0, u_0, x_1, \ldots, u_{N-1}, x_N)$ grows with the number of observations N and takes values in the *history space* $\mathbb{H}_N := (\mathbb{X} \times \mathbb{U})^N \times \mathbb{X}$. A control policy or controller for $\mathbf{M}(\theta)$ is a sequence of policies mapping the current execution history to a control input. More precisely, we define a control policy as follows.

Definition 5.1 (Control policy)**.** *A control policy* $\boldsymbol{\mu}$ *is a sequence* $\boldsymbol{\mu} = (\mu_0, \mu_1, \mu_2, \ldots)$ *of universally measurable maps* $\mu_t : \mathbb{H}_t \to \mathcal{P}(\mathbb{U}, \mathcal{B}(\mathbb{U}))$, $t \in \mathbb{N} := \{0, 1, 2, \ldots\}$, *from the execution history to a distribution on the input space.*

As special types of control policies, we differentiate Markov policies and finite memory policies. A *Markov policy* $\boldsymbol{\mu}$ is a sequence $\boldsymbol{\mu} = (\mu_0, \mu_1, \mu_2, \ldots)$ of universally measurable maps $\mu_t : \mathbb{X} \to \mathcal{P}(\mathbb{U}, \mathcal{B}(\mathbb{U}))$, $t \in \mathbb{N}$, from the state space $\mathbb{X}$ to a distribution on the input space. We say that a Markov policy is *stationary*, if $\boldsymbol{\mu} = (\mu, \mu, \mu, \ldots)$ for some μ. *Finite memory policies* first map the execution history of the system to a finite set (memory). The input is then chosen similarly to the Markov policy as a function of the system state and the memory state. This class of policies is needed for satisfying temporal specifications on the system executions.

In the following, a control policy for the model (5.2) is denoted by $\mathbf{C}$, and we denote the feedback composition of the model $\mathbf{M}(\theta)$ with $\mathbf{C}$ as $\mathbf{M}(\theta) \times \mathbf{C}$.

In the next subsection, we formally define the class of specifications studied in this chapter.

5.2c Temporal logic specifications

Consider a set of atomic propositions $\mathrm{AP} := \{p_1, \ldots, p_N\}$ that defines an *alphabet* 2^{AP}, where any *letter* $\pi \in 2^{\mathrm{AP}}$ is composed of a set of atomic propositions. An infinite string of letters forms a *word* $\boldsymbol{\pi} = \pi_0\pi_1\pi_2 \ldots \in (2^{\mathrm{AP}})^{\mathbb{N}}$. We denote the suffix of $\boldsymbol{\pi}$ by $\boldsymbol{\pi}_t = \pi_t\pi_{t+1}\pi_{t+2}\ldots$ for any $t \in \mathbb{N}$. Specifications imposed on the behavior of the system are defined as formulas composed of atomic propositions and operators. We consider the syntactically co-safe subset of linear-time temporal logic properties (Kupferman and Vardi 2001) abbreviated as scLTL. This subset of interest consists of temporal logic formulas constructed according to the following syntax

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

where $p \in \mathrm{AP}$ is an atomic proposition. The *semantics* of scLTL are defined recursively over $\boldsymbol{\pi}_t$ as $\boldsymbol{\pi}_t \models p$ iff $p \in \pi_t$; $\boldsymbol{\pi}_t \models \phi_1 \wedge \phi_2$ iff $(\boldsymbol{\pi}_t \models \phi_1) \wedge (\boldsymbol{\pi}_t \models \phi_2)$; $\boldsymbol{\pi}_t \models \phi_1 \vee \phi_2$ iff $(\boldsymbol{\pi}_t \models \phi_1) \vee (\boldsymbol{\pi}_t \models \phi_2)$; $\boldsymbol{\pi}_t \models \phi_1 \,\mathsf{U}\, \phi_2$ iff $\exists j \geq i$ subject to $(\boldsymbol{\pi}_j \models \phi_2)$ and $\boldsymbol{\pi}_k \models \phi_1, \forall k \in \{i, \ldots j-1\}$; and $\boldsymbol{\pi}_t \models \bigcirc\phi$ iff $\boldsymbol{\pi}_{i+1} \models \phi$. The eventually operator $\Diamond\phi$ is used in the sequel as a shorthand for $\mathrm{true}\,\mathsf{U}\,\phi$. We say that $\boldsymbol{\pi} \models \phi$ iff $\boldsymbol{\pi}_0 \models \phi$.

Consider a labeling function $L : \mathbb{Y} \to 2^{\mathrm{AP}}$ that assigns a letter to each output. Using this labeling map, we can define temporal logic specifications over the output of the system. Each output trace of the system $\mathbf{y} = y_0, y_1, y_2, \ldots$ can be translated to a word as $\boldsymbol{\pi} = \pi_0\pi_1\pi_2\ldots$, with $\pi_k = L(y_k)$ for $k \in \mathbb{N}$. We say that a system satisfies the specification ϕ with a probability of at least p_ϕ if $\mathbb{P}(\boldsymbol{\pi} \models \phi) \geq p_\phi$. To emphasize that the output traces of $\mathbf{M} \times \mathbf{C}$ are used for checking the satisfaction, we write $\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi)$ instead. Similarly, we denote by $\mathbb{P}\left(\mathbf{M}(\theta) \times \mathbf{C} \models \phi\right)$ the probability that the controlled system $\mathbf{M}(\theta) \times \mathbf{C}$ satisfies ϕ.

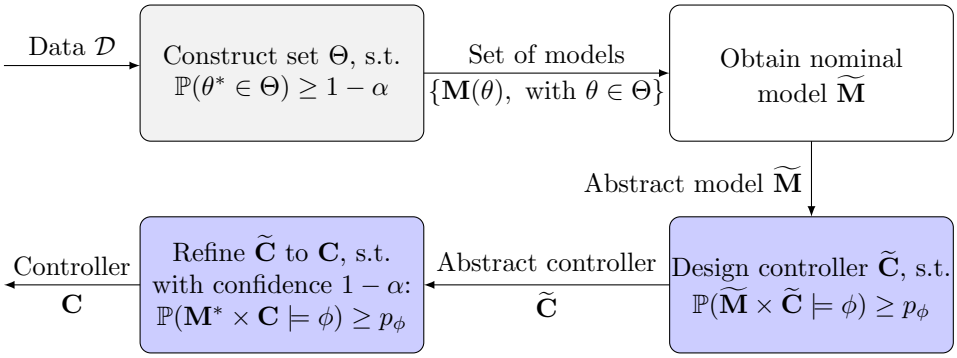


Figure 5.1: Overview of the different steps in the approach to solving Problem 5.1. Here, *s.t.* abbreviates *such that*. Subproblems 5.1a and 5.1b are respectively given in the gray and blue boxes. Note that for Problem 5.1b we follow Chapter 2 of this thesis and obtain two different steps, namely control synthesis (right blue box) and control refinement (left blue box).

5.2d Problem statement

Given model $\mathbf{M}^*$ defined as a parametrized form of (5.2), that is $\mathbf{M}^* := \mathbf{M}(\theta^*)$, where functions $f(\cdot)$ and $h(\cdot)$ are known and the fixed true parameter θ^* is unknown. Data $\mathcal{D}$ is obtained by sampling a finite number of points from $\mathbf{M}^*$:

$$\begin{aligned} \mathcal{D} &:= \left\{ (x^{(i)}, u^{(i)}, x^{+(i)}), \quad i \in \{1, \dots, N\} \right\}, \text{ where} \\ x^{+(i)} &\sim f(x^{(i)}, u^{(i)}; \theta^*) + \mathbb{P}_w(w_t), \end{aligned} \tag{5.3}$$

for arbitrary state-input pairs $(x^{(i)}, u^{(i)})$. We are interested in designing a controller $\mathbf{C}$ to satisfy temporal specifications ϕ on the output of the model, i.e., $\mathbf{M}(\theta^*) \times \mathbf{C} \models \phi$. As mentioned before in Problem 5.1, we are interested in designing a controller that ensures the satisfaction of ϕ with at least probability p_ϕ using only data $\mathcal{D}$ from the true system. In particular, we require the controller not to depend on the unknown θ^* directly. We decompose this problem into the following two sub-problems, as also visualized in Figure 5.1.

Problem 5.1a. *Construct a set Θ from data $\mathcal{D}$ that contains the true parameters θ^* with a given probability $1-\alpha \in [0,1]$, i.e.,*

$$\mathbb{P}(\theta^* \in \Theta) \geq 1-\alpha.$$

Bayesian linear regression (Bishop 2006), provides a solution to Problem 5.1a, which we detail in the next section. Based on the set Θ (Jaynes and Kempthorne 1976), we construct a parametrized set of models $\{\mathbf{M}(\theta) \text{ with } \theta \in \Theta\}$. Next, we solve Problem 5.1 by designing a controller valid for this parametrized set of models, formulated as follows.

Problem 5.1b. *For a given specification ϕ and a threshold $p_\phi \in [0,1]$, design a controller $\mathbf{C}$ independent of the parameter θ such that we obtain*

$$\mathbb{P}\left(\mathbf{M}(\theta) \times \mathbf{C} \models \phi\right) \geq p_\phi, \quad \forall \theta \in \Theta.$$

The controller synthesis for stochastic models through simulation relations is studied in Chapter 2 of this thesis. Although these simulation relations can relate one abstract model to a set of parameterized models $\mathbf{M}(\theta)$, they would lead to a control refinement that is still dependent on the true model or true parameter θ^*. Therefore, this approach is unfit to solve Problem 5.1b, since θ^* is unknown. As one of the main contributions of this chapter, we consider a parameter-independent control refinement and compute a novel simulation relation based on a sub-probability coupling (see Section 5.4) to synthesize a single controller for all $\theta \in \Theta$. We still follow the main steps in Chapter 2 of this thesis and consider a control synthesis step and control refinement step.

5.3 Data-driven parameter estimation via Bayesian linear regression

In this section, we present a solution to Problem 5.1a for the class of nonlinear systems that are linear in the unknown parameters $\theta^\top = [\theta_1, \ldots, \theta_{n_x}]^\top \in \mathbb{R}^{n_x \times n_\theta}$, i.e.,

$$\mathbf{M}(\theta) : \begin{cases} x_{t+1} & = \theta^\top f(x_t, u_t) + w_t, \\ y_t & = h(x_t), \end{cases} \tag{5.4}$$

where $x_t \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$, $y_t \in \mathbb{Y}$, $u_t \in \mathbb{U}$, and the noise $w \in \mathbb{W} \subseteq \mathbb{R}^{n_x}$ is i.i.d. Gaussian, that is $w_t \sim \mathcal{N}(\cdot|0, \Sigma)$, with zero mean and full-rank covariance matrix $\Sigma \in \mathbb{R}^{n_x \times n_x}$, and the functions $h : \mathbb{X} \to \mathbb{Y}$ and $f : \mathbb{X} \times \mathbb{U} \to \mathbb{R}^{n_\theta}$ are known. Note that $f := [f_1, \ldots, f_{n_\theta}]^\top$ can be a comprehensive library of nonlinear functions, hence, many systems can be expressed in the form of (5.4).

Based on state-input data $\mathcal{D}$ as in (5.3), *Bayesian linear regression* (e.g., Bishop (2006, Sec. 3.3)) allows us to find an *estimate* $\tilde{\theta}$ that approximates the true unknown parameters θ^* and to construct a *credible set* Θ for θ^* with a given confidence $(1-\alpha) \in [0, 1]$ (cf. Figure 5.2), defined as follows.

Definition 5.2 (Credible set)**.** *A set* Θ *is called a* credible set *of* θ^* *with confidence level* $(1-\alpha)$ *if for the given confidence* $(1-\alpha)$ *we have* $\mathbb{P}(\theta^* \in \Theta) \geq 1-\alpha$.

Remark 5.1 (Credible vs confidence sets)*.* Credible sets and *confidence sets* are two related notions stemming from two fundamentally different statistical paradigms. As a product of Bayesian statistics, a credible set of a random variable is a fixed quantile of the posterior distribution that contains a prescribed fraction of the posterior mass. A confidence set, on the other hand, is based on frequentist theory and is computed based on the frequency of random observations of a fixed true parameter. Bayesian credible sets treat their bounds as fixed and the estimated parameter as a random variable, whereas frequentist confidence sets treat their bounds as random variables and include the true fixed parameter with a certain probability. We refer the reader to Jaynes and Kempthorne (1976) for a thorough exposition.

Remark 5.2. The sub-simulation relations presented in subsequent sections are not restricted to the class of systems in (5.4) and can be used in conjunction with any other estimation method that gives a credible set Θ with $\tilde{\theta} \in \Theta$.

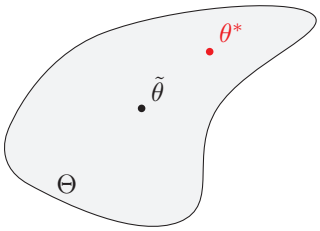


Figure 5.2: The unknown true parameter θ^* approximated with an estimate $\tilde{\theta}$ and contained within a credible set Θ (in light gray) with a confidence of $(1-\alpha)$.

5.3a Parameter estimate

To obtain an estimate of the parameters $\theta = [\theta_1, \ldots, \theta_{n_x}] \in \mathbb{R}^{n_\theta \times n_x}$, we define a prior *probability distribution* as

$$p(\theta) = \mathcal{N}(\theta|\mu_0, \Sigma_0), \tag{5.5}$$

where the mean μ_0 and covariance Σ_0 are either based on prior knowledge of the likelihood of θ, if available, or set to 0 and $\sigma_0^2 I$, respectively, with σ_0^2 a sufficiently large number. Based on the prior distributions, Bayesian linear regression allows us to compute the posterior distribution of the unknown parameters given data $\mathcal{D}$ in (5.3). For this, we define the state (transition) matrix $X^+ := [x^{+(1)}; \ldots; x^{+(N)}]$ and the *design matrix* Φ of the form

$$\Phi := \begin{bmatrix} f_1(x^{(1)}, u^{(1)}) & \ldots & f_{n_\theta}(x^{(1)}, u^{(1)}) \\ \vdots & \ddots & \vdots \\ f_1(x^{(N)}, u^{(N)}) & \ldots & f_{n_\theta}(x^{(N)}, u^{(N)}) \end{bmatrix}.$$

The posterior distribution can be computed as

$$\begin{aligned} p(\theta \mid \mathcal{D}) &= \mathcal{N}(\theta|\mu_N, \Sigma_N), \quad \text{with} \\ \mu_N &:= \Sigma_N(\Sigma_0^{-1}\mu_0 + \Sigma_i^{-1} \otimes (\Phi^\top X^+)), \qquad (5.6) \\ \Sigma_N^{-1} &:= \Sigma_0^{-1} + \Sigma_i^{-1} \otimes \Phi^\top \Phi, \qquad (5.7) \end{aligned}$$

and with $\otimes$ denoting the *Kronecker product* (Bishop 2006). Note that the parameter estimate $\tilde{\theta}$ is set to the mean, i.e., we have matrix $\tilde{\theta} = [\mu_{N,1}, \ldots, \mu_{N,n_x}]^\top \in \mathbb{R}^{n_\theta \times n_x}$ constructed by concatenating vectors $\mu_{N,i} \in \mathbb{R}^{n_\theta}$, $i \in \{1, \ldots, n_x\}$.

5.3b Credible set

For a desired confidence bound $(1-\alpha) \in [0, 1]$, we obtain the corresponding credible set

$$\Theta = \{\theta \in \mathbb{R}^{n_\theta \times n_x} \mid (\theta - \mu_N)^\top \Sigma_N^{-1} (\theta - \mu_N) \leq n_x \cdot \chi^{-1}(1 - \alpha \mid n_x)\}, \tag{5.8}$$

where $\chi^{-1}(p|\nu) := \inf\{\zeta : p \leq \chi(\zeta|v)\}$ denotes the inverse cumulative distribution function or *quantile function* of the *chi-squared* distribution

$$\chi(\zeta|v) := \int_0^\zeta \frac{t^{(v-2)/2} e^{-t/2}}{2^{v/2}\Gamma(\nu/2)} \mathrm{d}t \tag{5.9}$$

with v degrees of freedom and the *Gamma function* $\Gamma(v) := \int_0^\infty t^{v-1}e^{-t}\mathrm{d}t$ (Chew 1966). This concludes the construction of set Θ as in (5.8), such that Problem 5.1a is solved.

In the next section, we pave the way for answering Problem 5.1b for the class of scLTL specifications by an abstraction-based control design scheme. We provide a new simulation relation that enables parameter-independent control refinement from an abstract model to the original concrete model that belongs to a parameterized class.

5.4 Control refinement via sub-simulation relations

In order to find a controller $\mathbf{C}$ and answer Problem 5.1b, we employ the concept of simulation relations. In essence, simulation relations allow us to compute a controller for an abstract, e.g. nominal model $\widetilde{\mathbf{M}}$ and specification ϕ, and transfer the guarantees of satisfaction to the original model $\mathbf{M}(\theta)$ by quantifying their similarity. Using the results in Section 5.3, we construct an abstract model $\widetilde{\mathbf{M}}$, based on which we wish to design a controller:

$$\widetilde{\mathbf{M}} : \begin{cases} \tilde{x}_{t+1} & = \tilde{f}(\tilde{x}_t, \tilde{u}_t; \tilde{\theta}) + \tilde{w}_t, \quad \tilde{w}_t \sim \mathbb{P}_{\tilde{w}}(\cdot), \\ \tilde{y}_t & = \tilde{h}(\tilde{x}_t). \end{cases} \tag{5.10}$$

This abstract model equals $\mathbf{M}(\theta)$ but for a given evaluation of parameters $\theta = \tilde{\theta}$, i.e., $\widetilde{\mathbf{M}} := \mathbf{M}(\tilde{\theta})$ if $\tilde{h} \equiv h$, or any other model constructed by space reduction or discretization. Furthermore, we construct a set of models $\{\mathbf{M}(\theta) \text{ with } \theta \in \Theta\}$ using the credible set computed via the approach in Section. 5.3 or any other approach that solves Problem 5.1a.

Markov decision processes. The model $\widetilde{\mathbf{M}}$ in (5.10) and $\mathbf{M}(\theta)$ with a given fixed θ can equivalently be described by a general Markov decision process, studied previously for formal verification and synthesis of controllers (Soudjani and Abate 2013a; Haesaert et al. 2017b). Given $\tilde{x}_t, \tilde{u}_t$, we can model the stochastic state transitions of $\widetilde{\mathbf{M}}$ with a probability kernel $\tilde{\mathbf{t}}(\cdot|\tilde{x}_t, \tilde{u}_t)$ that is computed based on $\tilde{f}$ and $\mathbb{P}_{\tilde{w}}(\cdot)$ (similarly for $\mathbf{M}(\theta)$). This leads us to the representation of the systems as Markov decision processes, which is defined next.

Definition 5.3 (Markov decision process (MDP))**.** *An MDP is a tuple* $\mathbf{M} = (\mathbb{X}, x_0, \mathbb{U}, \mathbf{t})$ *with* $\mathbb{X}$ *the state space with states* $x \in \mathbb{X}$*;* $x_0 \in \mathbb{X}$ *the initial state;* $\mathbb{U}$ *the input space with inputs* $u \in \mathbb{U}$*; and* $\mathbf{t} : \mathbb{X} \times \mathbb{U} \times \mathcal{B}(\mathbb{X}) \to [0, 1]$*, a probability kernel assigning to each state* $x \in \mathbb{X}$ *and input* $u \in \mathbb{U}$ *pair a probability measure* $\mathbf{t}(\cdot|x, u)$ *on* $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$*.*

In each execution, the consecutive state $x_{t+1} \in \mathbb{X}$ is obtained as a realization $x_{t+1} \sim \mathbf{t}\,(\cdot|x_t, u_t)$ of the controlled Borel-measurable stochastic kernel. Note that for a parametrized MDP $\mathbf{M}(\theta)$ its transition kernel also depends on θ denoted as $\mathbf{t}\,(\cdot|x_t, u_t; \theta)$.

As in (5.10), we can assign an output mapping $h : \mathbb{X} \to \mathbb{Y}$ and a metric $\mathbf{d}_{\mathbb{Y}}$ to the MDP $\mathbf{M}$ to get a general MDP.

Definition 5.4 (General Markov decision process (gMDP))**.** *A gMDP is a tuple* $\mathbf{M} = (\mathbb{X}, x_0, \mathbb{U}, \mathbf{t}, h, \mathbb{Y})$ *that combines an MDP* $(\mathbb{X}, x_0, \mathbb{U}, \mathbf{t})$ *with the output space* $\mathbb{Y}$ *and a measurable output map* $h : \mathbb{X} \to \mathbb{Y}$*. A metric* $\mathbf{d}_{\mathbb{Y}}$ *decorates the output space* $\mathbb{Y}$*.*

The gMDP semantics are directly inherited from those of the MDP. Furthermore, output traces of the gMDP are obtained as mappings of (finite) MDP state executions, namely $\mathbf{y} := y_0, y_1, y_2, \ldots$ (respectively, $\mathbf{y}_{[0,N]} := y_0, y_1, y_2, \ldots, y_N$), where $y_t = h(x_t)$.

Deterministic finite-state automata. Analogous to representing systems as MDPs, the satisfaction of scLTL specifications can be checked using their alternative representation as deterministic finite-state automata, defined next.

Definition 5.5 (Deterministic finite-state automaton (DFA))**.** *A DFA is a tuple* $\mathcal{A} = (Q, q_0, \Sigma_{\mathcal{A}}, Q_f, \tau_{\mathcal{A}})$*, where* Q *is the finite set of states of the DFA,* $q_0 \in Q$ *is the initial state,* $\Sigma_{\mathcal{A}}$ *the finite alphabet,* $Q_f \subset Q$ *is the set of accepting states, and* $\tau_{\mathcal{A}} : Q \times \Sigma_{\mathcal{A}} \to Q$ *is the transition function.*

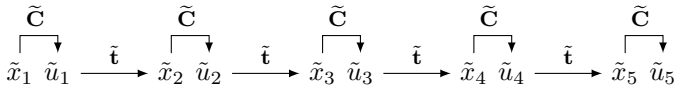
A word $\boldsymbol{\pi} = \pi_0 \pi_1 \pi_2 \ldots$ is accepted by a DFA $\mathcal{A}$ if there exists a finite run $q = q_0, q_1, \ldots, q_f$, with q_0 the initial state, $q_{i+1} = \tau_{\mathcal{A}}(q_i, \pi_i)$, and $q_f \in Q_f$.

Having defined the concepts of MDPs and DFAs, consider a set of models $\{\mathbf{M}(\theta)$ with $\theta \in \Theta\}$ and suppose that we have chosen an abstract (nominal) model $\widetilde{\mathbf{M}}$ based on which we wish to design a *single* controller and quantify the satisfaction probability over all models $\mathbf{M}(\theta)$ in the set of models.

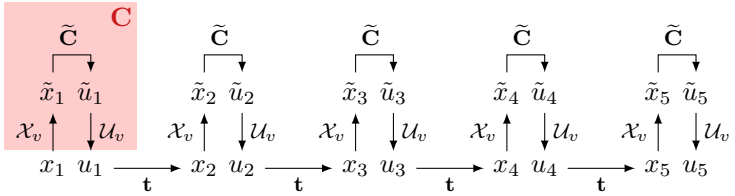
5.4a Control refinement

In this section, we first formalize the notion of a *state mapping* and an *interface function*, that together form the control refinement. Then, we investigate the conditions under which a single controller for $\widetilde{\mathbf{M}}$ can be refined to a controller for all $\mathbf{M}(\theta)$ independent of the parameter θ. Note that this is the central contribution of this work, since instead of synthesizing a parametrized controller $\mathbf{C}(\theta)$ for $\mathbf{M}(\theta)$ we synthesize a single controller $\mathbf{C}$ that works for all models $\mathbf{M}(\theta)$ in the set of models $\{\mathbf{M}(\theta)$ with $\theta \in \Theta\}$ and compute the associated lower bound on satisfying a temporal logic specification. This leads us to the novel concept of *sub-probability couplings* and simulation relations.

Consider the MDP $\widetilde{\mathbf{M}} = (\widetilde{\mathbb{X}}, \tilde{x}_0, \widetilde{\mathbb{U}}, \tilde{\mathbf{t}})$ as the abstract model, the MDP $\mathbf{M}(\theta) = (\mathbb{X}, x_0, \mathbb{U}, \mathbf{t}(\theta))$ referred to as the concrete MDP, and an abstract controller $\widetilde{\mathbf{C}}$ for $\widetilde{\mathbf{M}}$. To refine the controller $\widetilde{\mathbf{C}}$ on $\widetilde{\mathbf{M}}$ to a controller for $\mathbf{M}(\theta)$, we define a pair of interfacing functions consisting of a *state mapping* that translates the states $x \in \mathbb{X}$ to the states $\tilde{x} \in \widetilde{\mathbb{X}}$ and an *interface function* that refines the inputs $\tilde{u} \in \widetilde{\mathbb{U}}$ to the control inputs $u \in \mathbb{U}$.



(a) Controller $\widetilde{\mathbf{C}}$ on the abstract model $\widetilde{\mathbf{M}}$.



(b) Controller $\mathbf{C}$ for the concrete model $\mathbf{M}(\theta)$.

Figure 5.3: Control refinement in **(b)** that uses the abstract controller $\widetilde{\mathbf{C}}$ in **(a)** to control the concrete model $\mathbf{M}(\theta)$. Here, $\mathcal{X}_v$ and $\mathcal{U}_v$ denote respectively the state mapping and interface function.

Definition 5.6 (Interface function)**.** *An interface function (Girard and Pappas 2009; Girard and Pappas 2011) is defined as a Borel measurable stochastic kernel* $\mathcal{U}_v : \widetilde{\mathbb{X}} \times \mathbb{X} \times \widetilde{\mathbb{U}} \to \mathcal{B}(\mathbb{U})$ *such that any input* $\tilde{u}_t$ *for* $\widetilde{\mathbf{M}}$ *is refined to an input* u_t *for* $\mathbf{M}(\theta)$ *as*

$$u_t \sim \mathcal{U}_v(\cdot \mid \tilde{x}_t, x_t, \tilde{u}_t). \tag{5.11}$$

Definition 5.7 (State mapping)**.** *The state mapping is defined in a general form as a stochastic kernel* $\mathcal{X}_v$ *that maps the current state* $\tilde{x}_t$ *and input* $\tilde{u}_t$ *to the next state* $\tilde{x}_{t+1}$ *of the abstract model. This next state has the distribution specified by* $\mathcal{X}_v$ *as*

$$\tilde{x}_{t+1} \sim \mathcal{X}_v(\cdot|\tilde{x}_t, x_t, \tilde{x}_{t+1}, \tilde{u}_t).$$

The state mapping is coupled with the concrete model via its states x, x_{t+1} *and depends implicitly on* u_t *through* (5.11).

Figure 5.3a illustrates how the abstract controller $\widetilde{\mathbf{C}}$ defines a control input $\tilde{u}_t$ as a function of the abstract state $\tilde{x}_t$. Based on the state mapping $\mathcal{X}_v$ and the interface function $\mathcal{U}_v$, the abstract controller $\widetilde{\mathbf{C}}$ can be refined to a controller for $\mathbf{M}(\theta)$ as depicted in Figure 5.3b. In this figure, the states x_t from the concrete model $\mathbf{M}(\theta)$ are mapped to the abstract states $\tilde{x}_t$ with $\mathcal{X}_v$, the control inputs $\tilde{u}_t$ are obtained using $\widetilde{\mathbf{C}}$ and $\tilde{x}_t$, and these inputs are then refined to control inputs u_t for $\mathbf{M}(\theta)$ using the interface function $\mathcal{U}_v$. Hence, the controller $\mathbf{C}$ of the concrete model $\mathbf{M}(\theta)$ consists of the state mapping, the abstract controller, and the interface function.

We now question under which conditions on the interface function and the models the refinement is valid and preserves the satisfaction probability. This is addressed in the next subsection.

5.4b Valid control refinement and sub-simulation relations

Before diving into the definition of a valid control refinement that is also amenable to models with parametric uncertainty, we introduce a relaxed version of an (ϵ, δ)-stochastic simulation relation as in Definition 2.4 based on sub-probability couplings.

Similar to Chapter 2 of this thesis, we couple the two models via their inputs through an interface function (5.11) and via their states through a coupling kernel. By doing so, we can reason about the similarity between the trajectories of the two models. We define a sub-probability coupling with respect to a given relation $\mathcal{R} \subset \hat{\mathbb{X}} \times \mathbb{X}$ as follows.

Definition 5.8 (Sub-probability coupling)**.** *Given probability measures $\mathbb{P}_x \in \mathcal{P}(\mathbb{X})$, $\mathbb{P}_{\tilde{x}} \in \mathcal{P}(\widetilde{\mathbb{X}})$, relation $\mathcal{R} \subset \widetilde{\mathbb{X}} \times \mathbb{X}$, and a value $\delta \in [0, 1]$, we say that a sub-probability measure $\mathcal{W}^x_{sub}$ over $(\widetilde{\mathbb{X}} \times \mathbb{X}, \mathcal{B}(\widetilde{\mathbb{X}} \times \mathbb{X}))$ with $\mathcal{W}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X}) \geq 1 - \delta$ is a* sub-probability coupling *of $\mathbb{P}_{\tilde{x}}$ and $\mathbb{P}_x$ over $\mathcal{R}$ if*

a) $\mathcal{W}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X}) = \mathcal{W}^x_{sub}(\mathcal{R})$, that is, the probability mass of $\mathcal{W}^x_{sub}$ is located on $\mathcal{R}$,

b) for all measurable sets $\tilde{A} \subset \widetilde{\mathbb{X}}$, it holds that $\mathcal{W}^x_{sub}(\tilde{A} \times \mathbb{X}) \leq \mathbb{P}_{\tilde{x}}(\tilde{A})$; and

c) for all measurable sets $A \subset \mathbb{X}$, it holds that $\mathcal{W}^x_{sub}(\widetilde{\mathbb{X}} \times A) \leq \mathbb{P}_x(A)$.

Note that condition (a) of Definition 5.8 implies $\mathcal{W}^x_{sub}(\mathcal{R}) \geq 1 - \delta$.

Theorem 5.1 (Recovering probability coupling)**.** *For a sub-probability coupling $\mathcal{W}^x_{sub}$ as in Definition 5.8, we can complete $\mathcal{W}^x_{sub}$ to get a probability coupling $\mathcal{W}^x : \mathcal{B}(\hat{\mathbb{X}} \times \mathbb{X}) \to [0, 1]$ that couples the probability measures $\mathbb{P}_{\tilde{x}}$ and $\mathbb{P}_x$:*

$$\mathcal{W}^x(d\tilde{x} \times dx) = \tag{5.12}$$
$$\mathcal{W}^x_{sub}(d\tilde{x} \times dx) + \frac{1}{1 - \mathcal{W}^x_{sub}(\mathcal{R})} \big(\mathbb{P}(dx) - \mathcal{W}^x_{sub}(\widetilde{\mathbb{X}} \times dx)\big)\big(\mathbb{P}_{\tilde{x}}(d\tilde{x}) - \mathcal{W}^x_{sub}(d\tilde{x} \times \mathbb{X})\big).$$

In other words, (5.12) satisfies the conditions in Haesaert et al. (2017b, Definition 8).

The first term of the coupling (5.12) puts only weight on $\mathcal{R}$, while the second term assigns the remaining probability mass in an independent fashion. Note that the factor $\frac{1}{1-\mathcal{W}^x_{sub}(\mathcal{R})}$ normalizes the remaining probability mass such that $\mathcal{W}^x_{sub}$ satisfies the coupling properties $\mathcal{W}^x_{sub}(\mathrm{d}\tilde{x} \times \mathbb{X}) = \mathbb{P}_{\tilde{x}}(\mathrm{d}\tilde{x})$ and $\mathcal{W}^x_{sub}(\widetilde{\mathbb{X}} \times \mathrm{d}x) = \mathbb{P}_x(\mathrm{d}x)$ (cf. Definition 2.2 on Coupling of probability measures.)

Remark 5.3. Note that there are similarities between Definition 5.8, Definition 2.2 and Definition 8 in Haesaert et al. (2017b). Definition 2.2 defines a coupling $\mathcal{W}$ over the product space of the disturbances $(\mathbb{W} \times \mathbb{W}, \mathcal{B}(\mathbb{W} \times \mathbb{W}))$, which can be translated to a coupling $\mathcal{W}^x$ over the product space of the states $(\widetilde{\mathbb{X}} \times \mathbb{X}, \mathcal{B}(\widetilde{\mathbb{X}} \times \mathbb{X}))$. Hence, Definition 5.8 relaxes the conditions on the marginals of Definition 2.2 translated to this product state space.

Furthermore, Haesaert et al. (2017b, Definition 8) considers the associated simulation relation (defined on the product state space), and refers to this as δ-lifting of a relation. Here, they give conditions on the corresponding probability space for which Definition 5.8 gives a relaxed version.

Proof. First, we show that $\mathcal{W}^x$ is a probability measure by showing that $\mathcal{W}^x$ is fully supported on $\widetilde{\mathbb{X}} \times \mathbb{X}$, i.e., $\mathcal{W}^x(\widetilde{\mathbb{X}} \times \mathbb{X}) = 1$:

$$\mathcal{W}^x(\widetilde{\mathbb{X}} \times \mathbb{X}) = \mathcal{W}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X}) + \frac{1}{1 - \mathcal{W}^x_{sub}(\mathcal{R})}\big(1 - \mathcal{W}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X})\big)^2 = 1,$$

where we utilized the property of the sub-probability coupling as in Definition 5.8(a). The coupling properties are easily derived by calculating the marginals of $\mathcal{W}^x$, i.e.,

$$\begin{aligned}
\mathcal{W}^x(\mathrm{d}\tilde{x} \times \mathbb{X}) &= \mathcal{W}^x_{sub}(\mathrm{d}\tilde{x} \times \mathbb{X}) \\
&\quad + \frac{1}{1 - \mathcal{W}^x_{sub}(\mathcal{R})}\big(1 - \mathcal{W}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X})\big)\big(\mathbb{P}_{\tilde{x}} - \mathcal{W}^x_{sub}(\mathrm{d}\tilde{x} \times \mathbb{X})\big) \\
&= \mathbb{P}_{\tilde{x}}, \\
\mathcal{W}^x(\widetilde{\mathbb{X}} \times \mathrm{d}x) &= \mathcal{W}^x_{sub}(\widetilde{\mathbb{X}} \times \mathrm{d}x) \\
&\quad + \frac{1}{1 - \mathcal{W}^x_{sub}(\mathcal{R})}\big(\mathbb{P}_x - \mathcal{W}^x_{sub}(\widetilde{\mathbb{X}} \times \mathrm{d}x)\big)\big(1 - \mathcal{W}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X})\big) \\
&= \mathbb{P}_x.
\end{aligned}$$

Thus, the complete coupling $\mathcal{W}^x$ satisfies all conditions of Haesaert et al. (2017b, Definition 8) and of Definition 2.2 after translating it to the product space of the states. □

For the parametrized case, i.e., $\mathbb{P}_{\tilde{x}}(\cdot)$ and $\mathbb{P}_x(\cdot|\theta)$, the sub-probability coupling $\mathcal{W}^x_{sub}(\cdot|\theta)$ may likewise depend on θ. Furthermore, although we define a sub-probability coupling $\mathcal{W}^x_{sub}$ as a probability measure over the probability spaces $\mathbb{X}$ and $\widetilde{\mathbb{X}}$, we use it in its kernel form, denoted as $\bar{\mathcal{W}}^x_{sub}$ in the remainder of this chapter. We obtain a particular probability measure $\mathcal{W}^x_{sub}$ from $\bar{\mathcal{W}}^x_{sub}$ for a fixed choice of $(\tilde{x}, x, \tilde{u})$.

Let us now define (ϵ, δ)-sub-simulation relations for stochastic systems, where ϵ indicates the error in the output mapping and δ indicates the closeness in the probabilistic evolution of the two systems.

Definition 5.9 ((ϵ, δ)-sub-simulation relation (SSR))**.** *Consider two gMDPs* $\mathbf{M} = (\mathbb{X}, x_0, \mathbb{U}, \mathbf{t}, h, \mathbb{Y})$ *and* $\widetilde{\mathbf{M}} = (\widetilde{\mathbb{X}}, \tilde{x}_0, \widetilde{\mathbb{U}}, \tilde{\mathbf{t}}, \tilde{h}, \mathbb{Y})$*, a measurable relation* $\mathcal{R} \subset \widetilde{\mathbb{X}} \times \mathbb{X}$*, and an interface function* $\mathcal{U}_v : \widetilde{\mathbb{X}} \times \mathbb{X} \times \widetilde{\mathbb{U}} \to \mathbb{U}$*. If there exists a sub-probability kernel* $\bar{\mathcal{W}}^x_{sub}(\cdot|\tilde{x}, x, \tilde{u})$ *such that*

(a) $(\tilde{x}_0, x_0) \in \mathcal{R}$*;*

(b) for all $(\tilde{x}, x) \in \mathcal{R}$ *and* $\tilde{u} \in \widetilde{\mathbb{U}}$*,* $\bar{\mathcal{W}}^x_{sub}(\cdot|\tilde{x}, x, \tilde{u})$ *is a sub-probability coupling of* $\tilde{\mathbf{t}}(\cdot|\tilde{x}, \tilde{u})$ *and* $\mathbf{t}(\cdot|x, \mathcal{U}_v(\tilde{x}, x, \tilde{u}))$ *over* $\mathcal{R}$ *with respect to* δ *(see Definition 5.8);*

(c) $\forall(\tilde{x}, x) \in \mathcal{R} : \mathbf{d}_{\mathbb{Y}}(\tilde{h}(\tilde{x}), h(x)) \leq \epsilon$;

then $\widetilde{\mathbf{M}}$ *is in an* (ϵ, δ)*-SSR with* $\mathbf{M}$ *that is denoted as* $\widetilde{\mathbf{M}} \preceq_{\epsilon}^{\delta} \mathbf{M}$.

Remark 5.4. Both Definitions 5.8-5.9 are technical relaxations of the definitions of coupling (as discussed in Remark 5.3) and of (ϵ, δ)-stochastic simulation relations provided in Chapter 2 and in Haesaert et al. (2017b). These relations quantify the similarity between two models by bounding their transition probabilities with δ and their output distances with ϵ.

For a model class $\mathbf{M}(\theta)$, with $\tilde{\mathbf{t}}(\cdot|\tilde{x}, \tilde{u})$ and $\mathbf{t}(\cdot|x, \mathcal{U}_v(\tilde{x}, x, \tilde{u}); \theta)$, the above definition allows us to have an interface function $\mathcal{U}_v$ that is independent of θ but a sub-probability kernel $\bar{\mathcal{W}}_{sub}^{x}(\cdot|\tilde{x}, x, \tilde{u}; \theta)$ which may depend on θ. In order to solve Problem 5.1b, we require the state mapping $\mathcal{X}_v(\cdot|\tilde{x}_t, x_t, x_{t+1}, \tilde{u}_k)$ to also be independent of θ. This leads us to a condition under which the state mapping $\mathcal{X}_v$ gives a valid control refinement as formalized next.

Definition 5.10 (Valid control refinement). *Consider an interface function* $\mathcal{U}_v : \widetilde{\mathbb{X}} \times \mathbb{X} \times \widetilde{\mathbb{U}} \to \mathcal{B}(\mathbb{U})$ *and a sub-probability kernel* $\bar{\mathcal{W}}_{sub}^{x}$*, such that we get an SSR* $\mathcal{R}$ *according to Definition 5.9. We say that a state mapping* $\mathcal{X}_v : \widetilde{\mathbb{X}} \times \mathbb{X} \times \mathbb{X} \times \widetilde{\mathbb{U}} \to \mathcal{P}(\widetilde{\mathbb{X}})$ *defines a* valid control refinement *if the composed probability measure*

$$\begin{aligned} &\bar{\mathcal{X}}_v(d\tilde{x}_{t+1} \times dx_{t+1}|\tilde{x}_t, x_t, \tilde{u}_t) := \\ &\quad \mathcal{X}_v(d\tilde{x}_{t+1}|\tilde{x}_t, x_t, x_{t+1}, \tilde{u}_t)\mathbf{t}(dx_{t+1}|x_t, \mathcal{U}_v(\tilde{x}_t, x_t, \tilde{u}_t)) \end{aligned} \tag{5.13}$$

is lower bounded by the sub-probability coupling $\bar{\mathcal{W}}_{sub}^{x}$*, namely*

$$\bar{\mathcal{X}}_v(A|\tilde{x}_t, x_t, \tilde{u}_t) \geq \bar{\mathcal{W}}_{sub}^{x}(A|\tilde{x}_t, x_t, \tilde{u}_t), \tag{5.14}$$

for all measurable sets $A \subset \widetilde{\mathbb{X}} \times \mathbb{X}$*, all* $(\tilde{x}_t, x_t) \in \mathcal{R}$*, and all* $\tilde{u}_t \in \widetilde{\mathbb{U}}$.

Note that for a model class $\mathbf{M}(\theta)$ that is in relation with a model $\widetilde{\mathbf{M}}$, the right-hand side of (5.14) can depend on θ and the left-hand side can depend on θ only via the dynamics of $\mathbf{M}(\theta)$ represented by the kernel $\mathbf{t}$, i.e.,

$$\int_{(\tilde{x}^+, x^+) \in A} \mathcal{X}_v(\mathrm{d}\tilde{x}^+|\tilde{x}_t, x_t, x_{t+1}, \tilde{u}_t)\mathbf{t}(\mathrm{d}x^+|x_t, \mathcal{U}_v(\tilde{x}_t, x_t, \tilde{u}_t); \theta) \geq \bar{\mathcal{W}}_{sub}^{x}(A|\tilde{x}_t, x_t, \tilde{u}_t; \theta).$$

The next theorem states that there always exists at least one valid control refinement for our newly defined SSR.

Theorem 5.2. *For two gMDPs* $\widetilde{\mathbf{M}}$ *and* $\mathbf{M}$ *with* $\widetilde{\mathbf{M}} \preceq_{\epsilon}^{\delta} \mathbf{M}(\theta)$*, there always exists a valid control refinement.*

Proof. If we have SSR $\widetilde{\mathbf{M}} \preceq_{\epsilon}^{\delta} \mathbf{M}(\theta)$, then following Theorem 5.1, we can complete the corresponding sub-probability kernel $\bar{\mathcal{W}}_{sub}^{x}$. This implies that there also exists

a (complete) probabilistic kernel $\bar{\mathcal{W}}^x(\cdot|\cdot;\theta)$ satisfying the conditions for an (ϵ,δ)-stochastic simulation relation as in Definition 2.4 or in (Haesaert et al. 2017b, Def. 9). Furthermore, we can then compute the state mapping $\mathcal{X}_v$ as the conditional kernel obtained from $\bar{\mathcal{W}}^x(\cdot|\cdot;\theta)$, that is,

$$\mathcal{X}_v(\mathrm{d}\tilde{x}_{t+1}|\tilde{x}_t,x_t,x_{t+1},\tilde{u}_t;\theta) := \bar{\mathcal{W}}^x(\mathrm{d}\tilde{x}_{t+1}|\tilde{x}_t,x_t,x_{t+1},\tilde{u}_t;\theta),$$

where the right term is conditioned on x_{t+1} such that

$$\begin{aligned}&\bar{\mathcal{W}}^x(\mathrm{d}\tilde{x}_{t+1}|\tilde{x}_t,x_t,x_{t+1},\tilde{u}_t;\theta)\mathbf{t}(\mathrm{d}x_{t+1}|x_t,\mathcal{U}_v(\cdot);\theta)\\ &=\bar{\mathcal{W}}^x(\mathrm{d}\tilde{x}_{t+1}\times\mathrm{d}x_{t+1}|\tilde{x}_t,x_t,x_{t+1},\tilde{u}_t;\theta).\end{aligned}$$

This implies that the composed probability measure equals

$$\begin{aligned}&\bar{\mathcal{X}}_v(\mathrm{d}\tilde{x}_{t+1}\times\mathrm{d}x_{t+1}|\tilde{x}_t,x_t,\tilde{u}_t;\theta)\\ &=\mathcal{X}_v(\mathrm{d}\tilde{x}_{t+1}|\tilde{x}_t,x_t,x_{t+1},\tilde{u}_t;\theta)\mathbf{t}(\mathrm{d}x_{t+1}|x_t,\mathcal{U}_v(\tilde{x}_t,x_t,\tilde{u}_t);\theta)\\ &=\bar{\mathcal{W}}^x(\mathrm{d}\tilde{x}_{t+1}|\tilde{x}_t,x_t,x_{t+1},\tilde{u}_t;\theta)\mathbf{t}(\mathrm{d}x_{t+1}|x_t,\mathcal{U}_v(\tilde{x}_t,x_t,\tilde{u}_t);\theta)\\ &=\bar{\mathcal{W}}^x(\mathrm{d}\tilde{x}_{t+1}\times\mathrm{d}x_{t+1}|\tilde{x}_t,x_t,x_{t+1},\tilde{u}_t;\theta)\end{aligned}$$

and that it upper bounds the sub-probability coupling $\bar{\mathcal{W}}^x_{sub}$

$$\begin{aligned}\bar{\mathcal{X}}_v(\mathrm{d}\tilde{x}_{t+1}\times\mathrm{d}x_{t+1}|\tilde{x}_t,x_t,\tilde{u}_t;\theta) \quad &=\bar{\mathcal{W}}^x(\mathrm{d}\tilde{x}_{t+1}\times\mathrm{d}x_{t+1}|\tilde{x}_t,x_t,x_{t+1},\tilde{u}_t;\theta)\\ &\geq\bar{\mathcal{W}}^x_{sub}(\mathrm{d}\tilde{x}_{t+1}\times\mathrm{d}x_{t+1}|\tilde{x}_t,x_t,x_{t+1},\tilde{u}_t;\theta).\end{aligned}$$

Here, we used the fact that a completed kernel is always lower bounded by a sub-probability kernel, which can readily be concluded from (5.12) in Theorem 5.1. Therefore, the condition in Definition 5.10 is satisfied, and $\mathcal{X}_v$ is a valid control refinement. □

Note that in general there is more than one valid control refinement for given $\widetilde{\mathbf{M}}$ and $\mathbf{M}$ with $\widetilde{\mathbf{M}} \preceq^{\delta}_{\epsilon} \mathbf{M}(\theta)$. This allows us to choose an interface function $\mathcal{U}_v$ not dependent on θ. The above theorem states that our new framework fully recovers the results of Haesaert et al. (2017b) for non-parametric gMDPs and the results of Chapter 2 of this thesis for non-parametric stochastic difference equations.

Although Definition 5.10 gives a sufficient condition for a valid refinement, it is not a necessary condition. For specific control specifications, one can also use different refinement strategies such as the one presented in Haesaert et al. (2018), where information on the value function and relation is used to refine the control policy.

In the next theorem, we establish that our new similarity relation is transitive. This property is useful when the abstract model is constructed in multiple stages of approximating the concrete model, for example with an additional *finite-state* abstraction or *reduced model-order* abstraction as in Chapter 2 of this thesis. We exploit this property in the case study section.

Theorem 5.3. *Suppose* $\hat{\mathbf{M}} \preceq_{\epsilon_1}^{\delta_1} \widetilde{\mathbf{M}}$ *and* $\widetilde{\mathbf{M}} \preceq_{\epsilon_2}^{\delta_2} \mathbf{M}$. *Then, we have* $\hat{\mathbf{M}} \preceq_{\epsilon}^{\delta} \mathbf{M}$ *with* $\delta = \delta_1 + \delta_2$ *and* $\epsilon = \epsilon_1 + \epsilon_2$.

Proof. This proof is similar to the one given in Haesaert et al. (2017b) applied to sub-probability couplings. Consider three gMDPs $\mathbf{M} = (\mathbb{X}, x_0, \mathbb{U}, \mathbf{t}, h, \mathbb{Y})$, $\widetilde{\mathbf{M}} = (\widetilde{\mathbb{X}}, \tilde{x}_0, \widetilde{\mathbb{U}}, \tilde{\mathbf{t}}, \tilde{h}, \mathbb{Y})$, and $\hat{\mathbf{M}} = (\hat{\mathbb{X}}, \hat{x}_0, \hat{\mathbb{U}}, \hat{\mathbf{t}}, \hat{h}, \mathbb{Y})$. Given $\hat{\mathbf{M}} \preceq_{\epsilon_1}^{\delta_1} \widetilde{\mathbf{M}}$ and $\widetilde{\mathbf{M}} \preceq_{\epsilon_2}^{\delta_2} \mathbf{M}$ with $\mathcal{R}_1, \bar{\mathcal{W}}_{sub1}^x$ and $\mathcal{R}_2, \bar{\mathcal{W}}_{sub2}^x$, respectively, as formalized in Definition 5.9, define the relation $\mathcal{R} \subset \hat{\mathbb{X}} \times \mathbb{X}$ as $\mathcal{R} := \{(\hat{x}, x) \in \hat{\mathbb{X}} \times \mathbb{X} \mid \exists \tilde{x} \in \widetilde{\mathbb{X}} : (\hat{x}, \tilde{x}) \in \mathcal{R}_1, (\tilde{x}, x) \in \mathcal{R}_2\}$.

ϵ-deviation: From the definition of $\mathcal{R}$, we have that $\forall (\hat{x}, x) \in \mathcal{R}$, $\exists \tilde{x} \in \widetilde{\mathbb{X}} : (\hat{x}, \tilde{x}) \in \mathcal{R}_1, (\tilde{x}, x) \in \mathcal{R}_2$. Given $(\hat{x}, x, \tilde{x})$ and the mutual output metric $\mathbf{d}_{\mathbb{Y}}(\cdot)$, we can upper bound the output error as

$$\mathbf{d}_{\mathbb{Y}}(\hat{h}(\hat{x}), h(x)) \leq \mathbf{d}_{\mathbb{Y}}(\hat{h}(\hat{x}), \tilde{h}(\tilde{x})) + \mathbf{d}_{\mathbb{Y}}(\tilde{h}(\tilde{x}), h(x)) \leq \epsilon_1 + \epsilon_2.$$

δ-deviation: We start by completing the sub-probability couplings $\bar{\mathcal{W}}_{sub1}^x$ and $\bar{\mathcal{W}}_{sub2}^x$ to probability kernels $\bar{\mathcal{W}}_1^x$ and $\bar{\mathcal{W}}_2^x$ that couple $(\hat{\mathbf{t}}, \tilde{\mathbf{t}})$ and $(\tilde{\mathbf{t}}, \mathbf{t})$, respectively, using (5.12). Utilizing the proof in Haesaert et al. (2017b), App. D.2., we get that there exists a probability kernel $\bar{\mathcal{W}}^x$ over $(\hat{\mathbb{X}} \times \mathbb{X}, \mathcal{B}(\hat{\mathbb{X}} \times \mathbb{X}))$ that couples $\hat{\mathbf{t}}$ and $\mathbf{t}$ over $\mathcal{R}$. Furthermore, we get that for all $z \in \mathbb{Z}$, where $\mathbb{Z} := \{(\hat{x}, x, \hat{u}) \mid (\hat{x}, x) \in \mathcal{R} \text{ and } \hat{u} \in \hat{\mathbb{U}}\}$, we have $\bar{\mathcal{W}}^x(\mathcal{R} \mid z) \leq \delta_1 + \delta_2$. From (5.12) we get that for all measurable sets $\mathrm{d}\hat{x}^+ \subset \hat{\mathbb{X}}$, $\mathrm{d}x^+ \subset \mathbb{X}$ we have $\bar{\mathcal{W}}^x(\mathrm{d}\hat{x}^+ \times \mathrm{d}x^+) \geq \bar{\mathcal{W}}_{sub}^x(\mathrm{d}\hat{x}^+ \times \mathrm{d}x^+)$. Hence, we conclude that there exists an SSR $\hat{\mathbf{M}} \preceq_{\epsilon}^{\delta} \mathbf{M}$ with $\delta \leq \delta_1 + \delta_2$. □

5.4c Temporal logic control with sub-simulation relations

For designing controllers to satisfy temporal logic properties expressed as scLTL specifications, we employ the representation of the specification ϕ as a DFA $\mathcal{A}_\phi = (Q, q_0, \Sigma_{\mathcal{A}}, Q_f, \tau_{\mathcal{A}})$ (see Definition 5.5). We then use a robust version of the dynamic programming characterization of the satisfaction probability defined on the product of $\widetilde{\mathbf{M}}$ and $\mathcal{A}_\phi$ similar as in Section 2.6. We now detail the characterization that encodes the effects of both ϵ and δ for gMDPs.

Remark 5.5. In Section 2.6, we give more details and explanations on the following formulas. Even though the notation in this section differs from the one in Section 2.6, the underlying principles are exactly the same.

Denote the ϵ-neighborhood of an element $\tilde{y} \in \mathbb{Y}$ as

$$B_\epsilon(\tilde{y}) := \{y \in \mathbb{Y} | \, \mathbf{d}_{\mathbb{Y}}(y, \tilde{y}) \leq \epsilon\},$$

and a Markov policy as $\boldsymbol{\mu} : \widetilde{\mathbb{X}} \times Q \to \mathcal{P}(\widetilde{\mathbb{U}}, \mathcal{B}(\widetilde{\mathbb{U}}))$. Similar to dynamic programming with perfect model knowledge (Sutton and Barto 2018), we utilize value functions $V_k^\mu : \widetilde{\mathbb{X}} \times Q \to [0, 1]$, $k \in \{0, 1, 2, \ldots\}$ that represent the probability of starting in (x_0, q_0) and reaching the accepting set Q_f in k steps. These value functions are connected recursively via operators associated with the dynamics of the systems.

Let the initial value function be $V_0 \equiv 0$. Define the (ϵ, δ)-*robust operator* $\mathbf{T}^{\mu}_{\epsilon,\delta}$, acting on value functions as

$$\mathbf{T}^{\mu_k}_{\epsilon,\delta}(V)(\tilde{x}, q) := \mathbf{L}\Big(\int_{\widetilde{\mathbb{X}}} \min_{q^+ \in Q^+(q,\tilde{x}^+)} \max \left\{\mathbf{1}_{Q_f}(q^+), V(\tilde{x}^+, q^+)\right\} \tilde{\mathbf{t}}(\mathrm{d}\tilde{x}^+ | \tilde{x}, \mu_k) - \delta\Big), \quad (5.15)$$

where, $Q^+(q, \tilde{x}^+)$ denotes the set of possible next DFA states and $L : \mathbb{Y} \to 2^{\mathrm{AP}}$ is the labeling function. The function $\mathbf{L} : \mathbb{R} \to [0, 1]$ is the truncation function with $\mathbf{L}(\cdot) := \min(1, \max(0, \cdot))$, and $\mathbf{1}_{Q_f}$ is the indicator function of the set Q_f. The value function is now computed recursively for a policy $\boldsymbol{\mu}_i = (\mu_{i+1}, \ldots, \mu_N)$ with horizon $N - i$ as

$$V^{\boldsymbol{\mu}_{k-1}}_{N-k+1}(\tilde{x}, q) = \mathbf{T}^{\mu_k}_{\epsilon,\delta}(V^{\boldsymbol{\mu}_k}_{N-k})(\tilde{x}, q), \quad k \in \{0, 1, 2, \ldots, N\}.$$

initialized with $V_0 \equiv 0$. Furthermore, we define the *optimal* (ϵ, δ)-*robust operator* as

$$\mathbf{T}^{*}_{\epsilon,\delta}(V)(\tilde{x}, q) := \sup_{\mu} \mathbf{T}^{\mu}_{\epsilon,\delta}(V)(\tilde{x}, q).$$

The outer supremum is taken over Markov policies μ on the product space $\widetilde{\mathbb{X}} \times Q$. Based on Haesaert and Soudjani (2020), Cor. 4, we can now define the lower bound on the satisfaction probability by looking at the limit case $k \to \infty$, as given in the following proposition.

Proposition 5.1 ((ϵ, δ)-robust satisfaction probability)**.** *Suppose* $\widetilde{\mathbf{M}} \preceq^{\delta}_{\epsilon} \mathbf{M}$ *with* $\delta > 0$*, and the specification being expressed as a DFA* $\mathcal{A}_\phi$*. Then, for any such* $\mathbf{M}$ *we can construct* $\mathbf{C}$ *such that the specification is satisfied by* $\mathbf{M} \times \mathbf{C} \models \phi$ *with probability at least* $\mathbb{R}^{*}_{\epsilon,\delta}$*. This quantity is the* (ϵ, δ)-robust satisfaction probability *defined as*

$$\mathbb{R}^{*}_{\epsilon,\delta} := \max\left\{\mathbf{1}_{Q_f}(\bar{q}_0), V^{*}_{\infty}(\tilde{x}_0, \bar{q}_0)\right\},$$

with $\bar{q}_0 = \tau_{\mathcal{A}}(q_0, L(y_0))$*. Here,* V^{*}_{∞} *is the unique solution of the fixed-point equation*

$$V^{*}_{\infty} = \mathbf{T}^{*}_{\epsilon,\delta}(V^{*}_{\infty}), \quad (5.16)$$

obtained from $V^{*}_{\infty} := \lim_{k\to\infty}(\mathbf{T}^{*}_{\epsilon,\delta})^k(V_0)$ *with* $V_0 \equiv 0$*. The abstract controller* $\widetilde{\mathbf{C}}$ *is the stationary Markov policy* $\boldsymbol{\mu}^*$ *that maximizes the right-hand side of* (5.16)*, i.e.,* $\boldsymbol{\mu}^* := \arg\sup_{\mu} \mathbf{T}^{\mu}_{\epsilon,\delta}(V^{*}_{\infty})$*. The controller* $\mathbf{C}$ *is the refined controller obtained from the abstract controller* $\widetilde{\mathbf{C}}$*, the interface function* $\mathcal{U}_v$*, and the state mapping* $\mathcal{X}_v$*.*

Remark 5.6 (Bounded state space). To compute solutions for systems with unbounded support over a bounded state space $\mathbb{X}$, we add a sink state to capture transitions that leave the bounded set of states.

Combining $\mathbb{R}^{*}_{\varepsilon,\delta}$ from Proposition 5.1 with the results of Section 5.3, we get that for the dataset $\mathcal{D}$, the system under the designed controller satisfies the specification with a lower bound equal to the product of the two probabilities, that is $(1-\alpha)\mathbb{R}^{*}_{\varepsilon,\delta}$. This can be shown using the law of total probability as follows:

$$\begin{aligned}
\mathbb{P}(\mathbf{M}(\theta^*) \times \mathbf{C} \models \phi \,|\, \mathcal{D}) &= \mathbb{P}(\mathbf{M}(\theta^*) \times \mathbf{C} \models \phi \,|\, \theta^* \in \Theta, \mathcal{D})\ \mathbb{P}(\theta^* \in \Theta) \\
&\quad + \mathbb{P}(\mathbf{M}(\theta^*) \times \mathbf{C} \models \phi \,|\, \theta^* \notin \Theta, \mathcal{D})\ \mathbb{P}(\theta^* \notin \Theta) \\
&= \mathbb{R}^{*}_{\epsilon,\delta}\ (1-\alpha) + \mathbb{P}(\mathbf{M}(\theta^*) \times \mathbf{C} \models \phi \,|\, \theta^* \notin \Theta, \mathcal{D})\ \mathbb{P}(\theta^* \notin \Theta) \\
&\geq (1-\alpha)\mathbb{R}^{*}_{\epsilon,\delta}.
\end{aligned}$$

5.5 Simulation relations for nonlinear systems

In this section, we apply the previously defined concepts to construct simulation relations and show how to answer Problems 5.1a and 5.1b. Whilst we focus on nonlinear systems that are linear in the unknown parameters, the outlined approach is applicable to general nonlinear systems by choosing an appropriate parameter estimation method. Furthermore, whilst we restrict ourselves to systems with Gaussian noise here, the approach is generally applicable to other distributions as well.

Consider a nonlinear system $\mathbf{M}(\theta^*)$ as in (5.4). Let data $\mathcal{D}$ as in (5.3) be obtained from this unknown true system $\mathbf{M}(\theta^*)$. Based on $\mathcal{D}$, we approximate θ^* with an estimate $\tilde{\theta}$ and construct a credible set Θ as described in Section 5.3. For the fixed parameter estimate $\tilde{\theta}$ we construct the nominal model

$$\widetilde{\mathbf{M}} : \begin{cases} \tilde{x}_{t+1} &= \tilde{\theta}^\top f(\tilde{x}_t, \tilde{u}_t) + \tilde{w}_t, \\ \hat{y}_t &= \tilde{h}(\tilde{x}_t), \end{cases} \tag{5.17}$$

where we abbreviate $\widetilde{\mathbf{M}} := \mathbf{M}(\tilde{\theta})$. We assume that the noise $w_t, \tilde{w}_t \sim \mathcal{N}(\cdot \mid 0, \Sigma)$ has Gaussian distribution with a full rank covariance matrix $\Sigma \in \mathbb{R}^{n_x \times n_x}$ and $\tilde{h} \equiv h$.

To find a valid control refinement for the systems $\mathbf{M}(\theta)$ and $\widetilde{\mathbf{M}}$ in (5.4) and (5.17), respectively, we first write their transition kernels. The probability of transitioning from a state x_t with input u_t to a state x_{t+1} that is inside an arbitrary set S is given by

$$\mathbb{P}(x_{t+1} \in S | x_t, u_t) = \int_S \mathbf{t}(\mathrm{d}x_{t+1} | x_t, u_t),$$

where the kernel $\mathbf{t}$ is given by[1]

$$\mathbf{t}(\mathrm{d}x^+ | x, u; \theta) = \mathcal{N}(\mathrm{d}x^+ | \theta^\top f(x,u), \Sigma) = \int_{w \in \mathbb{W}} \delta_{\theta^\top f(x,u)+w}(\mathrm{d}x^+) \mathcal{N}(\mathrm{d}w | 0, \Sigma). \tag{5.18}$$

Similarly, the stochastic transition kernel of $\widetilde{\mathbf{M}}$ is

$$\tilde{\mathbf{t}}(\mathrm{d}\tilde{x}^+ | \tilde{x}, \tilde{u}) = \mathcal{N}(\mathrm{d}\tilde{x}^+ | \tilde{\theta}^\top f(\tilde{x}, \tilde{u}), \Sigma) = \int_{\tilde{w} \in \mathbb{W}} \delta_{\tilde{\theta}^\top f(\tilde{x},\tilde{u})+\tilde{w}}(\mathrm{d}\tilde{x}^+) \mathcal{N}(\mathrm{d}\tilde{w} | 0, \Sigma). \tag{5.19}$$

We can rewrite the model in (5.4) as

$$x_{t+1} = \underbrace{\tilde{\theta}^\top f(x_t, u_t)}_{\text{nominal dynamics}} + \underbrace{(\theta - \tilde{\theta})^\top f(x_t, u_t) + w_t}_{\text{disturbance}}. \tag{5.20}$$

Note that the disturbance part consists of a deviation caused by the unknown parameters θ and a deviation caused by the noise w_t. Let us assume that we can bound the former as

$$\|(\theta - \tilde{\theta})^\top f(x,u)\| \leq d(x,u), \quad \forall x \in \mathbb{X}, u \in \mathbb{U}, \theta \in \Theta.$$

[1] Here, we used the commutative property of integrating a Dirac measure as defined in (5.1).

Using the interface function $u_t = \mathcal{U}_v(\tilde{x}_t, x_t, \tilde{u}_t) := \tilde{u}_t$ and the noise coupling

$$\begin{aligned} &\tilde{w}_t \equiv \gamma(x_t, u_t, \theta; \tilde{\theta}) + w_t, \quad \text{with} \\ &\gamma(x, u, \theta; \tilde{\theta}) := (\theta - \tilde{\theta})^\top f(x, u), \end{aligned} \tag{5.21}$$

we get the state mapping

$$\tilde{x}^+ = x^+ + \tilde{\theta}^\top (f(\tilde{x}, \tilde{u}) - f(x, u)), \tag{5.22}$$

that is both deterministic and not dependent on θ, i.e., we get $\tilde{x}^+ \sim \mathcal{X}_v(\cdot|\tilde{x}, x, x^+, \tilde{u})$ as in (5.22).

Theorem 5.4. *Given data $\mathcal{D}$, confidence level $(1-\alpha)$, models $\mathbf{M}(\theta)$ and $\widetilde{\mathbf{M}}$, interface function $u_t = \tilde{u}_t$, relation (5.24), and sub-probability kernel (5.25). If $\epsilon = 0$ and*

$$\boldsymbol{\delta}(\tilde{x}, \tilde{u}) = 1 - 2\,\mathrm{cdf}\left\{-\frac{\sqrt{r}}{2}||f(\tilde{x}, \tilde{u})||\right\}, \tag{5.23}$$

with

$$r := ||\Sigma^{-1}||\ ||\Sigma_N||\ n_x\ \chi^{-1}(1-\alpha \mid n_x),$$

then the nonlinear models (5.4)-(5.17) are in an SSR $\widetilde{\mathbf{M}} \preceq_\epsilon^{\boldsymbol{\delta}} \mathbf{M}(\theta)$ and the state mapping (5.22) defines a valid control refinement.
Here, $\boldsymbol{\delta}$ is given in bold to indicate that it is a function instead of a scalar. Furthermore, $\mathrm{cdf}(\cdot)$ is the cumulative distribution function of a standard Gaussian distribution, i.e., $\mathrm{cdf}(\zeta) := \int_{-\infty}^{\zeta} \frac{1}{\sqrt{2\pi}} \exp(-\beta^2/2)\,d\beta$, covariance matrix Σ_N via (5.7), Σ is the covariance of the noise in (5.4), and χ is the chi-squared distribution as in (5.9).

Note that this theorem gives a deviation bound $\boldsymbol{\delta}$ that depends on $(\tilde{x}_t, \tilde{u}_t)$, thus providing a less conservative result compared to a constant global deviation bound δ. The proof of Theorem 5.4 is given in Appendix 5.A and is based on the following steps. First, we show that the coupling in (5.25) is indeed a sub-probability coupling by proving the conditions in Definition 5.8. Next, we prove that for the values of ϵ and $\boldsymbol{\delta}$ in Theorem 5.4, the conditions in Definition 5.9 are satisfied, hence $\widetilde{\mathbf{M}}$ is in an $(\epsilon, \boldsymbol{\delta})$-SSR with $\mathbf{M}(\theta)$. To this end, we select the identity relation

$$\mathcal{R} := \{(\tilde{x}, x) \in \widetilde{\mathbb{X}} \times \mathbb{X} \mid x = \tilde{x}\} \tag{5.24}$$

and we define the sub-probability coupling $\bar{\mathcal{W}}^x_{sub}(\cdot \mid \theta)$ over $\mathcal{R}$:

$$\begin{aligned} &\bar{\mathcal{W}}^x_{sub}(\mathrm{d}\tilde{x}^+ \times \mathrm{d}x^+ \mid \theta) = \\ &\int_{\tilde{w}\in\mathbb{W}} \int_{w\in\mathbb{W}} \left(\delta_{\tilde{\theta}^\top f(\tilde{x},\tilde{u})+\tilde{w}}(\mathrm{d}\tilde{x}^+)\delta_{\theta^\top f(x,u)+w}(\mathrm{d}x^+)\delta_{\gamma+w}(\mathrm{d}\tilde{w})\rho_{min}\right), \\ &\text{with } \rho_{min} = \min\{\mathcal{N}(\mathrm{d}w|0,\Sigma), \mathcal{N}(\mathrm{d}\tilde{w}|-\gamma,\Sigma)\}. \end{aligned} \tag{5.25}$$

This sub-probability coupling takes the minimum of two probability measures. Note that we dropped some of the arguments of $\bar{\mathcal{W}}^x_{sub}$ and γ to lighten the notation.

Finally, we prove that the state mapping in (5.22) is a valid control refinement by proving the condition in Definition 5.10.

Visualization of the sub-probability coupling: We can reduce (5.25) using the definition of γ in (5.21) to

$$\bar{\mathcal{W}}^x_{sub}(\mathrm{d}\tilde{x}^+ \times \mathrm{d}x^+ \mid \theta) = \int_{\tilde{w}\in\mathbb{W}} \delta_{\tilde{x}^+}(\mathrm{d}x^+)\delta_{\tilde{\theta}^\top f(x,u)+\tilde{w}}(\mathrm{d}\tilde{x}^+)\min\{\mathcal{N}(\mathrm{d}\tilde{w} \mid -\gamma,\Sigma),\mathcal{N}(\mathrm{d}w \mid 0,\Sigma)\},$$

which is illustrated in Figure 5.4 in the 2D plane. This coupling is constructed based on the sub-probability coupling of the two noise distributions

$$\bar{\mathcal{W}}_{sub}(\mathrm{d}\tilde{w} \times \mathrm{d}w \mid \theta) := \delta_{\gamma+w}(\mathrm{d}\tilde{w})\min\{\mathcal{N}(\mathrm{d}w|0,\Sigma),\mathcal{N}(\mathrm{d}\tilde{w}|-\gamma,\Sigma)\},$$

where γ quantifies the offset between the two distributions.

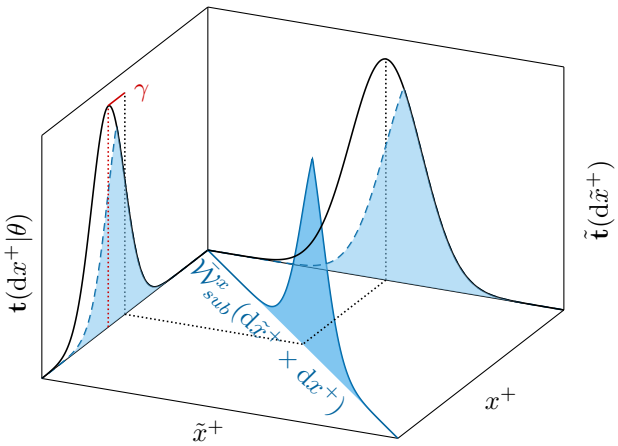
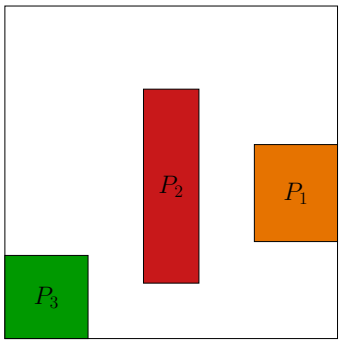


Figure 5.4: 2D representation of the sub-probability coupling $\bar{\mathcal{W}}^x_{sub}$ for the stochastic transition kernels $\tilde{\mathbf{t}}$ and $\mathbf{t}$ given some $(\tilde{x}, x, \tilde{u})$. Displayed is the coupling over the product space $\tilde{\mathbb{X}} \times \mathbb{X}$ as well as its marginals (cf. Definition 5.8 b)-c)).

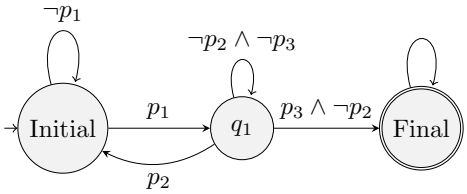
With the approach presented in this chapter, parametric uncertainty is compensated by shifting the noise distributions relative to each other by the offset γ in the sub-probability coupling. Hence, parametric uncertainty can only be compensated on state variables that are perturbed by noise.

5.6 Case studies

We demonstrate the effectiveness of our approach on a linear package delivery system subject to a complex temporal logic specification and the nonlinear discrete-time version of a Van der Pol oscillator.



(a) Regions P_1, P_2, and P_3 on the output space $\mathbb{Y}$.



(b) DFA corresponding to the specification $\phi_{PD} = \Diamond(p_1 \wedge (\neg p_2 \,\mathsf{U}\, p_3))$.

Figure 5.5: Regions and DFA corresponding to the linear package delivery case study in Section 5.6a.

5.6a Linear system with complex specification

Consider a 2D uncertain linear system in the form of (5.4) with $\theta = \left[\begin{smallmatrix} \theta_{11} & \theta_{21} & \theta_{31} & \theta_{41} \\ \theta_{12} & \theta_{22} & \theta_{32} & \theta_{42} \end{smallmatrix}\right]^\top \in \mathbb{R}^{4\times 2}$, $f(x_t, u_t) = [x_t, u_t]^\top \in \mathbb{R}^4$, $h(x_t) = x_t \in \mathbb{R}^2$, and $\Sigma = \sqrt{0.1} I_2$, describing a vehicle moving in a 2D space. Define the state space $\mathbb{X} = [-6, 6]^2$, input space $\mathbb{U} = [-1, 1]^2$, and output space $\mathbb{Y} = \mathbb{X}$. The goal is to compute a controller for a package delivery problem. For this, the controller should navigate the vehicle to pick up a package at region P_1 and deliver it to target region P_3. If the vehicle passes the region P_2 on its path, it loses the package and must pick up a new one at P_1. Here, we consider regions $P_1 = [3, 6] \times [-2.5, 1]$, $P_2 = [-1, 1] \times [-4, 3]$, and $P_3 = [-6, -3] \times [-6, -3]$ defined on the output space $\mathbb{Y}$ and labeled as respectively p_1, p_2, and p_3. The regions are shown in Figure 5.5a. The desired behavior is captured by the specification $\phi_{PD} = \Diamond(p_1 \wedge (\neg p_2 \,\mathsf{U}\, p_3))$. Note that the corresponding DFA presented in Figure 5.5b contains a backward loop from DFA state q_1 and is hence not expressible as a reach-avoid specification over the state space only.

For estimating the parameters, we sample uniform excitations u from $\mathbb{U}$ and draw N data points from the true system with parametrization $\theta^* = \left[\begin{smallmatrix} 0.6 & 0.3 & 1.2 & 0 \\ 0.2 & 0.7 & 0 & 1.4 \end{smallmatrix}\right]^\top$. We set the prior distribution to a Gaussian distribution with mean 0 and variance $10I$ for each θ_{ij} with $i \in \{1, 2, 3, 4\}$ $j \in \{1, 2\}$, that is for each column of θ we set the prior $\mathcal{N}(\cdot \mid 0_{4\times 2}, 10 I_4)$. Using Bayesian linear regression we obtain an estimate $\tilde{\theta}$ and a credible set Θ (displayed for $(\theta_{11}, \theta_{31})$ in Figure 5.6) for a lower confidence bound of 0.9 as outlined in Section 5.3. Note that the credible set is contracting for increasing amounts of data. We select $\widetilde{\mathbf{M}} = \mathbf{M}(\tilde{\theta})$ to be the nominal model and get $\epsilon_1 = 0$. δ_1 is computed using (5.23) as a function of the state and input. The values of $\max_{\tilde{u}} \delta_1$ are displayed in Figures 5.7a-5.7c for an increasing amount of data samples. Note that δ_1 grows rapidly for states away from the central region, thus a global upper bound on δ_1 would result in a poor lower bound on the satisfaction probability.

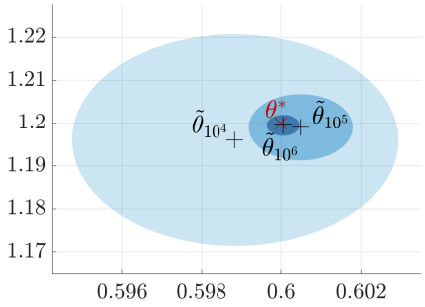


Figure 5.6: Contracting credible set Θ and parameter estimates $\tilde{\theta}_{11}, \tilde{\theta}_{31}$ for data sizes $N = 10^4, 10^5, 10^6$ (light to dark blue) with associated confidence bound $(1 - \alpha) = 0.9$.

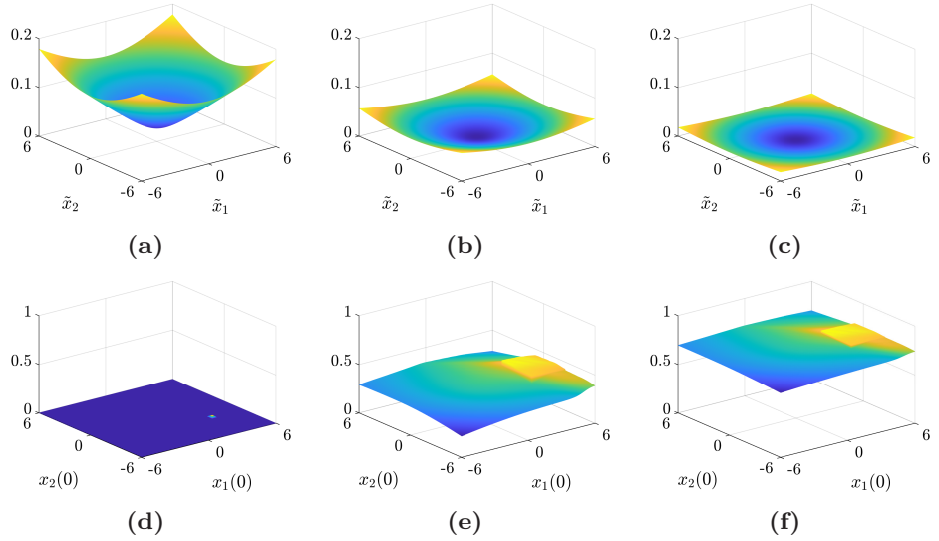


Figure 5.7: Results of the package delivery case study as described in Section 5.6a. The top figures show the maximum value of δ_1 with respect to $\tilde{u}$ as a function of the initial state $(\tilde{x}_1, \tilde{x}_2)$. Here, we used a confidence bound of $(1 - \alpha) = 0.9$ and the total number of samples equals $N = 10^4, 10^5, 10^6$ in respectively **(a)**, **(b)** and **(c)**. The bottom figures show the lower bound on the satisfaction probability with the total number of samples $N = 10^4, 10^5, 10^6$ in respectively **(d)**, **(e)**, and **(f)**. Note that the scale of the vertical axes is the same for all figures, while the colors are associated with each figure individually to indicate the shape. Here, the color scale goes from blue (lowest probability) to yellow (highest probability).

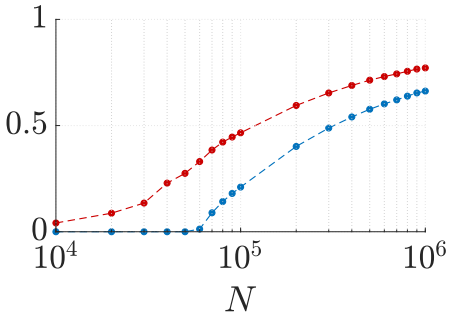


Figure 5.8: Upper bound (red) and lower bound (blue) of the robust satisfaction probability for a specific initial state as a function of the size of the dataset N.

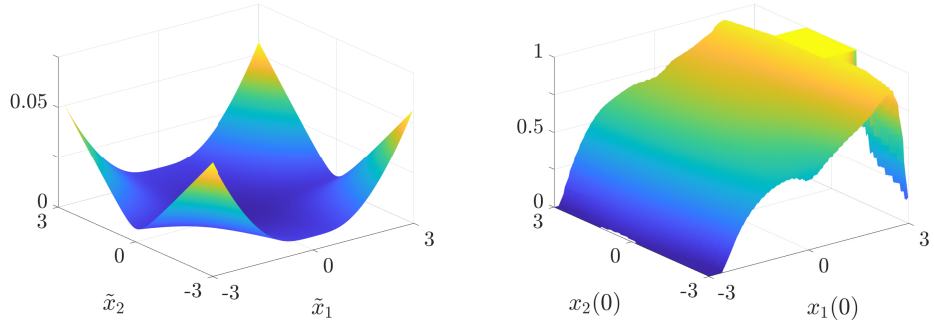
The following steps are performed using the toolbox `SySCoRe` (described in Chapter 3). We compute a second abstract model $\widehat{\mathbf{M}}$ by discretizing the state space of $\widetilde{\mathbf{M}}$ with 2500×2500 grid cells. Then, we use the results of Chapter 2 to get $\widehat{\mathbf{M}} \preceq^{\delta_2}_{\epsilon_2} \widetilde{\mathbf{M}}$ with $\epsilon_2 = 0.034$ and $\delta_2 = 0.004$. Finally, using the transitivity property in Theorem 5.3, we have $\widehat{\mathbf{M}} \preceq^{\delta}_{\epsilon} \mathbf{M}(\theta)$ with $\delta = \delta_1 + \delta_2$ and $\epsilon = \epsilon_1 + \epsilon_2$. The robust probability of satisfying the specification with this ϵ and δ is computed based on Proposition 5.1 and is depicted in Figures 5.7d-5.7f as a function of the initial state of $\mathbf{M}(\theta)$ for an increasing amount of data samples. Figure 5.8 shows the upper bound and lower bound of the robust satisfaction probability with respect to a specific initial state for increasing data set sizes. All of those figures show that the robust satisfaction probability gets higher by increasing the amount of data.

5.6b Van der Pol Oscillator

Consider a 2D uncertain nonlinear system that is a Van der Pol oscillator. Its dynamics are given in the form of (5.4) with $\theta = \left[\begin{smallmatrix} \theta_{11} & \theta_{21} & \theta_{31} & \theta_{41} \\ \theta_{12} & \theta_{22} & \theta_{32} & \theta_{42} \end{smallmatrix}\right]^\top \in \mathbb{R}^{4\times 2}$, $f(x_t, u_t) = \begin{bmatrix} x_{1,t} & x_{2,t} & x_{1,t}^2 x_{2,t} & u_t \end{bmatrix}^\top$, $h(x_t) = x_t \in \mathbb{R}^2$, and $\Sigma = \sqrt{0.2} I_2$. We consider the state space $\mathbb{X} = \mathbb{R}^2$, input space $\mathbb{U} = [-1, 1]$, and output space $\mathbb{Y} = \mathbb{X}$. We want to design a controller such that the system remains inside region P_1 while reaching target region P_2, written as $\phi_{pol} = p_1 \,\mathsf{U}\, p_2$, where p_1, and p_2 are the labels corresponding to respectively regions P_1, and P_2. The regions defined on the output space Y are $P_1 = [-3, 3]^2$ and $P_2 = [2, 3] \times [-1, 1]$.

For estimating the parameters, we sample uniform inputs u from $\mathbb{U}$ and draw $N = 10^6$ data points[2] from the true unknown system with parametrization $\theta^* = \left[\begin{smallmatrix} 1 & \tau & 0 & 0 \\ -\tau & 1+\tau a & \tau a & 0 \end{smallmatrix}\right]^\top$, where $a = 0.9$ and $\tau = 0.1$. We set the prior distribution to a Gaussian distribution with mean 0 and variance $10I$ for each θ_{ij} with $i \in \{1, 2, 3, 4\}$ $j \in \{1, 2\}$, that is for each column of θ we set the prior $\mathcal{N}(\cdot \mid 0_{4\times 2}, 10 I_4)$.

[2]When using fewer data points, we observed similar trends as for the linear system in Section 5.6a.



(a) **(b)**

Figure 5.9: Results of the van der Pol oscillator case study as described in Section 5.6b. In **(a)** we show the maximum value of $\boldsymbol{\delta}_1$ with respect to $\tilde{u}$ as a function of the initial state $(\tilde{x}_1, \tilde{x}_2)$ for a confidence bound of $(1-\alpha) = 0.9$. In **(b)** we show the lower bound on the satisfaction probability with respect to the initial state of $\mathbf{M}(\theta)$. The number of samples equals $N = 10^6$.

Remark 5.7. Note that the state dynamics of the unknown Van der Pol oscillator can equivalently be written as

$$x_{1,t+1} = x_{1,t} + x_{2,t}\tau + w_{1,t},$$
$$x_{2,t+1} = x_{2,t} + (-x_{1,t} + a(1 - x_{1,t}^2)x_{2,t})\tau + u_t + w_{2,t},$$

with parameter a and sampling time τ. In this case, we assume that the complete parametrization is unknown, hence only $f(x_t, u_t), h(x_t)$ and Σ are given.

Using Bayesian linear regression we obtain an estimate $\tilde{\theta}$ and a credible set Θ for a lower confidence bound of $(1-\alpha) = 0.9$. We select $\widetilde{\mathbf{M}} = \mathbf{M}(\tilde{\theta})$ to be the nominal model and get $\epsilon_1 = 0$. The value of $\boldsymbol{\delta}_1$ is computed using (5.23) as a function of the state and input. The maximum of $\boldsymbol{\delta}_1$ with respect to the input is displayed in Figure 5.9a. Using `SySCoRe` (see Chapter 3) we compute a second abstract model $\hat{\mathbf{M}}$ by discretizing the space of $\widetilde{\mathbf{M}}$ with 200×200 grid cells. Then, we use the method described in Chapter 2 and an extended version of `SySCoRe` to get $\hat{\mathbf{M}} \preceq^{\boldsymbol{\delta}_2}_{\epsilon_2} \widetilde{\mathbf{M}}$ with $\epsilon_2 = 0.1$ and $\boldsymbol{\delta_2}$ as a function of the abstract states of $\hat{\mathbf{M}}$. Using the transitivity property in Theorem 5.3, we have $\hat{\mathbf{M}} \preceq^{\boldsymbol{\delta}}_{\epsilon} \mathbf{M}(\theta)$ with $\boldsymbol{\delta} = \boldsymbol{\delta}_1 + \boldsymbol{\delta}_2$ and $\epsilon = \epsilon_1 + \epsilon_2$. The probability of satisfying the specification with this ϵ and $\boldsymbol{\delta}$ is computed based on Proposition 5.1 and is given in Figure 5.9b as a function of the initial state of $\mathbf{M}(\theta)$.

5.7 Discussion and conclusions

The approach presented in this chapter is the first to integrate parameter identification techniques with robust abstraction-based methods for stochastic systems. We presented a new simulation relation for stochastic systems that can establish a quantitative relation between a parameterized class of models and an abstract model. With this, it extends the applicability of previous work and allows us to synthesize robust controllers from data and provide quantified guarantees for unknown nonlinear systems with unbounded noise and complex infinite-horizon specifications. Moreover, the approach is compatible with *model order reduction* techniques and can hence be applied to higher dimensional systems.

The method described in this work is naturally extendable to *linear temporal logic* specifications over *finite-traces* (LTL_f) using the native approach outlined in Wells et al. (2020). LTL_f semantics are useful for formalizing finite-horizon planning problems. With the approach in (Wells et al. 2020), our results for SSRs can be extended to LTL_f by first translating the LTL_f formula to an equivalent first-order logic (FOL) formula (De Giacomo and Vardi 2013) and subsequently generating a minimal DFA using, e.g., the tool MONA (Henriksen et al. 1995). Thereafter, the procedure remains the same, by constructing a product MDP and solving the reachability problem (cf. Haesaert and Soudjani (2020)).

Currently, we focus on disturbances with the same dimension as the state. For future work, we plan to remove this restriction by adding a B_w-term as in Chapter 2. Besides that, in this chapter, we define a valid control refinement. However, finding a valid control refinement that does not depend on the parameter θ in a computationally efficient way is a topic for future work. Other future extensions address the limitation of the presented approach to state, input data without any measurement noise. In order to achieve formal guarantees when only noisy measurement data is available (i.e., when the observations are extended to have probability distributions conditioned on the state), relation (5.24) has to be relaxed since the state is not observed directly. The coupling in its current form has to be adapted to accommodate this.

Appendix

5.A Proof of Theorem 5.4

We first show that (5.25) is a sub-probability coupling of $\tilde{\mathbf{t}}(\cdot \mid \tilde{x}, \tilde{u})$ in (5.18) and $\mathbf{t}(\cdot \mid x, \tilde{u}; \theta)$ in (5.19) over $\mathcal{R}$ in (5.24) by proving that the conditions of Definition 5.8 are satisfied. Next, we derive $\boldsymbol{\delta}$ and prove the $(\epsilon, \boldsymbol{\delta})$-SSR between the two models using the conditions in Definition 5.9. Finally, we show that (5.22) is a valid control refinement.

To make the notation more compact, we define the tuple $z := (\tilde{x}, x, \tilde{u})$, where $z \in \mathbb{Z}$ with $\mathbb{Z} := \{(\tilde{x}, x, \tilde{u}) \mid (\tilde{x}, x) \in \mathcal{R} \text{ and } \tilde{u} \in \tilde{\mathbb{U}}\}$, and denote with ρ_{min} the sub-probability density function defined as

$$\rho_{min}(\mathrm{d}w, \mathrm{d}\tilde{w}) := \min\{\rho(\mathrm{d}w), \tilde{\rho}_\gamma(\mathrm{d}\tilde{w})\}$$

with $\rho(\mathrm{d}w)$ and $\tilde{\rho}_\gamma(\mathrm{d}\tilde{w})$ respectively the probability density functions $\mathcal{N}(\mathrm{d}w \mid 0, \Sigma)$ and $\mathcal{N}(d\tilde{w} \mid -\gamma, \Sigma)$.

Step 1: Proof of sub-probability coupling

Condition Definition 5.8 a). For all $z \in \mathbb{Z}$ and $\theta \in \Theta$ we first show that $\bar{\mathcal{W}}^x_{sub}$ is entirely located on $\mathcal{R}$, i.e., $\bar{\mathcal{W}}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X} \mid z; \theta) = \bar{\mathcal{W}}^x_{sub}(\mathcal{R} \mid z; \theta)$. We start by integrating $\bar{\mathcal{W}}^x_{sub}$ over $\widetilde{\mathbb{X}} \times \mathbb{X}$. From (5.25) we get that for all $z \in \mathbb{Z}$ and $\theta \in \Theta$ we have

$$\begin{aligned}
&\bar{\mathcal{W}}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X} \mid z; \theta) \\
&= \int_{\tilde{x}^+ \in \widetilde{\mathbb{X}}} \int_{x^+ \in \mathbb{X}} \int_{\tilde{w} \in \mathbb{W}} \int_{w \in \mathbb{W}} \delta_{\tilde{\theta}^\top f(\tilde{x}, \tilde{u}) + \tilde{w}}(\mathrm{d}\tilde{x}^+) \delta_{\theta^\top f(x,u) + w}(\mathrm{d}x^+) \delta_{\gamma + w}(\mathrm{d}\tilde{w}) \rho_{min}(\mathrm{d}w, \mathrm{d}\tilde{w}) \\
&= \int_{\tilde{w} \in \mathbb{W}} \int_{w \in \mathbb{W}} \int_{\tilde{x}^+ \in \widetilde{\mathbb{X}}} \int_{x^+ \in \mathbb{X}} \delta_{\tilde{\theta}^\top f(\tilde{x}, \tilde{u}) + \tilde{w}}(\mathrm{d}\tilde{x}^+) \delta_{\theta^\top f(x,u) + w}(\mathrm{d}x^+) \delta_{\gamma + w}(\mathrm{d}\tilde{w}) \rho_{min}(\mathrm{d}w, \mathrm{d}\tilde{w}) \\
&= \int_{w \in \mathbb{W}} \rho_{min}(\mathrm{d}w, \mathrm{d}\tilde{w}),
\end{aligned} \tag{5.26}$$

where we have changed the order of integration and used the fact that the integral of the Dirac delta measure over the whole domain is equal to one.

Similarly, we integrate (5.25) over $\mathcal{R}$ to get that for all $z \in \mathbb{Z}$ and $\theta \in \Theta$ we have

$$\bar{\mathcal{W}}^x_{sub}(\mathcal{R} \mid z; \theta) = \int_{w \in \mathbb{W}} \int_{(\tilde{x}^+, x^+) \in \mathcal{R}} \delta_{\tilde{\theta}^\top f(\tilde{x}, \tilde{u}) + w + \gamma}(\mathrm{d}\tilde{x}^+) \delta_{\theta^\top f(x,u) + w}(\mathrm{d}x^+) \rho_{min}(\mathrm{d}w, \mathrm{d}\tilde{w}).$$

The choice of $\gamma(x, u, \theta) = (\theta - \tilde{\theta})^\top f(x, u)$ ensures that $(\tilde{x}^+, x^+) \in \mathcal{R}$ for $(\tilde{x}, x) \in \mathcal{R}$ and $u = \tilde{u}$. Therefore, the integration of two Dirac delta measures over $\mathcal{R}$ becomes

one, and we get $\bar{\mathcal{W}}^x_{sub}(\mathcal{R} \mid z;\theta) = \bar{\mathcal{W}}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X} \mid z;\theta)$, thus Definition 5.8 a) is satisfied.

Condition Definition 5.8 b). For any measurable set $S \subset \tilde{\mathbb{X}}$, we integrate $\bar{\mathcal{W}}^x_{sub}$ in (5.25) over $S \times \mathbb{X}$ to get that for all $z \in \mathbb{Z}$ and $\theta \in \Theta$, $\bar{\mathcal{W}}^x_{sub}$ satisfies condition b) of Definition 5.8:

$$\begin{aligned}\bar{\mathcal{W}}^x_{sub}(S \times \mathbb{X} \mid z;\theta) &= \int_{w\in\mathbb{W}} \int_{\tilde{x}^+\in S} \delta_{\tilde{\theta}^\top f(\tilde{x},\tilde{u})+w}(d\tilde{x}^+)\rho_{min}(\mathrm{d}w,\mathrm{d}\tilde{w}) \\ &\leq \int_{w\in\mathbb{W}} \int_{\tilde{x}^+\in S} \delta_{\tilde{\theta}^\top f(\tilde{x},\tilde{u})+w}(d\tilde{x}^+)\rho(\mathrm{d}w) = \tilde{\mathbf{t}}(S|z).\end{aligned}$$

Condition Definition 5.8 c). Similarly, for any measurable set $S \subset \mathbb{X}$, we integrate $\bar{\mathcal{W}}^x_{sub}$ in (5.25) over $\widetilde{\mathbb{X}} \times S$ and get that for all $z \in \mathbb{Z}$ and $\theta \in \Theta$, $\bar{\mathcal{W}}^x_{sub}$ satisfies condition c) of Definition 5.8:

$$\bar{\mathcal{W}}^x_{sub}(\widetilde{\mathbb{X}} \times S|z;\theta) \leq \int_{w\in\mathbb{W}} \int_{x^+\in S} \delta_{\theta^\top f(\tilde{x},\tilde{u})+w}(\mathrm{d}x^+)\rho(\mathrm{d}w) = \mathbf{t}(S|z).$$

Hence, since (5.25) satisfies all conditions of Definition 5.8, it is a sub-probability coupling.

Step 2: Proof of SSR

Condition a) of Definition 5.9 holds by setting the initial states $\tilde{x}_0 = x_0$. Condition c) is satisfied with $\epsilon = 0$ since both systems use the same output mapping. For condition b), we use sub-probability coupling (5.25) and derive $\boldsymbol{\delta}$, such that the condition in Definition 5.9 is satisfied. Note that this derivation is an extension of the proof of Lemma 2.1 to stochastic systems that are influenced by non-standard Gaussian noise and with additional parametric uncertainty.

Derivation of $\boldsymbol{\delta}$ in (5.23). We show that $\boldsymbol{\delta}(\tilde{x},\tilde{u})$ in (5.23) satisfies $\bar{\mathcal{W}}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X}|z;\theta) \geq 1 - \boldsymbol{\delta}(\tilde{x},\tilde{u})$. For this, we want to find the relation between $\boldsymbol{\delta}(\tilde{x},\tilde{u})$ and $\gamma(\tilde{x},\tilde{u},\theta)$. We recognize that $\bar{\mathcal{W}}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X}|z;\theta)$ takes the minimum of two normal distributions as in (5.26). Due to the symmetric property of the normal distribution, we get

$$\bar{\mathcal{W}}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X}|z;\theta) = 2\int_E \rho(\mathrm{d}w) = 2\int_E \mathcal{N}(\mathrm{d}w|0,\Sigma), \tag{5.27}$$

with $E = \{w \,|\, \gamma^\top\Sigma^{-1}(2w-\gamma) \geq 0\}$ as illustrated in Figure 5.10. Note that the set E is a half-space obtained by simplifying the inequality $\tilde{\rho}(\mathrm{d}\tilde{w}) \geq \rho(\mathrm{d}w)$. Due to the symmetry of the variance, (5.27) can equivalently be computed as $\bar{\mathcal{W}}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X}|z;\theta) = 2\mathbb{P}(\gamma^\top\Sigma^{-1}(2w+\gamma) \leq 0)$ with $w \sim \rho(\mathrm{d}w) = \mathcal{N}(\mathrm{d}w \mid 0,\Sigma)$. This amounts to integrating a multivariate normal distribution over a half-space.

Note that this is equivalent to integrating a *one-dimensional* standard Gaussian random variable until the point $w = -\frac{1}{2}||\gamma||_{\Sigma^{-1}}$, which implies that we have the

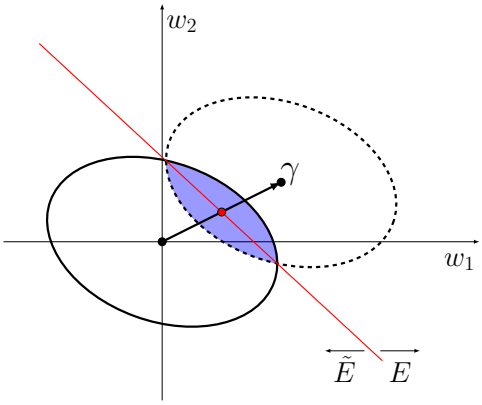


Figure 5.10: Level sets of probability density functions $\rho(\mathrm{d}w)$ (black circle) and $\tilde{\rho}_\gamma(\mathrm{d}\tilde{w})$ (dashed circle). Half spaces $\tilde{E}$ and E are respectively the $\mathbb{R}^2$-plane left and right of the red line. The area underneath $\rho_{min}(\mathrm{d}w, \mathrm{d}\tilde{w})$ *for these level sets* is indicated in blue.

following.

$$\bar{\mathcal{W}}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X}|z;\theta) = 2\,\mathrm{cdf}\left(-\frac{1}{2}||\gamma||_{\Sigma^{-1}}\right) = 2\,\mathrm{cdf}\left(-\frac{1}{2}\sqrt{\gamma^\top \Sigma^{-1}\gamma}\right).$$

Remark 5.8. For Gaussian distributions with $\Sigma = I$, we recover the results from Lemma 2.1 in Chapter 2, that is $1-\delta = \inf_{\gamma\in\Gamma} 2\,\mathrm{cdf}\,(-\frac{1}{2}||\gamma||)$, where we have equality (instead of $\leq$) because in Chapter 2 we consider a probability coupling instead of a sub-probability coupling.

Therefore, we have $\bar{\mathcal{W}}^x_{sub}(\widetilde{\mathbb{X}} \times \mathbb{X}|z;\theta) \geq 1 - \boldsymbol{\delta}(\tilde{x},\tilde{u})$, with

$$\begin{aligned} \boldsymbol{\delta}(\tilde{x},\tilde{u}) &:= 1 - 2\,\mathrm{cdf}\left(-\frac{1}{2}\sqrt{\zeta(\tilde{x},\tilde{u})}\right), \\ \zeta(\tilde{x},\tilde{u}) &:= \sup_{\theta\in\Theta}\left(\gamma^\top(\tilde{x},\tilde{u},\theta)\Sigma^{-1}\gamma(\tilde{x},\tilde{u},\theta)\right). \end{aligned} \tag{5.28}$$

Note that we calculate a $\boldsymbol{\delta}$ dependent on $(\tilde{x},\tilde{u})$ to reduce the conservatism of our approach. Given the definition of γ and the characterization of Θ in (5.8), the computation of $\zeta(\tilde{x},\tilde{u})$ is through the optimization

$$\begin{aligned} \zeta(\tilde{x},\tilde{u}) := \sup_{\theta}\ & f^\top(\tilde{x},\tilde{u})(\theta-\tilde{\theta})\Sigma^{-1}(\theta-\tilde{\theta})^\top f(\tilde{x},\tilde{u}) \\ \text{s.t.}\quad & (\Delta\theta)^\top \Sigma_N^{-1}(\Delta\theta) \leq n_x \chi^{-1}(1-\alpha|n_x), \end{aligned} \tag{5.29}$$

where $\Delta\theta := [\theta_1 - \tilde{\theta}_1; \ldots; \theta_{n_x} - \tilde{\theta}_{n_x}]$. This is a quadratically constrained quadratic program, which is convex and can be solved efficiently using interior point methods. Here, we compute an upper bound for the optimal solution, which is sufficient for

the purpose of getting a correct formal bound for $\boldsymbol{\delta}$:

$$f^\top(\tilde{x},\tilde{u})(\theta-\tilde{\theta})\Sigma^{-1}(\theta-\tilde{\theta})^\top f(\tilde{x},\tilde{u}) = \Delta\theta^\top(I_{n_x}\otimes f(\tilde{x},\tilde{u})^\top)^\top\Sigma^{-1}(I_{n_x}\otimes f(\tilde{x},\tilde{u})^\top)\Delta\theta$$
$$\leq ||I_{n_x}\otimes f(\tilde{x},\tilde{u})^\top||^2||\Sigma^{-1}||\ ||\Delta\theta||^2 = ||f(\tilde{x},\tilde{u})||^2||\Sigma^{-1}||\ ||\Delta\theta||^2$$
$$\leq ||f(\tilde{x},\tilde{u})||^2||\Sigma^{-1}||\ \frac{n_x\cdot\chi^{-1}(1-\alpha|n_x)}{\lambda_{min}(\Sigma_N^{-1})} = ||f(\tilde{x},\tilde{u})||^2||\Sigma^{-1}||\ ||\Sigma_N||n_x\chi^{-1}(1-\alpha|n_x).$$

In the above, we have used the following (in)equalities: $x^\top Ax \leq ||A||\ ||x||^2$ and $x^\top Ax \geq \lambda_{min}(A)||x||^2$ for any vector x and symmetric positive semi-definite matrix A; $||AB|| \leq ||A||\ ||B||$ and $||A\otimes B|| = ||A||\ ||B||$ for any two matrices A and B; and $||I_{n_x}|| = 1$. Therefore, we have

$$\sqrt{\zeta(\tilde{x},\tilde{u})} \leq ||f(\tilde{x},\tilde{u})||\sqrt{r},$$

with $r := ||\Sigma^{-1}||\ ||\Sigma_N||n_x\chi^{-1}(1-\alpha|n_x)$. Substituting this into (5.28) completes the proof of the expression for $\boldsymbol{\delta}$ and shows that condition b) in Definition 5.9 is satisfied.

Since there exists a sub-probability coupling $\bar{\mathcal{W}}^x_{sub}$, such that all conditions in Definition 5.9 are satisfied, we get SSR $\widetilde{\mathbf{M}} \preceq^\delta_\epsilon \mathbf{M}(\theta)$ with $\epsilon = 0$ and $\boldsymbol{\delta}(\tilde{x},\tilde{u})$ as in (5.23).

Step 3: Proof of a valid control refinement

Since we have an SSR $\widetilde{\mathbf{M}} \preceq^\delta_\epsilon \mathbf{M}(\theta)$, we can follow Theorem 5.2 to conclude that a valid control refinement exists. The state mapping in (5.22) is chosen as described in the proof of Theorem 5.2. Hence, following this proof, we can conclude that state mapping $\mathcal{X}_v$ in (5.22) satisfies the condition in Definition 5.10 and, therefore, is a valid control refinement.

Since we have proven that we get an SSR with $\epsilon = 0$ and $\boldsymbol{\delta}(\tilde{x},\tilde{u})$ as in (5.23), and that the state mapping in (5.22) is valid, the proof of Theorem 5.4 is concluded.

6

Direct data-driven control with signal temporal logic specifications

Most control synthesis methods under temporal logic properties require a model of the system, however, identifying such a model can be a challenging task. In this chapter, we develop a direct data-driven controller synthesis method for temporal logic specifications, which does not require this explicit modeling step. That is, we provide certificates for the general class of linear systems. After collecting a single sequence of input-output data from the system, we construct a data-driven characterization of the system behavior. Using this data-driven characterization we show that we can synthesize a controller, such that the controlled system satisfies a signal temporal logic specification. The underlying optimization problem is solved by mixed-integer linear programming. We demonstrate the applicability of the results through benchmark simulation examples.

6.1 Introduction

To achieve reliability of safety-critical systems, such as autonomous vehicles and power grids, it is crucial to give formal guarantees on their functionality. Hence, it is beneficial to automatically construct a provably correct controller, where the desired behavior is described by temporal logic specifications. In this work, we focus on *signal temporal logic*, since it does not only provide a qualitative answer to whether the specification is satisfied but also provides insight into the quality of satisfaction (or violation). More specifically, it is an extension of Linear Temporal Logic (LTL) with real-time and real-valued constraints.

Methods that synthesize a provably correct controller for a system subject to temporal logic specifications are often referred to as *correct-by-design control synthesis*

methods. Most of them (Wieland and Allgöwer 2007; Tabuada 2009; Belta et al. 2017) depend on the availability of an analytic model of the system. However, due to the increasing complexity of systems, obtaining an accurate model has become a challenging task. Furthermore, there is always unmodeled behavior due to first-principles-based modeling or uncertainty of the estimated model. A way to circumvent this issue is to directly synthesize a controller from data. By doing so, we avoid the possibility that some form of approximation error occurs in the modeling process. In this work, we bring down correct-by-design control synthesis to the level of data by using a data-driven method within the behavioral framework to directly synthesize a controller. We also analyze the soundness and completeness of the developed algorithm.

Literature. A common methodology in control design is to start with first-principles-based modeling combined with parameter estimation to obtain a model and use that model to design a (model-based) controller. Similar approaches are used in correct-by-design control synthesis in Haesaert et al. (2017c) and in Chapter 5 of this thesis, where one typically uses Bayesian linear regression to obtain a (stochastic) continuous-state model of the system. After obtaining a description of the continuous-state model one resorts to abstraction-based or abstraction-free methods to construct a provably correct controller. Unfortunately, this modeling step is challenging for complex systems and is prone to errors that propagate to the control design. Instead of starting with a first-principle-based continuous-state model, there also exist correct-by-design control synthesis methods (Devonport et al. 2021; Kazemi et al. 2022; Lavaei et al. 2022b) that use data to obtain a finite-state abstract model. However, these methods are prone to similar challenges and are, therefore, challenging for complex systems that are hard to model.

Recent work (Kapoor et al. 2020; Kazemi and Soudjani 2020; Wang et al. 2020; Kalagarla et al. 2021) uses reinforcement learning to learn a control policy such that a specification is satisfied. In these approaches, learning is used to obtain a system's description that is based on the state trajectory, while these methods refrain from using information about the state of the system and focus solely on input-output data.

Besides control synthesis, an interesting and somewhat related research direction is to determine signal temporal logic (STL) specifications that are satisfied by the behavior of the system (Kong et al. 2014; Aksaray et al. 2016; Bombara et al. 2016).

Recently, data-driven control has gained a lot of research interest, mainly because it shows very promising results for analysis, simulation, and control of complex systems. Such methods are generally based on the Fundamental Lemma (Willems et al. 2005), which is inferred from the behavioral framework for LTI systems (Willems and Polderman 1997) to obtain a full characterization of the system behavior, using only measurement data of the system. The developed methods for LTI systems have been used in a wide range of applications, namely data-driven simulation (Markovsky and Rapisarda 2008), performance analysis (Romer et al. 2019; Koch et al. 2021; Van Waarde et al. 2022), and control with closed-loop performance guarantees (Markovsky and Rapisarda 2007; Coulson et al. 2019; De Persis and Tesi 2019; Berberich et al. 2020b). However, these methods have never been exploited to give behavioral guarantees on the system through temporal logic specifications. In

this chapter, we develop a direct data-driven approach to automatically synthesize a controller subject to signal temporal logic specifications, using only input-output data.

This chapter is structured as follows. We start by introducing the relevant theory and defining the problem statement. In Section 6.3, we describe how to obtain a characterization of the system using data, which we use for control design subject to a temporal logic specification in Section 6.4. In Section 6.5, we analyze whether the algorithm satisfies important properties, namely soundness, and completeness. In Section 6.6, we employ our method to multiple case studies and we discuss the results and possible extensions in Section 6.7. Finally, we draw our conclusions on the proposed methodology.

6.2 Problem statement

6.2a Notation

We define the set of natural numbers by $\mathbb{N}$, the set integers by $\mathbb{Z}$, and the set of real numbers by $\mathbb{R}$. In the sequel, we denote with finite intervals $[a, b]$ the ordered sequence $\{a, a+1, \ldots, b\}$ with $a < b$ and $a, b \in \mathbb{N}$. Similarly we use $\mathbf{z}$ to denote the infinite sequence $\{z_0, z_1, z_2, \ldots\}$ and the associated infinite suffix is denoted as $\mathbf{z}_t = \{z_t, z_{t+1}, z_{t+2}, \ldots\}$. We use $\mathbf{z}_{[0,N]}$ to denote the finite sequence $\{z_0, z_1, \ldots, z_N\}$ or its stacked version[1]. The Hankel matrix of row size $L+1$ associated with a sequence $\mathbf{z}_{[0,N]}$ with elements $z_t \in \mathbb{R}^{n_z}$ for $t \in [0, N]$ is

$$\mathcal{H}_{L+1}(\mathbf{z}_{[0,N]}) = \begin{bmatrix} z_0 & z_1 & \cdots & z_{N-L} \\ z_1 & z_2 & \cdots & z_{N-L+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_L & z_{L+1} & \cdots & z_N \end{bmatrix}, \tag{6.1}$$

and has dimension $\mathcal{H}_{L+1}(\mathbf{z}_{[0,N]}) \in \mathbb{R}^{n_z(L+1) \times N-L+1}$.

6.2b Discrete-time dynamical systems

As in Willems and Polderman (1997), we define discrete-time dynamical systems based on their behavior as follows.

Definition 6.1 ((Willems and Polderman 1997, Definition 1.3.4)). *A dynamical system $\boldsymbol{\Sigma}$ is defined as a triple*

$$\boldsymbol{\Sigma} = (\mathbb{T}, \mathbb{V}, \mathfrak{B})$$

with $\mathbb{T}$ a subset of $\mathbb{Z}$, called the time axis, $\mathbb{V}$ a set called the signal space, and $\mathfrak{B}$ a subset of $\mathbb{V}^{\mathbb{T}}$ called the behavior. Here, $\mathbb{V}^{\mathbb{T}}$ is the notation for the collection of all maps from $\mathbb{T}$ to $\mathbb{V}$.

[1]Sequences can be stacked either row-wise or column-wise depending on the context.

The behavior $\mathfrak{B}$ of a system is a set of trajectories or time-dependent functions that are compatible with the system.

In this work, we define a system, its initialization, and its control design via its behavioral set. This way no exact knowledge on the structure of the system such as the dimension of the state space of the system is required.

Linear Time-Invariant Systems. In the sequel, we will focus on Linear Time-Invariant (LTI) systems (Willems and Polderman 1997, Def. 1.4.1 and 1.4.2). Dynamical systems are *linear* if the superposition principle holds, i.e., when v_1 and v_2 are trajectories that are both elements of $\mathfrak{B}$, then for the constant scalars α and β, it is implied that $\alpha v_1 + \beta v_2 \in \mathfrak{B}$. *Time-invariance* of a system implies that any time-shifted version of a trajectory in $\mathfrak{B}$ is again an element of $\mathfrak{B}$. More precisely, let q^τ denote the τ-shift, i.e., $\mathrm{q}^\tau v_t = v_{t+\tau}$ if $t + \tau \in \mathbb{T}$ (and 0 otherwise). The discrete-time dynamical system $\mathbf{\Sigma}$ is time-invariant if $\mathrm{q}^\tau \mathfrak{B} \subseteq \mathfrak{B}$ for all $\tau \in \mathbb{T}$. Note that for $\mathbb{T} = \mathbb{Z}$, we have $\mathrm{q}^\tau \mathfrak{B} = \mathfrak{B}$. This condition is called the *shift-invariance* of $\mathfrak{B}$.

A state-space representation is a realization of $\mathfrak{B}$. Consider the generic state-space representation of an LTI system given as

$$\begin{cases} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t + Du_t \end{cases} \tag{6.2}$$

with state $x_t \in \mathbb{X} = \mathbb{R}^{n_\mathrm{x}}$, input $u_t \in \mathbb{U} = \mathbb{R}^{n_\mathrm{u}}$, and output $y_t \in \mathbb{Y} = \mathbb{R}^{n_\mathrm{y}}$. Matrices A, B, C, and D are of appropriate sizes. Let the signal $\mathbf{v}$ have a signal space $\mathbb{U} \times \mathbb{Y}$ such that $\mathbf{v}$ can be partitioned into input signal $\mathbf{u}$ and output signal $\mathbf{y}$, that is, $\mathbf{v} = (\mathbf{u}, \mathbf{y})$. Then, this representation (6.2) defines a dynamical system $\mathbf{\Sigma} = (\mathbb{Z}, \mathbb{V}, \mathfrak{B})$ with signal space $\mathbb{V} = \mathbb{U} \times \mathbb{Y}$ and with behavior

$$\mathfrak{B} := \{\mathbf{v} \in \mathbb{V}^\mathbb{Z} \mid \exists \mathbf{x} \in \mathbb{X}^\mathbb{Z}, \text{ s.t. } (\mathbf{y}, \mathbf{x}, \mathbf{u}) \text{ satisfy (6.2) } \forall t \in \mathbb{Z}\}.$$

It is easy to see that $\mathbf{\Sigma}$ is linear, time-invariant, and shift invariant. Intuitively, we define the *order* of the system $\mathbf{n}(\mathfrak{B})$ as the minimal state dimension required in (6.2) such that the input-output behavior of (6.2) fully represents $\mathfrak{B}$. Similarly, we define the *lag* of the system $\mathbf{l}(\mathfrak{B})$ as the minimal length of the observations $\mathbf{y}_{[0,N]}$ required to always fully reconstruct the state. Lastly, behavior is considered *controllable* if we can steer a system to a desired trajectory in the behavior. Mathematical definitions of these concepts are given in Markovsky and Dörfler (2021, Section 2.3), and Willems and Polderman (1997, Section 1.5 and 5.2). In this chapter, we do not require exact knowledge of the structure of the system, however, the results in the chapter hold if we have rough upper bounds on the lag and the order of the system (Markovsky and Rapisarda 2008).

Finite behaviors and concatenations. We denote the behavior of an LTI system that only contains finite trajectories of length $T + 1$ as $\mathfrak{B}|_{[0,T]}$, i.e.,

$$\mathfrak{B}|_{[0,T]} := \{\mathbf{v}_{[0,T]} \mid \exists\, \mathbf{w} \in \mathfrak{B} \text{ s.t. } v_t = w_t \text{ for } 0 \leq t \leq T\}.$$

We say that $\mathbf{v}_{[0,T]}$ is a trajectory of $\mathbf{\Sigma}$ if $\mathbf{v}_{[0,T]} \in \mathfrak{B}|_{[0,T]}$.

For a given dynamical system $\mathbf{\Sigma}$, consider two input-output trajectories $\mathbf{v}^a \in \mathfrak{B}|_{[0,T_a]}$ and $\mathbf{v}^b \in \mathfrak{B}|_{[0,T_b]}$, the *concatenated* trajectory is defined as $\mathbf{v} = \mathbf{v}^a \wedge \mathbf{v}^b$ with $\mathbf{v}_{[0,T_a]} = \mathbf{v}^a$ and $\mathbf{v}_{[T_a+1,T_a+T_b+1]} = \mathbf{v}^b$. The concatenation $\mathbf{v}^a \wedge \mathbf{v}^b$ might, or might not belong to $\mathfrak{B}|_{[0,T_a+T_b+1]}$. If it does, we say that $\mathbf{v}^b$ is a (compatible) continuation of $\mathbf{v}^a$ in $\mathfrak{B}|_{[0,T_a+T_b+1]}$, or that $\mathbf{v}^a$ is a (compatible) precedent of $\mathbf{v}^b$ in $\mathfrak{B}|_{[0,T_a+T_b+1]}$. Given $\mathbf{v}^a$, we are now interested in finding trajectories $\mathbf{v}^b$ such that $\mathbf{v}^b$ is a (compatible) continuation of $\mathbf{v}^a$ in $\mathfrak{B}|_{[0,T_a+T_b+1]}$, that is $\mathbf{v}^a \wedge \mathbf{v}^b \in \mathfrak{B}|_{[0,T_a+T_b+1]}$.

Control design and initialization. Given a system $\mathbf{\Sigma}$, the design of a controller corresponds to selecting a (finite) sequence of inputs that lead to a (finite) trajectory $\mathbf{v}$ in $\mathfrak{B}$. To this end, let $L > 0$ and let $\mathbf{u}_{[0,L]}$ be a (given) control input signal. Let $\mathbf{v}^{\text{ini}} \in \mathfrak{B}|_{[0,T_{\text{ini}}]}$ with $T_{\text{ini}} > 0$ be given. If $\mathbf{v}_{[0,L]} = (\mathbf{u}_{[0,L]}, \mathbf{y}_{[0,L]})$ is a compatible continuation of $\mathbf{v}^{\text{ini}}$, then $\mathbf{y}_{[0,L]}$ is called the controlled output associated with the control input $\mathbf{u}_{[0,L]}$ and $\mathbf{v}^{\text{ini}}$. We will be interested in the situation that T_{ini} is such that for all initial trajectories $\mathbf{v}^{\text{ini}} \in \mathfrak{B}|_{[0,T_{\text{ini}}]}$, the controlled output $\mathbf{y}_{[0,L]}$ associated with $\mathbf{u}_{[0,L]}$ and $\mathbf{v}^{\text{ini}}$ is uniquely defined. The existence of such a T_{ini} is guaranteed by the following proposition.

Proposition 6.1 (Initial Condition (Markovsky and Rapisarda 2008; Markovsky and Dörfler 2021))**.** *Let $T_{ini} + 1 \geq \mathbf{l}(\mathfrak{B})$. Then for any given $\mathbf{v}^{ini} \in \mathfrak{B}|_{[0,T_{ini}]}$ and $\mathbf{u}_{[0,L]} \in (\mathbb{R}^{n_u})^{L+1}$, there is a unique $\mathbf{y}_{[0,L]} \in (\mathbb{R}^{n_y})^{L+1}$, such that $\mathbf{v}^{ini} \wedge (\mathbf{u}_{[0,L]}, \mathbf{y}_{[0,L]}) \in \mathfrak{B}|_{[0,T_{ini}+L]}$.*

This means that $\mathbf{v}^{\text{ini}}$ is the collection of past inputs and outputs that is sufficient to uniquely determine the compatible continuations of $\mathbf{v}^{\text{ini}}$, once the control input $\mathbf{u}_{[0,L]}$ is known. We, therefore, say that $\mathbf{v}^{\text{ini}}$ is an *initialization* or initial condition.

6.2c Signal temporal logic specifications.

Consider the language of *signal temporal logic* (STL) (Maler and Nickovic 2004; Deshmukh et al. 2017) whose syntax is recursively defined as

$$\varphi ::= \sigma \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Box_{[a,b]}\varphi \mid \varphi_1 \mathsf{U}_{[a,b]}\, \varphi_2,$$

starting from predicate $\sigma : \sigma(y) > 0$ for predicate function $\sigma : \mathbb{Y} \to \mathbb{R}$. The semantics are given below.

$$\begin{aligned}
\mathbf{y}_t \models \sigma &\Leftrightarrow \quad \sigma(y_t) > 0 \\
\mathbf{y}_t \models \neg\varphi &\Leftrightarrow \quad \neg(\mathbf{y}_t \models \varphi) \\
\mathbf{y}_t \models \varphi_1 \wedge \varphi_2 &\Leftrightarrow \quad \mathbf{y}_t \models \varphi_1 \wedge \mathbf{y}_t \models \varphi_2 \\
\mathbf{y}_t \models \varphi_1 \vee \varphi_2 &\Leftrightarrow \quad \mathbf{y}_t \models \varphi_1 \vee \mathbf{y}_t \models \varphi_2 \\
\mathbf{y}_t \models \Box_{[a,b]}\varphi &\Leftrightarrow \quad \forall t' \in [t+a, t+b] : \ \mathbf{y}_{t'} \models \varphi \\
\mathbf{y}_t \models \varphi_1 \mathsf{U}_{[a,b]}\, \varphi_2 &\Leftrightarrow \quad \exists t' \in [t+a, t+b] \ \text{ such that } \mathbf{y}_{t'} \models \varphi_2 \\
&\qquad\qquad \wedge \forall t'' \in [t, t'] : \ \mathbf{y}_{t''} \models \varphi_1.
\end{aligned}$$

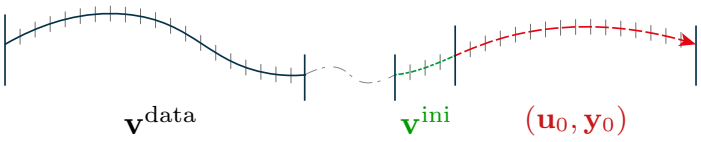


Figure 6.1: Illustration of the different trajectories used in this approach. Here, we see a trajectory of a system $\boldsymbol{\Sigma}$ build up by concatenating multiple trajectories. The trajectory used for data and initialization are given in black and green respectively. The designed trajectory is shown in red.

Additionally, we define the *eventually*-operator as $\Diamond_{[a,b]}\varphi := \mathit{true}\ \mathsf{U}_{[a,b]}\ \varphi$, which is true if φ holds at some time on the interval between a and b. Similarly, an *implication* is defined as $\varphi_1 \implies \varphi_2 := \neg\varphi_1 \vee \varphi_2$, which is true if φ_1 implies that φ_2 is true. A (possibly infinite) trajectory $\mathbf{y} = y_0 y_1 y_2 \dots$ satisfies φ denoted by $\mathbf{y} \models \varphi$ iff $\mathbf{y}_0 \models \varphi$ holds.

Furthermore, for a given input sequence $\mathbf{u}_0 = \{u_0, u_1, u_2, \dots\}$, and initialization $\mathbf{v}^{\text{ini}}$, we say that $\boldsymbol{\Sigma}$ satisfies specification φ, if the unique continuation $\mathbf{y}_0$ is such that $\mathbf{y}_0 \models \varphi$.

Remark 6.1. As is common for optimization-based approaches that synthesize controllers that provably satisfy temporal logic requirements, we use STL to describe the specifications. However, we should point out that exactly the same results can be obtained for bounded versions of LTL (Belta and Sadraddini 2019).

6.2d Cost function

To optimize the performance of the controller, we introduce a cost function as a performance measure. Since we use only input-output data of the system, the cost function can only be a function of u and y. More specifically, we define a cost function $J : \mathbb{U} \times \mathbb{Y} \to \mathbb{R}$ that assigns a cost to a trajectory $(\mathbf{u}_0, \mathbf{y}_0)$. In this work, we consider cost functions that are quadratic and of the form

$$J(\mathbf{u}_0, \mathbf{y}_0) = ||\mathbf{u}_0||_R^2 + ||\mathbf{y}_0||_Q^2,$$

with $||\boldsymbol{x}_{[0,L]}||_P$ denoting the weighted-norm of a trajectory defined as $\sqrt{\boldsymbol{x}^\top P \boldsymbol{x}} = \sum_{t=0}^{L} \sqrt{x_t^\top P x_t}$.

6.2e Problem statement

We are now interested in data-driven control that achieves STL specifications, while also optimizing performance. Data-driven control ranges from controllers that are designed from data-driven models to controllers synthesized directly from (a collection of) data. In this work, we are interested in the latter – *direct data-driven control*. More precisely, we will focus on using a single sequence of data to directly design a sound data-driven controller. Thus, given a finite input-output sequence

$\mathbf{v}^{\text{data}}$ from an unknown LTI system $\mathbf{\Sigma} = (\mathbb{Z}, \mathbb{U} \times \mathbb{Y}, \mathfrak{B})$, $\mathbf{v}^{\text{data}} \in \mathfrak{B}|_{[0,T]}$, we want to synthesize a control input $\mathbf{u}_0$, such that the trajectory initialized with $\mathbf{v}^{\text{ini}}$ has an optimal continuation for which $\mathbf{y}_0 \models \varphi$. Formally this can be written as.

Problem 6.1 (Direct data-driven STL control)**.** *Given a finite data sequence* $\mathbf{v}^{data} \in \mathfrak{B}|_{[0,T]}$, $T_{ini} + 1 \geq \mathbf{l}(\mathfrak{B})$, *initial condition* $\mathbf{v}^{ini} \in \mathfrak{B}|_{[0,T_{ini}]}$, *STL specification* φ, *and cost function* J, *solve*

$$\arg\min_{\mathbf{u}_0} \; J(\mathbf{u}_0, \mathbf{y}_0) \tag{6.3a}$$

$$\textit{subject to} \;\; \mathbf{v}^{ini} \wedge (\mathbf{u}_0, \mathbf{y}_0) \in \mathfrak{B}, \tag{6.3b}$$

$$\textit{and} \;\; \mathbf{y}_0 \models \varphi. \tag{6.3c}$$

Approach. In this chapter, we develop a direct data-driven control synthesis method by performing the following steps. First, we obtain input-output data of the system collected in $\mathbf{v}^{\text{data}}$. As long as these trajectories contain enough information, this allows us to obtain a data-driven characterization of the system. Next, we rewrite the STL specification into a set of mixed-integer linear programming constraints. After obtaining an initial trajectory that initializes the system and is not part of $\mathbf{v}^{\text{data}}$, we solve an optimization problem that synthesizes a controller. This controller yields the input trajectory $\mathbf{u}_{[0,L]}$ such that $(\mathbf{u}_{[0,L]}, \mathbf{y}_{[0,L]})$ satisfies the STL specifications. In this chapter, we focus on finite trajectories with horizon $L+1$, hence we have $(\mathbf{u}_{[0,L]}, \mathbf{y}_{[0,L]})$ instead of $(\mathbf{u}_0, \mathbf{y}_0)$. More specifically, for now we compute the entire input sequence at the beginning of the interval and then apply it fully without any re-computation. The different trajectories used in this approach are illustrated in Figure 6.1.

6.3 Data-driven characterization of the system

In order to obtain a characterization of the system using input-output data, we use the Fundamental Lemma from Willems et al. (2005). Suppose that we are given $\mathbf{v}^{\text{data}} \in \mathfrak{B}|_{[0,T]}$, of an LTI system, then based on the linearity and shift-invariance of the LTI system, we know that for any $\alpha \in \mathbb{R}^{T-L+1}$ it holds that

$$\mathcal{H}_{L+1}(\mathbf{v}^{\text{data}})\alpha \in \mathfrak{B}|_{[0,L]}.$$

Under some additional conditions, we also have that for every $\mathbf{v}_{[0,L]} \in \mathfrak{B}|_{[0,L]}$ there exists an $\alpha \in \mathbb{R}^{T-L+1}$ such that

$$\mathcal{H}_{L+1}(\mathbf{v}^{\text{data}})\alpha = \mathbf{v}_{[0,L]}. \tag{6.4}$$

We will formalize these statements and the required conditions in the remainder of this section. Furthermore, we will show that these properties hold true.

Let us first introduce the persistence of excitation condition.

Definition 6.2. *A finite sequence* $\mathbf{u}^{data} \in \mathbb{U}^{T+1}$ *is persistently exciting of order* $L+1$ *if and only if*

$$\operatorname{rank}(\mathcal{H}_{L+1}(\mathbf{u}^{data})) = (L+1)n_{\mathrm{u}},$$

with the Hankel matrix $\mathcal{H}_{L+1}(\cdot)$ *as in* (6.1).

The Fundamental Lemma (Willems et al. 2005, Thm 1) gives the conditions for equality (6.4) and shows that once the input is persistently exciting of order $L+1+n_{\mathrm{x}}$, the Hankel matrix of depth $L+1$ constructed from the data, spans the full behavior of the LTI system, restricted to length $L+1$ trajectories. Translating this to the classical control setting, we obtain the result from Berberich and Allgöwer (2020).

Proposition 6.2 (Data-driven control (Berberich and Allgöwer 2020))**.** *Suppose* $\mathbf{v}^{data} = (\mathbf{u}^{data}, \mathbf{y}^{data})$ *is a trajectory of an LTI system* $\boldsymbol{\Sigma}$*, where* $\mathbf{u}^{data}$ *is persistently exciting of order* $L+1+n_{\mathrm{x}}$*. Then,* $\mathbf{v}_{[0,L]} = (\mathbf{u}_{[0,L]}, \mathbf{y}_{[0,L]})$ *is a trajectory of* $\boldsymbol{\Sigma}$ *if and only if there exists* $\alpha \in \mathbb{R}^{T-L+1}$ *such that*

$$\begin{bmatrix} \mathcal{H}_{L+1}(\mathbf{u}^{data}) \\ \mathcal{H}_{L+1}(\mathbf{y}^{data}) \end{bmatrix} \alpha = \begin{bmatrix} \mathbf{u}_{[0,L]} \\ \mathbf{y}_{[0,L]} \end{bmatrix}. \tag{6.5}$$

The main interpretation is that any length $L+1$ trajectory is spanned by the time-shifts of the measured trajectories in the data-dictionary $\mathbf{v}^{\mathrm{data}}$. Hence, the left-hand side of (6.5) can be interpreted as a fully data-based representation of the LTI system. This implies that with sufficiently informative data, you can characterize the full system behavior consisting of all length $L+1$ system trajectories.

6.4 Direct data-driven temporal logic control synthesis

Now that we know how to obtain a characterization of the system as in (6.5), we can rewrite the constraints of optimization problem (6.3) into a mixed-integer linear program.

In optimization problem (6.3) we have two constraints. The first constraint on *the system dynamics* implies that the trajectory belongs to the behavior of the model. This can be equivalently written using the characterization of the system obtained through data, i.e., as in (6.5). The second *STL constraint* in (6.3) guarantees the satisfaction of the STL specification. This can equivalently be rewritten as a mixed-integer linear program (MILP) using the method described in Raman et al. (2014). Both steps are well known and are summarized here for completeness. Together these steps define a new direct data-driven algorithm for STL specifications whose soundness is analyzed in Section 6.5.

Compute minimal L for φ. We start by computing the required trajectory length $L+1$. For a finite trajectory $\mathbf{y}_{[0,L]}$ to satisfy an STL specification φ, that is $\mathbf{y}_0 \models \varphi$, it has to be sufficiently long. For example to satisfy the specification $\varphi = \Box_{[a,b]}\Diamond_{[c,d]}(y_t \geq 1)$, the trajectory $\mathbf{y}_{[0,L]}$ should have a horizon larger than $b+d$, that is $L+1 > b+d$. The necessary horizon $L+1$ can be computed based on the structure of the formula φ, cf. Maler and Nickovic (2004). Denote the required length associated to formula φ as $\|\varphi\|$, then we have the following relations:

- $\|\sigma\| = 0$,
- $\|\neg\varphi\| = \|\varphi\|$,
- $\|\varphi_1 \wedge \varphi_2\| = \max(\|\varphi_1\|, \|\varphi_2\|)$,
- $\|\varphi_1 \vee \varphi_2\| = \max(\|\varphi_1\|, \|\varphi_2\|)$,
- $\|\Box_{[a,b]}\varphi\| = \|\varphi\| + b$,
- $\|\varphi_1 \,\mathsf{U}_{[a,b]}\, \varphi_2\| = \max(\|\varphi_1\|, \|\varphi_2\|) + b$.

Hence, to achieve $\mathbf{y}_{[0,L]} \models \varphi$, we need horizon

$$L + 1 > \|\varphi\|. \tag{6.6}$$

STL satisfaction as MILP constraints. Following Raman et al. (2014), we can rewrite the satisfaction of an STL specification as MILP constraints. To this end, introduce the binary auxiliary variable $\zeta_t^\varphi \in \{0,1\}$, with $t \in [0,1,\ldots,L]$, associated to a set of MILP constraints such that $\zeta_t^\varphi = 1$ if φ holds at time t . As before, the satisfaction of a formula is considered at $t = 0$, hence if $\zeta_0^\varphi = 1$, the formula is satisfied. Variable ζ_0^φ is computed recursively using the associated MILP constraints.

Associate to each predicate σ its own auxiliary binary variable $\zeta_t^\sigma \in \{0,1\}$ for $t \in [0,1,\ldots,L]$. The following constraints

$$\sigma(y_t) \leq M_t \zeta_t^\sigma - \epsilon_t \tag{6.7}$$
$$-\sigma(y_t) \leq M_t(1 - \zeta_t^\sigma) - \epsilon_t, \tag{6.8}$$

make sure that $\zeta_t^\sigma = 1$ if and only if $\sigma(y_t) > 0$. Here, $M_t > 0$ are sufficiently large numbers and $\epsilon_t > 0$ are sufficiently small numbers, included to avoid numerical issues.

For each Boolean operator, that is $\varphi = \neg\varphi, \varphi = \wedge_{i=1}^m \varphi_i$ and $\varphi = \vee_{i=1}^m \varphi_i$, we add an auxiliary variable ζ_t^φ. We define the encodings of these Boolean operators on binary variables ζ_t^φ as in Raman et al. (2014) and Wolff et al. (2014) (repeated here for completeness).

$$\begin{array}{lcl}
\zeta_t^\varphi = \neg\zeta_t^\varphi & \Leftrightarrow & \zeta_t^\varphi = 1 - \zeta_t^\varphi, \\
\zeta_t^\varphi = \wedge_{i=1}^m \zeta_{t_i}^{\varphi_i} & \Leftrightarrow & \zeta_t^\varphi \leq \zeta_{t_i}^{\varphi_i}, \text{ with } i = 1,2,\ldots,m \\
 & & \zeta_t^\varphi \geq 1 - m + \sum_{i=1}^m \zeta_{t_i}^{\varphi_i}, \\
\zeta_t^\varphi = \vee_{i=1}^m \zeta_{t_i}^{\varphi_i} & \Leftrightarrow & \zeta_t^\varphi \geq \zeta_{t_i}^{\varphi_i}, \text{ with } i = 1,2,\ldots,m \\
 & & \zeta_t^\varphi \leq \sum_{i=1}^m \zeta_{t_i}^{\varphi_i},
\end{array} \tag{6.9}$$

These constraints make sure that $\zeta_t^\varphi = 1$ only if φ holds at time t.

Next, we define the temporal operators $\varphi = \Box_{[a,b]}\psi$ and $\varphi = \Diamond_{[a,b]}\psi$ as in Raman et al. (2014). To this end, we add auxiliary variables ζ_t^φ and define their encodings as follows:

$$\begin{array}{lcl}
\zeta_t^\varphi = \Box_{[a,b]}\zeta_t^\psi & \Leftrightarrow & \zeta_t^\varphi = \wedge_{i=\min(t+a,L)}^{\min(t+b,L)} \zeta_i^\psi, \\
\zeta_t^\varphi = \Diamond_{[a,b]}\zeta_t^\psi & \Leftrightarrow & \zeta_t^\varphi = \vee_{i=\min(t+a,L)}^{\min(t+b,L)} \zeta_i^\psi,
\end{array} \tag{6.10}$$

with the conjunction and disjunction operators defined in (6.9). Finally, the bounded until operator can be written as a logical combination of the bounded always, bounded eventually, and unbounded until operator U.

$$\varphi_1 \,\mathsf{U}_{[a,b]}\, \varphi_2 = \Box_{[0,a]}\varphi_1 \wedge \Diamond_{[a,b]}\varphi_2 \wedge \Diamond_{[a,a]}(\varphi_1 \,\mathsf{U}\, \varphi_2). \tag{6.11}$$

Hence, we define the encoding over auxiliary variables $\zeta_t^{\varphi} \in [0,1]$ as

$$\zeta_t^{\varphi} = \zeta_t^{\varphi_1 \mathsf{U}_{[a,b]} \varphi_2} = \zeta_t^{\Box_{[0,a]}\varphi_1} \wedge \zeta_t^{\Diamond_{[a,b]}\varphi_2} \wedge \zeta_t^{\Diamond_{[a,a]}(\varphi_1 \mathsf{U} \varphi_2)}, \tag{6.12}$$

with

$$\zeta_t^{\varphi_1 \mathsf{U} \varphi_2} = \zeta_t^{\varphi_2} \vee (\zeta_t^{\varphi_1} \wedge \zeta_{t+1}^{\varphi_1 \mathsf{U} \varphi_2}), \tag{6.13}$$

for $t = 1, 2, \ldots, L$ and $\zeta_L^{\varphi_1 \mathsf{U} \varphi_2} = \zeta_L^{\varphi_2}$.

Direct data-driven STL control synthesis. In order to design a controller, we need an initial trajectory that satisfies Proposition 6.1. Hence, we have a given initializing trajectory $\mathbf{v}^{\text{ini}} = (\mathbf{u}^{\text{ini}}, \mathbf{y}^{\text{ini}})$ of length $T_{\text{ini}}+1$, with $T_{\text{ini}} \geq \mathbf{l}(\mathfrak{B})$, such that $\mathbf{y}_{[0,L]}$ is uniquely defined by $\mathbf{u}_{[0,L]}$. Furthermore, we have data $\mathbf{v}^{\text{data}} = (\mathbf{u}^{\text{data}}, \mathbf{y}^{\text{data}})$, with $\mathbf{u}^{\text{data}}$ sufficiently rich according to Proposition 6.2. As in (6.3b), the total trajectory, $\mathbf{v}^{\text{ini}} \wedge (\mathbf{u}_{[0,L]}, \mathbf{y}_{[0,L]})$, consists of an initialization part $\mathbf{v}^{\text{ini}}$ and a controlled part $\mathbf{v}_{[0,L]}$, which extends the data-driven characterization as in (6.5). Denote the set of MILP constraints associated with a given STL formula φ as $\boldsymbol{\zeta}_t^{\varphi} = 1$ with $t \in [0, 1, \ldots, L]$. The optimization problem (6.3) can now be written as

$$\underset{\mathbf{u}_{[0,L]} \in \mathbb{U}^{T+1}, \boldsymbol{\zeta}_t^{\varphi}}{\arg\min} \; J(\mathbf{u}_{[0,L]}, \mathbf{y}_{[0,L]}) \tag{6.14a}$$

$$\text{s.t.} \begin{bmatrix} \mathcal{H}_{T_{\text{ini}}+L+1}(\mathbf{u}^{\text{data}}) \\ \mathcal{H}_{T_{\text{ini}}+L+1}(\mathbf{y}^{\text{data}}) \end{bmatrix} \alpha = \begin{bmatrix} \mathbf{u}^{\text{ini}} \\ \mathbf{u}_{[0,L]} \\ \mathbf{y}^{\text{ini}} \\ \mathbf{y}_{[0,L]} \end{bmatrix} \tag{6.14b}$$

$$\boldsymbol{\zeta}_t^{\varphi} = 1 \text{ constructed with (6.7) – (6.13)}, \tag{6.14c}$$

$$\zeta_0^{\varphi} = 1, \tag{6.14d}$$

which is solvable using an MILP algorithm, e.g. with `Gurobi` (Gurobi Optimization, LLC 2023).

The complete direct data-driven control synthesis procedure is summarized in Algorithm 6.1.

Algorithm 6.1 Direct data-control synthesis procedure

1: **Input:** $\mathbf{v}^{\text{data}}, \mathbf{v}^{\text{ini}}, \varphi, J$,
2: Compute L as in (6.6)
3: Compute $\mathcal{H}_{T_{\text{ini}}+L+1}(\mathbf{v}^{\text{data}})$ for constraint (6.14b)
4: Rewrite constraint $\mathbf{y}_0 \models \varphi$ to MILP constraints (6.14c)
5: $\mathbf{u}_{[0,L]} \leftarrow$ solve optimization problem (6.14)

6.5 Soundness and completeness analysis

In this section, we analyze the soundness and completeness of Algorithm 6.1. To do this, we require two assumptions. For soundness, we make an assumption on the length of the initializing trajectory, and for completeness, we require persistence of excitation of the data as a condition.

6.5a Soundness

We say that an algorithm is sound if the following holds. *If the algorithm returns a controller, then the controlled system will satisfy the STL specification.* In the following theorem we show that the direct-data driven control approach summarized in Algorithm 6.1 is sound.

Theorem 6.1 (Direct data-driven control is sound)**.** *Given initialization trajectory* $\mathbf{v}^{ini}$ *of length* $T_{ini} + 1 \geq \mathbf{l}(\mathfrak{B})$*, any solution* $\mathbf{u}_{[0,L]}$ *of Algorithm 6.1 will have a corresponding output* $\mathbf{y}_{[0,L]}$ *that satisfies the STL specification.*

Proof. Since $\mathbf{u}_{[0,L]}$ is a solution of Algorithm 6.1, it is a solution of optimization problem (6.14). Therefore, constraint (6.14b) is satisfied, which implies that the concatenation $\mathbf{v}^{\text{ini}} \wedge (\mathbf{u}_{[0,L]}, \mathbf{y}_{[0,L]})$ is in the behavior of the system, that is $\mathbf{v}^{\text{ini}} \wedge (\mathbf{u}_{[0,L]}, \mathbf{y}_{[0,L]}) \in \mathfrak{B}|_{[0,T_{\text{ini}}+L+1]}$. Following Proposition 6.1, T_{ini} is large enough for $\mathbf{v}^{\text{ini}}$ to fully initialize the behavior, which implies that the output trajectory $\mathbf{y}_{[0,L]}$ is a unique optimal solution and, therefore, the only possible output of the system. Since the MILP constraints (6.14c)-(6.14d) are also satisfied, it holds by construction that this output trajectory satisfies the STL specification φ, that is $\mathbf{y}_{[0,L]} \models \varphi$ holds. Concluding, the controller $\mathbf{u}_{[0,L]}$ obtained from Algorithm 6.1 applied to the system $\mathbf{\Sigma}$, will yield an output trajectory $\mathbf{y}_{[0,L]}$, such that $\mathbf{y}_{[0,L]} \models \varphi$ holds. This concludes the proof of Theorem 6.1. □

6.5b Completeness

We say that an algorithm is complete if the following holds. *If a controller exists such that the controlled system satisfies the specification, the algorithm will find it.* The completeness of Algorithm 6.1 depends on the solver used in the last line, therefore, we analyze the completeness of optimization problem (6.14) instead of the completeness of Algorithm 6.1. To this end, we state the following.

Theorem 6.2 (Completeness of the optimization problem)**.** *Given a data sequence* $\mathbf{v}^{data}$ *from the controllable system* $\mathbf{\Sigma}$ *that is persistently exciting of order* $T_{ini} + L + 1 + n_{\text{x}}$ *according to Definition 6.2. If there exists an input* $\mathbf{u}_{[0,L]}$ *such that the corresponding output* $\mathbf{y}_{[0,L]}$ *satisfies the STL specification, then the optimization problem (6.14) has a feasible solution.*

Proof. Since $\mathbf{v}_{[0,L]} = (\mathbf{u}_{[0,L]}, \mathbf{y}_{[0,L]})$ is a trajectory of $\mathbf{\Sigma}$ and $\mathbf{v}^{\text{data}}$ is persistently exciting according to Definition 6.2, we can follow Proposition 6.2 to conclude that there exists α such that (6.14b) is satisfied. Since $\mathbf{y}_0$ satisfies the STL specification, the set of MILP constraints (6.14c)-(6.14d) is satisfied. Since all constraints are satisfied, optimization problem (6.14) has a feasible solution. This concludes the proof of Theorem 6.2. □

6.6 Case studies

In this section, we apply our approach to two case studies and evaluate the results. All simulations have been performed on a computer with a 2.3 GHz Quad-Core Intel Core i5 processor and 16 GB 2133 MHz memory. For each case study, we compute the average computation time after performing 10 simulations and mention the observed standard deviation. The computation time includes all operations, so also includes acquiring data from the data-generating systems. Besides that, we compute the memory usage by considering all data stored in the `MATLAB` workspace.

6.6a Car platoon

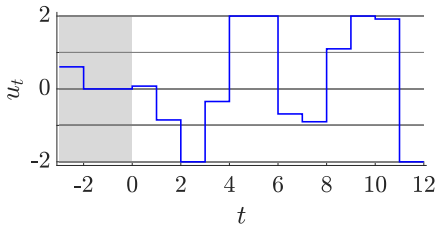
Inspired by Haesaert and Soudjani (2020), as a first case study, we consider a car platooning example with two cars, a leader and a follower. To design a data-driven controller that controls this distance, we get data from the following data-generating system with the corresponding matrices defined in the appendix of this chapter.

$$\begin{aligned} x_{t+1} &= A_1 x_t + B_1 u_t \\ y_t &= C_1 x_t. \end{aligned} \tag{6.15}$$

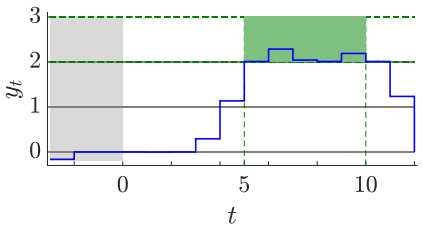
The data-generatingsystem has as output $y_t = x_{t,1}$ the distance between the cars. Furthermore, its three-dimensional state consists of $x_{t,2}$ and $x_{t,3}$ denoting, respectively, the velocity of the follower car and the leader car. We are interested in the distance between the two cars. We consider a bounded input $u \in \mathbb{U} = [-2, 2]$ that influences the velocity of the follower car and, therefore, also the distance between the cars. Note that the analytic form of the data-generating system (6.15) is only used to generate a data sequence $\mathbf{v}^{\text{data}}$ of length 31.

We consider two different scenarios. In the first scenario we want to achieve a safe following distance, while in the second scenario, we want to drive close to the leading car. In both scenarios, we evaluate the results for two different cost functions. Cost function $J_1(\mathbf{u}_0, \mathbf{y}_0) = \|\mathbf{y}_{[0,L]}\|$ minimizes the distance between the cars, while $J_2(\mathbf{u}_0, \mathbf{y}_0) = \|\mathbf{u}_{[0,L]}\|$ minimizes the actuation of the car. For both scenarios, we use $L = 12$.

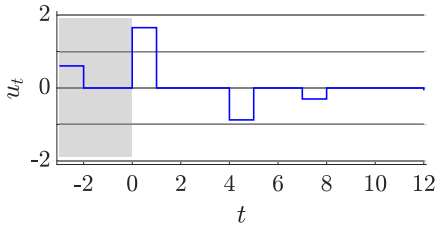
Scenario 1, safe following distance. In this scenario, both cars started very close to each other with the same constant velocity. We have designed a controller for the follower car such that it achieves a safe distance to the leader car and maintains this for a specified time frame. This can be written using STL as $\varphi = \square_{[5,10]}(|y| \geq 2 \wedge |y| \leq 3)$. Note that we did not specify which car should



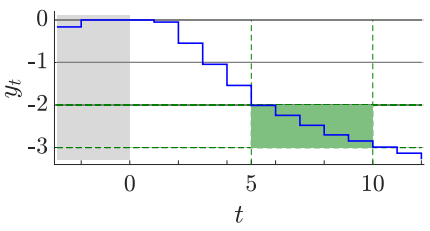
(a) Input u_t at time steps t.



(b) Output y_t at time steps t in blue.



(c) Input u_t at time steps t.



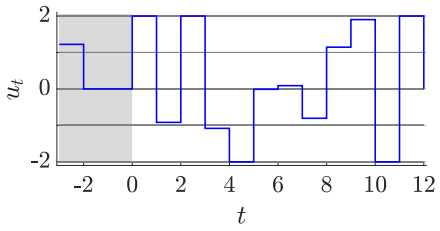
(d) Output y_t at time steps t.

Figure 6.2: Results of Scenario 1 of the car platoon problem for cost function $J_1(\mathbf{u}_0, \mathbf{y}_0) = \|\mathbf{y}_{[0,L]}\|$, displayed in panels **(a)**, **(b)** and for cost function $J_2(\mathbf{u}_0, \mathbf{y}_0) = \|\mathbf{u}_{[0,L]}\|$, displayed in panels **(c)**, **(d)**.

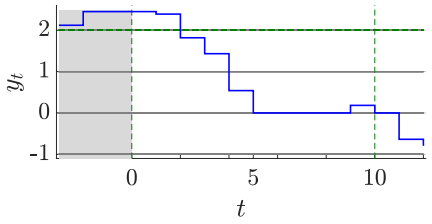
drive in front. After obtaining data $\mathbf{v}^{\text{data}}$, we have considered initial trajectory $\mathbf{v}^{\text{ini}} = (\mathbf{u}^{\text{ini}}, \mathbf{y}^{\text{ini}})$ with $\mathbf{u}^{\text{ini}} = \{0.6058, 0, 0\}$ and $\mathbf{y}^{\text{ini}} = \{-0.1636, 0, 0\}$, and obtained the results in Figure 6.2. Here, $\mathbf{v}^{\text{ini}}$ is highlighted in the gray area. We observed an average computation time of 3.85 seconds, with a standard deviation of 0.32 seconds and a memory usage of 3.35 MB.

In Figures 6.2b, 6.2d, we see that the specification (indicated by the green box) is satisfied, since the output at time steps $t = 5, \ldots, 10$ is between 2 and 3 in Figure 6.2b and between -3 and -2 in Figure 6.2d. Besides that, we conclude that the cost function has a big influence on the results. For cost function J_1, we see that the controller minimizes the distance between the cars, keeping it as close to zero as possible. This is not the case for cost function J_2, since the absolute distance increases over the complete time horizon. Besides that, comparing Figures 6.2a and 6.2c, we can conclude that for cost function J_2 the input is substantially smaller over the complete horizon than when cost function J_1 is considered.

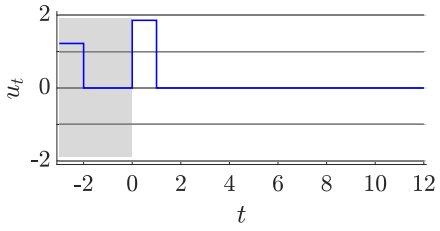
Scenario 2, close following distance. In this scenario, the cars start with a rather large distance between them and with the same constant velocity. We have designed a controller for the follower car such that it drives towards the leader car and remains close to it for several time steps. This can be written using STL as $\varphi = \Diamond_{[0,10]} \Box_{[0,3]} |y| \leq 2$. After obtaining data $\mathbf{v}^{\text{data}}$, we have considered initial trajectory $\mathbf{v}^{\text{ini}} = (\mathbf{u}^{\text{ini}}, \mathbf{y}^{\text{ini}})$ with $\mathbf{u}^{\text{ini}} = \{1.2224, 0, 0\}$ and $\mathbf{y}^{\text{ini}} = \{2.12, 2.45, 2.45\}$, and obtained the results in Figure 6.3. Here, the initial trajectory is highlighted by the gray area. We see that the specification is satisfied for both cost functions since the output is between -2 and 2 for at least 3 time steps within the time frame



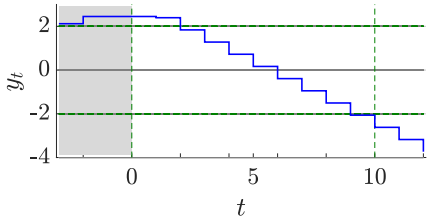
(a) Input $u_t$at time steps t.



(b) Output y_t at time steps t.



(c) Input u_t at time steps t.



(d) Output y_t at time steps t.

Figure 6.3: Results of Scenario 2 of the car platoon problem for cost function $J_1(\mathbf{u}_0, \mathbf{y}_0) = \|\mathbf{y}_{[0,L]}\|$, displayed in panels **(a)**, **(b)** and for cost function $J_2(\mathbf{u}_0, \mathbf{y}_0) = \|\mathbf{u}_{[0,L]}\|$, displayed in panels **(c)**, **(d)**.

$[0, 10]$. Besides that, we see that the influence of the cost function is similar to the first scenario. For this scenario, we observed an average computation time of 4.22 seconds, with a standard deviation of 0.17 seconds and a memory usage of 5.7 MB.

6.6b Temperature control in a building

As a second case study, we have considered controlling the temperature inside a building based on the model in Haghighi (2013), where the continuous-time thermodynamics of the multi-room building are approximated with an analogous model of an electric circuit. Details about the model can be found in Haghighi (2013) and Raman et al. (2014). After performing a time-discretization (zero-order hold), with a sampling time of 30 minutes, we have obtained the following data-generating system with the corresponding matrices defined in the appendix of this chapter.

$$\begin{aligned} x_{t+1} &= A_2 x_t + \begin{bmatrix} B_2 & B_d \end{bmatrix} \begin{bmatrix} u_t \\ d_t \end{bmatrix} \\ y_t &= C_2 x_t. \end{aligned} \tag{6.16}$$

The model as in Haghighi (2013) and Raman et al. (2014) has an additional disturbance term $B_d d_t$, which we included as a known time-varying term. Note that the analytic form of the data-generating system (6.16) is only used to generate a data sequence $\mathbf{v}^{\text{data}}$ with a length of 1050.

The goal of the controller is to keep the temperature of a room, given by $y_t = T_{t,\text{room}}$ above a certain time-varying comfort level $T_{t,\text{comf}}$ whenever the room is

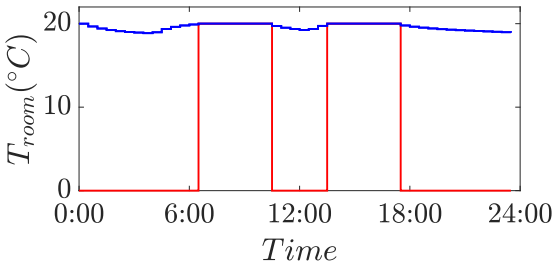


Figure 6.4: Room temperature (in degrees Celsius) with respect to the time of the day of the temperature control case study. The output T_{room} is given in blue and the comfort level combined with the occupancy is given in red.

occupied $\text{occ} = 1$ while minimizing the input. We have considered a time horizon of $L + 1 = 48$ (corresponding to 24 hours). The STL specification is given as $\varphi_{\text{temp}} = \square_{[0,L+1]}\text{occ} \implies y > T_{t,\text{comf}}$ and the cost function is equal to $J(\mathbf{u}_0, \mathbf{y}_0) = \|\mathbf{u}_0\|$.

We have obtained data $\mathbf{v}^{\text{data}}$, and initialized with $\mathbf{v}^{\text{ini}} = (\mathbf{u}^{\text{ini}}, \mathbf{y}^{\text{ini}})$ with $\mathbf{u}^{\text{ini}}$ and $\mathbf{y}^{\text{ini}}$ equal to respectively 43.65 and 20 for 5 time steps. After converting from degrees Fahrenheit to degrees Celsius, we have obtained the results as shown in Figure 6.4. Here, the blue line is the observed output $y_t = T_{t,\text{room}}$ and the red line is the threshold value for the temperature. This is obtained by combining the occupancy with the comfort level, that is $T_{t,\text{ref}} = \text{occ}_t * (T_{t,\text{comf}})$, with $\text{occ}_t = 1$ if the room is occupied and 0 otherwise. In Figure 6.4, we can see that the specification φ_{temp} is satisfied, since $T_{t,\text{room}} \geq T_{t,\text{ref}}$. For this case study, we observed an average computation time of 26.99 seconds, with a standard deviation of 7.9 seconds and a memory usage of 81.1 MB.

6.7 Discussion on the results

This chapter is a first step towards direct data-driven control with temporal logic specifications. In this section, we describe our vision on multiple possible extensions. These extensions will allow this theory to eventually become applicable to realistic systems.

Extension to longer time horizons. The approach in this chapter is currently limited to finite-time horizon specifications, however, an extension to infinite-time horizon is feasible. In this case, one can assume some repeating behavior and adjust the trajectory length L to the length of the repeated behavior. More specifically, an infinite trajectory can be viewed as a finite trajectory followed by a loop, hence assuming periodic continuation of a trajectory. In that case, we can satisfy infinite-time properties. Similar approaches already exist in the literature on data-driven methods, such as the multi-shooting method (Ou et al. 2022) to extend the length of the finite trajectory that you can represent with your data. An alternative is to extend this approach in a receding horizon sense. In that case, by showing recursive feasibility, we can handle infinite horizon properties as well.

Beyond signal temporal logic specifications. Besides signal temporal logic, there exist multiple temporal logic languages to describe the desired behavior of a system. A very common one is Linear Temporal Logic (LTL). Writing LTL specification into MILP constraints has already been developed through bounded model checking in Schuppan et al. (2006) and is extended to infinite-time properties in Wolff et al. (2014). In the latter, they make use of the assumption that the trajectory is eventually periodic. The approach in this chapter could be extended to such specifications in the future as well.

Beyond Linear Systems. The behavioral theory and the Fundamental Lemma have been extended to the field of Linear Parameter-Varying (LPV) systems (Verhoek et al. 2021b) and their control (Verhoek et al. 2021a). Since the LPV framework has the potential to represent nonlinear and time-varying systems (Tóth 2010), and direct data-driven control approaches are currently being developed (Verhoek et al. 2022), the approach in this chapter could be extended to such systems in the future as well.

Extensions to noisy data. In this work, we considered data that is noise free. In recent work, on data-driven control for LTI systems, methodologies have been developed to cope with noisy data (Berberich et al. 2020a; Pan et al. 2022a), including developments of a stochastic Fundamental Lemma. Applying these techniques can push this line for data-driven control subject to temporal logic properties further and to more realistic settings. This is an interesting, but challenging topic for future research.

Extension to stochastic systems. In this work, we focused on deterministic systems, however, as discussed in Chapter 1 of this thesis, safety-critical systems are best modeled by *stochastic systems*. In contrast to noisy data, where a noise signal influences the *output* of the system, for stochastic systems a stochastic disturbance influences the *state dynamics* of the system. An extension of the direct data-driven technique within the behavioral framework towards stochastic systems is ongoing work and results are emerging in literature (Pan et al. 2022a; Pan et al. 2022b; Faulwasser et al. 2023). Hence, we believe this methodology shows promise toward the extension to stochastic systems making this approach more applicable to realistic settings as well.

6.8 Conclusion

In this chapter, we have developed a direct data-driven controller synthesis method for linear time-invariant systems subject to a temporal logic specification, which does not require an explicit modeling step. We build upon the promising results of the behavioral framework and the Fundamental Lemma, which allows us to obtain a characterization of the system, after collecting a single sequence of input-output data from it. By exploiting existing results on rewriting signal temporal logic specifications to MILP constraints, we can efficiently solve an optimization problem to automatically synthesize a control policy that ensures that the specifications are satisfied during closed-loop operation of the system. We have analyzed the soundness and completeness of our algorithm and applied it to multiple benchmark

simulation examples. The main benefit of this method is that it only requires input-output data of the system and no exact knowledge of the structure of the system itself. We have elaborated on our vision for future work and showed that extensions of this method are very promising considering its application to realistic systems.

Appendix

6.A Data-generating systems

To generate the data required for the case studies, we used the following matrices for the car platoon dynamics.

$$A_1 = \begin{bmatrix} 1 & -0.3 & 0.3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, B_1 = \begin{bmatrix} -0.03 \\ 1 \\ 0 \end{bmatrix}, C_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}.$$

We used the following matrices for the system describing the temperature in a building.

$$\begin{aligned} A_2 &= \begin{bmatrix} 0.9233 & 0.00135 & 0.0009377 & 0.002662 & 0.03775 \\ 0.0009377 & 0.9606 & 0.0004754 & 0.00135 & 0.01928 \\ 0.0009377 & 0.0006846 & 0.9604 & 0.00135 & 0.01928 \\ 0.001849 & 0.00135 & 0.0009377 & 0.9241 & 0.03775 \\ 0.07636 & 0.05617 & 0.039 & 0.11 & 0.7142 \end{bmatrix}, \\ B_2 &= \begin{bmatrix} 3.1194\text{e}{-4} \\ 1.5815\text{e}{-4} \\ 1.5815\text{e}{-4} \\ 3.1194\text{e}{-4} \\ 0.0131 \end{bmatrix}, \\ C_2 &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \end{aligned}$$

and

$$B_d = \begin{bmatrix} -8.0390\text{e}{-6} & 0.0340 & 1.9696\text{e}{-5} & 3.2720\text{e}{-5} & 0.0014 & 0 & 0 \\ -4.0756\text{e}{-6} & 1.1479\text{e}{-5} & 0.0173 & 1.6530\text{e}{-5} & 0.0230 & 0 & 0 \\ -4.0756\text{e}{-6} & 1.1479\text{e}{-5} & 0.0173 & 1.6530\text{e}{-5} & 0.0007 & 0 & 0 \\ -8.0390\text{e}{-6} & 2.2722\text{e}{-5} & 1.9696\text{e}{-5} & 0.0340 & 0.0014 & 0 & 0 \\ -3.3691\text{e}{-4} & 0.0014 & 0.0011 & 0.0021 & 0.0568 & 0 & 0 \end{bmatrix}.$$

7 Conclusions and future research directions

In this chapter, we provide conclusions on this thesis and discuss future research directions. We re-iterate on the objective to improve the reliability of safety-critical systems by giving guarantees on their behavior. We do this by reflecting on the research questions, and by providing conclusions and the main results described in this thesis. After a short recap of the general objective, we summarize the main contributions of each chapter and discuss their results. Next, we take a step back and envision the most important research directions for the future.

7.1 Conclusions

To cope with the increasing level of intelligence in autonomous systems and their interaction with people, there is a growing need to achieve a high level of *reliability* in their operation, usage, and performance. This is especially the case for safety-critical systems, where malfunctions have disastrous consequences. In this thesis, we have worked towards achieving a higher level of reliability that cannot be achieved through (extensive) scenario testing. To this end, we have explored the area of formal methods to automatically synthesize controllers subject to temporal logic specifications.

The main focus of this thesis has been to explicitly take *uncertainty* into account, while maintaining the guarantees obtained through formal methods. More specifically, as formulated in the research question, we have set the goal to *manage uncertainty* when automatically synthesizing controllers subject to temporal logic requirements. We have identified two main sources of uncertainty:

1. The *stochastic influence on the state dynamics*, which is usually caused by uncertain environments, such as the weather or behavioral, medical, biological or cognitive characteristics of people.

2. *Model uncertainty*, such as parametric uncertainty, measurement noise, and unmodeled dynamical features. This type of uncertainty is particularly relevant when obtaining a *model*.

In the same fashion, we have split this thesis into two parts. In Part I, we have focused on the first type of uncertainty by considering stochastic models. In Part II, we have focused on model uncertainty by both extending some of the theory in Part I and by giving an initial data-driven approach for unknown deterministic systems. We have disregarded measurement noise in this thesis.

7.1a Coupling-based framework for stochastic systems

A well-known approach to automatically synthesize controllers for continuous-state stochastic systems is through a finite-state approximate model, called an *abstraction*. For such techniques, it is crucial to accurately quantify the similarity between the original (continuous-state) model and its (finite-state) abstraction. In Chapter 2, we have developed a general framework that makes it possible to reason about, and efficiently compute this similarity relation that yields an explicit quantification of accuracy of the abstraction by designing a *coupling* between the two stochastic models.

We have done this in a complex setting with respect to both the considered models and specifications. More specifically, we have focused on stochastic models without imposing restrictions on the variance of the stochastic disturbance. Besides that, we made sure that our approach is suitable for specifications with an infinite-time horizon by computing a fixed point. By doing so, we have solved the first subquestion, that is *How to automatically synthesize controllers for stochastic systems influenced by an unbounded disturbance, such that it provably satisfies specifications over infinite-time horizons, and accurately compute this satisfaction probability?*

The main conclusions of Chapter 2 can be summarized as follows:

- We have developed the concept of a coupling compensator to align general continuous-state stochastic models with their abstraction (e.g. an approximation that can be finite-state or reduced-order). It allows for models with probability measures of possibly unbounded support and specifications with an infinite-time horizon.
- To facilitate an accurate computation of the lower bound on the satisfaction probability, we have connected the coupling compensator to simulation relations that quantify the similarity between the two stochastic models through the deviations in outputs and in transition probability.
- We have explicitly worked out this coupling-based concept for linear stochastic systems subject to disturbances with unbounded support. This has led (in Theorem 2.1, Chapter 2) to an explicit and computable upper bound for approximate simulation relations through controlled invariant sets.

- We have developed a generalization of this framework to piecewise-affine abstractions for more general nonlinear stochastic systems. This has led to an explicit quantification of the similarity between a nonlinear stochastic model and its piecewise-affine abstraction in Theorem 2.4 of Chapter 2.
- We have illustrated the theory with multiple examples, showing computability and feasibility of the theoretical results.

Discussion. The results of Chapter 2 do not imply that control synthesis for stochastic systems is now fully completed. There is still room for improvement, even in this specific setting. Although the theoretical part of Chapter 2 functions as a strong mathematical foundation, the approach on its own is not suitable for large-scale systems with a high complexity, due to the required space discretization. We expect that this can be achieved by building upon the developed theory. Since this chapter provides a general framework that simplifies reasoning about and computation of the similarity quantification, it is very suitable for extensions, as done in, for example, Chapters 4 and 5.

Further improvements with respect to the computational implementation are also possible. The implementation is currently based on simulation relations with an ellipsoidal shape, while it is more general to consider polytopes or zonotopes. It is expected that such an extension gives more accurate results, while possibly also improving the computation time.

Additionally, the theory is set up in a general fashion, but in the implementation, we consider systems with an additive disturbance generated by a distribution with an unbounded support. In general, such models are very difficult to tackle in automatic control synthesis subject to temporal logic requirements, and many methods give a trivial lower bound on the satisfaction probability. As part of future work, it would be interesting to see the impact of our approach on models with multiplicative noise and/or disturbances coming from a distribution with a bounded support.

Finally, the approach for nonlinear systems could be improved significantly. Our approach based on a piecewise-affine abstraction adds a lot of computational complexity, computation time, and memory usage. So, developing a computationally more efficient approach specifically for nonlinear systems is an interesting topic for future work. One possible first step is learning the piecewise-affine abstraction in an efficient manner using neural networks, as introduced in Abate et al. (2023).

7.1b Efficient software tool

Besides developing theoretical results it is crucial to develop software tools that are applicable to multiple case studies. Initially, software is often used as an implementation of the theory to verify its correctness, to see if there are gaps in the theory, and to evaluate its impact. However, when software is written in a general way, it can achieve even more. It can among others be used by other researchers to continue the development, to compare several methods to each other, or to motivate people to go into this research direction. In Chapter 3, we have developed a `MATLAB` toolbox, called `SySCoRe`, that based on the theory in Chapter 2, synthesizes controllers for stochastic continuous-state systems to satisfy temporal

logic specifications. The toolbox is structured into several steps to make it easy for users to alter specific parts. Those steps are illustrated in Figure 7.1.

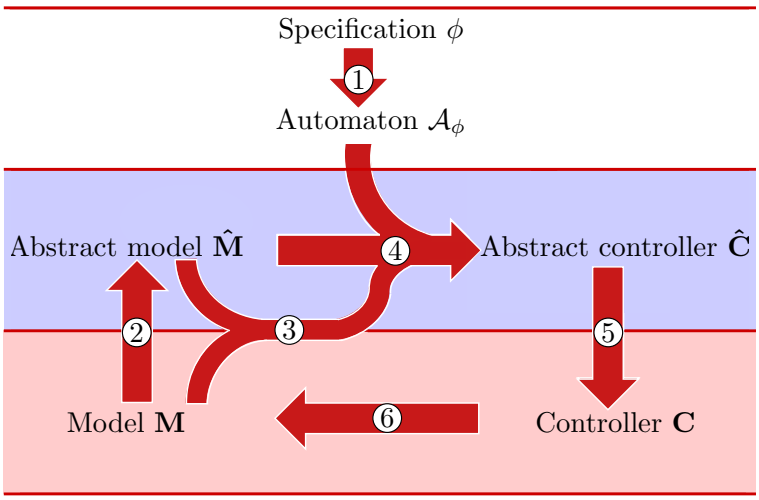


Figure 7.1: Illustration of the different steps that can be performed by `SySCoRe`.

Discussion. `SySCoRe` is at an early stage of development, so there is a lot of work ahead to keep up with other existing tools. Currently, the main limitation is the large amount of tuning that is required to get good results. To make it easier for users, an automatically tuned version where the user only has to input the problem parameters is desired.

7.1c A versatile approach through multiple layers

In Chapter 4, we have increased the accuracy of our approach by extending the method from Chapter 2 to allow for switching between different pairs of deviation bounds. Here, we constructed multiple layers, each with its own simulation relation and corresponding deviation pair. We refer to this as a multi-layered approach with homogeneous layers. In Chapter 2, we have used a simulation relation that quantifies the similarity between the original model and its finite-state abstraction, through two deviation bounds; the deviation in output and in probability. When quantifying the similarity between the two models, decreasing one of the deviation bounds leads to an increase in the other one. Hence, there is a clear trade-off between the two deviation bounds. Furthermore, we noticed that the accuracy of the lower bound on the satisfaction probability strongly depends on the choice of deviation bounds.

As formulated in the third research question *How to improve the computational efficiency and scalability of provably correct control synthesis methods for stochastic systems, while maintaining accurate lower bounds on the satisfaction probability?*, we are also interested in improving the computational efficiency and scalability

of such methods. In Chapter 2 and 3, we have been focusing on discretization-based techniques, but when it comes to efficiency and scalability, discretization-free techniques are more promising. More specifically, discretization-free approaches are generally fast and efficient, but the underlying method limits the achievable performance. On the other hand, for discretization-based approaches, the achievable performance increases with the available memory and computation time. Therefore, in Chapter 4, we have also extended the multi-layered methodology to allow for switching between layers with discretization-free models and discretization-based models. We refer to this as a multi-layered approach with heterogeneous layers.

The main conclusions of Chapter 4 can be summarized as follows:

- We have motivated the idea behind a multi-layered approach with homogeneous and heterogeneous layers, and explained why this improves the computational efficiency and scalability of provably correct control synthesis while maintaining accuracy.
- We have made a connection between the dynamic programming operators and the switching between layers. This has led to the implementation of a *multi-layered DP approach* that assesses multiple value functions (one per layer) and takes into account the different deviation pairs while maintaining the probability of satisfying specifications over infinite-time horizons. On the computational side, we have developed an approach for approximating the optimal strategy for switching between homogeneous layers through a surrogate model (Algorithm 3 in Chapter 4) that is computationally attractive.
- To maintain the lower bound on the satisfaction probability computed on the abstract model with multiple layers, we have developed a control refinement approach that yields a robust controller for the original model.
- We have explicitly worked out the multi-layered approach in a linear setting leading to an explicit quantification of approximate simulation relations (stated in Theorem 4.1).
- We have motivated the importance of including a discretization-free layer for an efficient and scalable approach and described what it should look like, such that it can be connected to the discretization-based layers. We have explicitly defined when switching between heterogeneous layers is allowed and made a connection between the dynamic programming operators and the switching between layers. This led to the implementation of a *heterogeneous DP approach* that combines multiple value functions and directly determines when to switch between layers while maintaining the probability of satisfying specifications over infinite-time horizons.
- We have illustrated the theory with multiple examples, showing computability and feasibility of the theoretical results.

Discussion. The developed theory should be followed with the development of a toolbox that implements the concepts in this chapter and that allows for further extension including an advanced method in the discretization-free layer and a complex case study showing the full potential of this approach. This is something we plan to do for a paper that is currently in preparation.

7.1d Stochastic models with explicit parametric uncertainty

In the second part of this thesis, we have considered an additional type of uncertainty, namely model uncertainty. We have expanded the theory of Chapter 2 to models with explicit parametric uncertainty. To this end, we used two main steps. First, we obtained a credible set for the unknown parameterization from data by using Bayesian linear regression. Next, we defined a sub-simulation relation based on concepts from Chapter 2 to quantify the similarity between the original model and the nominal model in the credible set. By refining the control in a suitable manner, we obtained a controller that is independent of the unknown parameter and is robust against parametric uncertainty. Thus, we have given an answer to the question: *After obtaining a stochastic model with explicit parametric uncertainty from data, how to synthesize provably correct controllers that are robust against parametric uncertainty?*.

The main conclusions of Chapter 5 can be summarized as follows:

- Based on a given data set and a specified confidence level, we have defined the concept of a *credible set* that with a certain confidence contains the true but unknown parameters of the parameterized system. We did this such that it allows a connection to correct-by-design control synthesis and gave an explicit characterization of this credible set.
- To tackle parametric uncertainty, we have defined a sub-probability coupling and sub-simulation relation that is required to quantify the similarity between a parameterized system and its abstraction. We connected those concepts (and most of the concepts in Chapter 5) to the original coupling and simulation relation from Chapter 2.
- We have developed a (clever) interplay between state mappings and interface functions to infer a control refinement procedure for parameterized systems. In this context, we defined conditions for a *valid control refinement* that led to Theorem 5.2, where we conclude that a valid control refinement always exists if there is a sub-simulation relation between the parameterized system and its abstraction.
- We have shown that the transitivity property holds for the sub-simulation relation (Theorem 5.3), which is required to connect this approach to finite-state abstract models and to compute a provably correct controller.
- For nonlinear systems, we have explicitly derived expressions for the deviation bounds of the sub-simulation relation, as in Theorem 5.4.
- We have illustrated the theory with two examples; a linear model and a nonlinear model, showing computability and feasibility of the theoretical results.

Discussion. The current implementation could be conservative for different types of model uncertainty, future work is on the development of improved algorithms that can deal with this.

Besides that, for this method, we require input-*state* data of the system. However, in practice, you often obtain input-*output* data from the system instead. It would

therefore be interesting to adjust this method such that it can handle input-output data. By doing so, an extension to noisy data might also be feasible.

7.1e Model-free approach for unknown systems

In Chapter 6, we have given an initial data-driven approach to obtain a provably correct controller without using an explicit model. By doing so, we have prevented modeling errors to influence the controller and rely solely on input-output data. Instead of using a model, we used a data-driven characterization of the system, which we used to synthesize a controller using mixed-integer linear programming. We took a step back and considered *deterministic* systems instead of stochastic systems, but envision possible extensions for the future.

The main conclusions of Chapter 6 can be summarized as follows:

- We have developed a direct data-driven approach for an unknown system that is assumed to be linear time-invariant. Here, the controller synthesis problem amounts to developing a controller directly from data, that (i) minimizes a cost function on input-output variables, while (ii) achieving an STL specification.
- On the computational side, we have shown that this problem is solvable as a mixed-integer linear problem (MILP), provided that conditions (as defined in Chapter 6) on the length of the data sequence are satisfied.
- We have proven that the MILP algorithm developed for direct data-driven control design is both sound and complete as specified in Theorems 6.1 and 6.2.
- We have illustrated the theory with multiple examples, showing computability and feasibility of the theoretical results.

Discussion. Developing direct data-driven methods is a promising direction in which less tuning is required and there are still a lot of open subjects to research. The main direction for future research is the extension to stochastic systems and noisy output measurements. More details are given in the chapter itself.

7.2 Future research directions

The area of automatic control synthesis with temporal logic requirements is moving fast and a lot of developments are happening, however, the step to apply them to realistic situations is big. We envision seven main directions to push the development further that we consider vital to achieving the goal of reliable behavior of safety-critical systems.

a. Generalization
b. Nonlinear systems
c. Partially observable systems
d. Data-driven approaches
e. Network of systems
f. Scalability
g. Computational methods and tooling

Each of these directions is discussed in more detail next.

7.2a Generalization

Generalization is a broad term, where we distinguish between methods and tools that should be generally applicable to different model structures and specifications, as well as a general platform to compare the different methods and tools.

There are a lot of methods and tools within the area of automatic control synthesis, however, most of them are only applicable to a specific situation, hence they are either limited with respect to the model structure or the specification. What is missing is a method and tool that can do it all, or as we do in Chapter 4, a method that combines different approaches. Similarly, a tool that based on the model and specification selects the best tool to use, would bring great benefits to this research area.

Within research, we notice the growing importance of comparing different methods and tools to each other. However, due to the large number of model structures, modeling techniques, and languages to describe specifications, comparing methods and their computability is very difficult. Especially, since the developed methods and tools are often focusing on a specific model structure and specification language. To push the development further, a general platform to compare methods and tools is necessary. This is also the reason the ARCH-COMP (Applied veRification for Continuous and Hybrid systems COMPetition) has been initiated. Within this friendly competition, researchers from different areas develop benchmarks to which they apply their tools and compare the results. However, the same issues as mentioned before are occurring within ARCH. The benchmarks are pretty much tailored to what a specific tool needs, hence lacking generality.

To assist researchers in developing methods and tools that are generally applicable, a platform that allows them to easily compare different methods and tools would be beneficial.

7.2b Nonlinear systems

In this thesis, we have developed a theory that is applicable to both linear and (classes of) nonlinear systems. However, the implementation for nonlinear systems requires additional research. This is especially the case for the development of efficient computational methods that require small computation time and memory usage when automatically synthesizing controllers for nonlinear systems. Since many realistic systems behave in a nonlinear fashion, this development is crucial. In the previous section, we already mentioned the usage of neural networks that can be directly used in combination with the implementation discussed in Chapter 2. As an alternative, we can consider the linear parameter varying (LPV) framework. This framework embeds the nonlinear system dynamics in an easier-to-work-with model structure and performs well in guaranteeing stability and performance (Koelewijn 2023). The concept of the coupling compensator introduced in Chapter 2 enables the development of an efficient implementation for nonlinear systems through the LPV framework.

7.2c Partially observable systems

Extensions towards partially observable systems, such as systems influenced by measurement noise, are crucial for automatic control synthesis methods to succeed. For partially observable systems, a different simulation relation is necessary. More specifically, the simulation relations should be extended from the state space to relations over probability distributions of the state spaces. Combining such a simulation relation with Kalman filters or belief spaces might be suitable for partially observable systems that are influenced by measurement noise. However, this is still a challenging topic to tackle. The main challenge is due to the fact that the labeling function is defined over the outputs of a system. When there is uncertainty about the outputs, e.g., due to measurement noise, this directly affects the labels and, therefore, also the deterministic finite-state automaton and the complete control synthesis. Even though there exists theoretical research on this topic (Ding et al. 2013; Lesser and Oishi 2016; Badings et al. 2021), there is a lack of methods that are computationally efficient while not being restricted to safety specifications.

Neural networks for an uncertain environment. In this thesis, we consider the (static) environment, i.e. the labels of the outputs, to be known. However, in practice, one has to consider an unknown, uncertain, or changing environment. One way to alleviate this issue is by using data. For example, in the signal processing domain one can train neural networks to recognize and classify objects in the environment. This research allows one to not only perform the classification but also attach a confidence to their observation (that is a probability). By comparing data to the trained neural network, one can distinguish between (moving or parked) cars, (moving or parked) bicycles, trees, people, animals, and so on. By building upon Chapters 2 and 5, we can now make the steps towards this additional type of uncertainty.

7.2d Data-driven approaches

Behavioral framework. In Chapter 6, we have discussed an approach with a lot of potential, since it is model-free and requires only a single sequence of input-output data. Its extension towards stochastic systems with noisy data is very challenging and at an early stage of development. We consider direct data-driven approaches as very valuable and, therefore, this topic is worth exploring. Details about possible extensions of Chapter 6 are given in the chapter itself.

Learning in real-time. An interesting topic related to the data-driven research direction is learning in real time. Hence, instead of only designing a controller beforehand, we can update it based on the data that we acquire while performing the tasks. Such methods require an efficient computation time with small memory usage, which is one of the major challenges in this research area. Learning in real-time can be used on different levels concerning the uncertainty of a model, such as completely unknown dynamics, an unknown parameterization, or an unknown disturbance distribution. We can also use it to tackle uncertainty about the environment as discussed in Section 7.2c.

7.2e Network of systems

Besides that, it is interesting to look into networks of systems. On the one hand, we can model the environment as a system and consider the interconnection between a system and its environment as a network, or we consider several systems that perform tasks and communicate with each other through signals. This constitutes a networked system. The latter is a research area that is often considered in research (Lavaei et al. 2019; Lavaei et al. 2020c; Qi et al. 2023). The former is less covered in research.

Abstraction of the environment. As an example of a networked model, one may consider incorporating the environment into a model and describe the way it interacts with the model. An abstracted model can, for example, be obtained by abstracting the environment separate from abstracting the model. For example, when an autonomous vehicle wants to overtake, it is enough to roughly estimate the relative velocity of the vehicle in front. Instead of estimating its dynamics (e.g. velocity and acceleration) *exactly* it seems sufficient to decide among the three states whether the vehicle is *driving slower, at the same speed* or *faster* than the controlled system. By using such an abstraction of the environment, we might be able to quickly make decisions, while maintaining a high satisfaction probability. The main challenge here is to define the interaction between a system and its environment and how to give guarantees for a network consisting of a system and an environment (or even multiple environments).

7.2f Scalability

Realistic systems are often complex, behave in a nonlinear fashion, and require models with a state space of a *high dimensional order*. In this thesis, we presented

methods and tools that push the envelope in scalability and computational efficiency of stochastic control synthesis. In Chapter 2, we show how to reduce the dimension of a system through model-order reduction techniques, which helps with scalability issues. In Chapter 4, we combine discretization-based methods with discretization-free methods, which have less issues with scalability. This is a promising direction in solving the scalability issue but requires more research. Furthermore, a thorough analysis of the scalability of the methods developed in this thesis is necessary to tackle it in an efficient manner.

In order to achieve applicability to realistic systems, we need to develop methods that can handle high-dimensional models, hence scalability has to be a main focus during development. Obviously this goes hand in hand with the development of scalable computational tools, as described next.

7.2g Computational methods and tooling

To apply automatic control synthesis approaches to realistic systems, we need computational methods and tools that are efficient with respect to computation time and memory usage. With our `MATLAB` toolbox `SySCoRe` (Chapter 3 of this thesis), we contribute to the tooling need of automatic control synthesis.

The improvement of existing tools, their extensions to higher-order systems including for example stochastic state transitions and noisy data, and the development of new tools is crucial for the adoption of automatic control synthesis by industry.

Bibliography

Abate, A., Blom, H., Cauchi, N., Delicaris, J., Hartmanns, A., Khaled, M., Lavaei, A., Pilch, C., Remke, A., Schupp, S., et al. (2020). ARCH-COMP20 Category Report: Stochastic Models. *In Proceedings of 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH 2020)*. Vol. 74. EasyChair, p. 76–106.

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.

Abate, A., Blom, H., Bouissou, M., Cauchi, N., Chraibi, H., Delicaris, J., Haesaert, S., Hartmanns, A., Khaled, M., Lavaei, A., et al. (2021). ARCH-COMP21 Category Report: Stochastic Models. *In Proceedings of 8th International Workshop on Applied Verification of Continuous and Hybrid Systems, (ARCH 2021)*. Vol. 80. EasyChair, p. 55–89.

Abate, A., Blom, H., Cauchi, N., Haesaert, S., Hartmanns, A., Lesser, K., Oishi, M. M. M. K., Sivaramakrishnan, V., Soudjani, S. E. Z., Vasile, C. I., et al. (2018). ARCH-COMP18 Category Report: Stochastic Models. *In Proceedings of 5th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH 2018)*. Vol. 54. EasyChair, p. 71–103.

Abate, A., Blom, H., Delicaris, J., Haesaert, S., Hartmanns, A., Huijgevoort, B. van, Lavaei, A., Ma, H., Niehage, M., Remke, A., et al. (2022). ARCH-COMP22 Category Report: Stochastic Models. *In Proceedings of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems, (ARCH 2022)*. Vol. 90. EasyChair, p. 113–141.

Abate, A., Edwards, A., Giacobbe, M., Punchihewa, H., and Roy, D. (2023). Quantitative verification with neural networks for probabilistic programs and stochastic systems. *arXiv preprint*.

Abrial, J.-R. (1996). *The B-book*. Vol. 1. 6. Cambridge university press Cambridge.

Adams, R. A. and Essex, C. (2009). *Calculus: a complete course*. 7th ed. Vol. 4. Pearson Addison Wesley.

Aksaray, D., Jones, A., Kong, Z., Schwager, M., and Belta, C. (2016). Q-learning for robust satisfaction of signal temporal logic specifications. *In Proceedings of*

55th Conference on Decision and Control (CDC). IEEE, p. 6565–6570.

Allen, E. J., Allen, L. J., Arciniega, A., and Greenwood, P. E. (2008). Construction of equivalent stochastic differential equation models. *Stochastic analysis and applications*, 26(2):274–297.

Althoff, M., Stursberg, O., and Buss, M. (2010). Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear analysis: hybrid systems*, 4(2):233–249.

Alur, R. (2015). *Principles of cyber-physical systems*. MIT press.

Anand, M., Lavaei, A., and Zamani, M. (2021). Compositional Synthesis of Control Barrier Certificates for Networks of Stochastic Systems against ω-Regular Specifications. *arXiv preprint*.

Anand, M., Lavaei, A., and Zamani, M. (2022). From small-gain theory to compositional construction of barrier certificates for large-scale stochastic systems. *IEEE Transactions on Automatic Control*, 67(10):5638–5645.

ApS, M. (2019). Mosek optimization toolbox for matlab. *User's Guide and Reference Manual, Version*, 4.

Asarin, E., Dang, T., and Girard, A. (2007). Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43(7):451–476.

Axelrod, C. W. (2013). Managing the risks of cyber-physical systems. *In 2013 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. IEEE, p. 1–6.

Badings, T. S., Jansen, N., Poonawala, H. A., and Stoelinga, M. (2021). Filter-Based Abstractions for Safe Planning of Partially Observable Dynamical Systems. *arXiv preprint*.

Badings, T. S., Romao, L., Abate, A., and Jansen, N. (2023). Probabilities are not enough: Formal controller synthesis for stochastic dynamical models with epistemic uncertainty. *In Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. (12), p. 14701–14710.

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

Banerjee, S. S., Jha, S., Cyriac, J., Kalbarczyk, Z. T., and Iyer, R. K. (2018). Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data. *In 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, p. 586–597.

Belitskii, G. R. and Lyubich Y. I. (1988). *Matrix norms and their applications*. Birkhäuser Verlag.

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

Belta, C., Bicchi, A., Egerstedt, M., Frazzoli, E., Klavins, E., and Pappas, G. J. (2007). Symbolic planning and control of robot motion [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):61–70.

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.

Berberich, J. and Allgöwer, F. (2020). A trajectory-based framework for data-driven system analysis and control. *In Proceedings of 2020 European Control Conference (ECC)*. IEEE, p. 1365–1370.

Berberich, J., Koch, A., Scherer, C. W., and Allgöwer, F. (2020a). Robust data-driven state-feedback design. *In Proceedings of 2020 American Control Conference (ACC)*. IEEE, p. 1532–1538.

Berberich, J., Köhler, J., Müller, M. A., and Allgöwer, F. (2020b). Data-driven model predictive control with stability and robustness guarantees. *IEEE Transactions on Automatic Control*, 66(4):1702–1717.

Berry, G. (2008). Synchronous design and verification of critical embedded systems using SCADE and Esterel. *Lecture Notes in Computer Science*, 4916:2–2.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag. ISBN: 0387310738.

Bisoffi, A. and Dimarogonas, D. V. (2018). A hybrid barrier certificate approach to satisfy linear temporal logic specifications. *In Proceedings of 2018 Annual American Control Conference (ACC)*. IEEE, p. 634–639.

Blanchini, F. and Miani, S. (2008). *Set-theoretic methods in control*. Springer.

Blute, R., Desharnais, J., Edalat, A., and Panangaden, P. (1997). Bisimulation for labelled Markov processes. *Proceedings of 12th Annual IEEE Symposium on Logic in Computer Science*:149–158.

Bogachev, V. I. (2007). *Measure theory*. Springer Science & Business Media.

Bogomolov, S., Forets, M., Frehse, G., Potomkin, K., and Schilling, C. (2019). JuliaReach: a toolbox for set-based reachability. *In Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, p. 39–44.

Bombara, G., Vasile, C.-I., Penedo, F., Yasuoka, H., and Belta, C. (2016). A decision tree approach to data classification using signal temporal logic. *In Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control (HSCC)*, p. 1–10.

Bowen, J. P. and Hinchey, M. G. (2005). Ten commandments revisited: a ten-year perspective on the industrial application of formal methods. *In Proceedings of the 10th international workshop on formal methods for industrial critical systems*, p. 8–16.

Boyd, S., El Ghaoui, L., Feron, E., and Balakrishnan, V. (1994). *Linear matrix inequalities in system and control theory*. SIAM.

Cámara, J., Girard, A., and Gössler, G. (2011a). Safety controller synthesis for switched systems using multi-scale symbolic models. *In Proceedings of the*

50th IEEE Conference on Decision and Control (CDC) and European Control Conference (ECC). IEEE, p. 520–525.

Cámara, J., Girard, A., and Gössler, G. (2011b). Synthesis of switching controllers using approximately bisimilar multiscale abstractions. *In Proceedings of the 14th international conference on Hybrid Systems: Computation and Control (HSCC)*, p. 191–200.

Cauchi, N. and Abate, A. (2018). Benchmarks for cyber-physical systems: A modular model library for building automation systems. *IFAC-PapersOnLine*, 51(16):49–54.

Cauchi, N. and Abate, A. (2019). StocHy-automated verification and synthesis of stochastic processes. *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*:258–259.

Cauchi, N., Laurenti, L., Lahijanian, M., Abate, A., Kwiatkowska, M., and Cardelli, L. (2019). Efficiency through uncertainty: scalable formal synthesis for stochastic hybrid systems. *In Proceedings of the 22nd international conference on Hybrid Systems: Computation and Control (HSCC)*, p. 240–251.

Chatterjee, K. and Doyen, L. (2016). Perfect-information stochastic games with generalized mean-payoff objectives. *In Proceedings of the ACM/IEEE Symposium on Logic in Computer Science*, p. 247–256.

Chatterjee, K. and Henzinger, T. A. (2012). A survey of stochastic ω-regular games. *Journal of Computer and System Sciences*, 78(2):394–413.

Chew, V. (1966). Confidence, prediction, and tolerance regions for the multivariate normal distribution. *Journal of the American Statistical Association*, 61(315):605–617.

Cohen, M. H., Belta, C., and Tron, R. (2022). Robust control barrier functions for nonlinear control systems with uncertainty: A duality-based approach. *In Proceedings of the 61st IEEE Conference on Decision and Control (CDC)*. IEEE, p. 174–179.

Coulson, J., Lygeros, J., and Dörfler, F. (2019). Data-enabled predictive control: In the shallows of the DeePC. *In Proceedings of 2019 18th European Control Conference (ECC)*. IEEE, p. 307–312.

De Giacomo, G. and Vardi, M. Y. (2013). Linear temporal logic and linear dynamic logic on finite traces. *In Proceedings of the 23rd international joint conference on Artificial Intelligence (IJCAI)*. Association for Computing Machinery, p. 854–860.

De Persis, C. and Tesi, P. (2019). Formulas for data-driven control: Stabilization, optimality, and robustness. *IEEE Transactions on Automatic Control*, 65(3):909–924.

Desharnais, J., Gupta, V., Jagadeesan, R., and Panangaden, P. (2003). Approximating labelled Markov processes. *Information and Computation*, 184(1):160–200.

Desharnais, J., Gupta, V., Jagadeesan, R., and Panangaden, P. (2004). Metrics for

labelled Markov processes. *Theoretical Computer Science*, 318(3):323–354.

Deshmukh, J. V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., and Seshia, S. A. (2017). Robust online monitoring of signal temporal logic. *Formal Methods in System Design*, 51:5–30.

Devonport, A., Saoud, A., and Arcak, M. (2021). Symbolic abstractions from data: A PAC learning approach. *In Proceedings of 60th IEEE Conference on Decision and Control (CDC)*. IEEE, p. 599–604.

Dijkstra, E. W. et al. (1970). Notes on structured programming.

Ding, J., Abate, A., and Tomlin, C. (2013). Optimal control of partially observable discrete time stochastic hybrid systems for safety specifications. *In Proceedings of 2013 American Control Conference*. IEEE, p. 6231–6236.

Dutreix, M., Huh, J., and Coogan, S. (2022). Abstraction-based synthesis for stochastic systems with omega-regular objectives. *Nonlinear Analysis: Hybrid Systems*, 45:101204.

Engelaar, M. H. W., Haesaert, S., and Lazar, M. (2023). Stochastic Model Predictive Control with Dynamic Chance Constraints. *arXiv preprint*.

Faulwasser, T., Ou, R., Pan, G., Schmitz, P., and Worthmann, K. (2023). Behavioral theory for stochastic systems? A data-driven journey from Willems to Wiener and back again. *Annual Reviews in Control*.

Gastin, P. and Oddoux, D. (2001). Fast LTL to Büchi automata translation. *In International Conference on Computer Aided Verification*. Springer, p. 53–65.

Ghose, A. (2000). Formal methods for requirements engineering. *In Proceedings International Symposium on Multimedia Software Engineering*. IEEE, p. 13–13.

Girard, A. and Gössler, G. (2020). Safety synthesis for incrementally stable switched systems using discretization-free multi-resolution abstractions. *Acta Informatica*, 57(1):245–269.

Girard, A. and Pappas, G. J. (2009). Hierarchical control system design using approximate simulation. *Automatica*, 45(2):566–571.

Girard, A. (2012). Controller synthesis for safety and reachability via approximate bisimulation. *Automatica*, 48(5):947–953.

Girard, A. and Pappas, G. J. (2007). Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5):782–798.

Girard, A. and Pappas, G. J. (2011). Approximate bisimulation: A bridge between computer science and control theory. *European Journal of Control*, 17(5-6):568–578.

Gottlieb, S. and Ketcheson, D. I. (2016). Time discretization techniques. *Handbook of Numerical Analysis*. Vol. 17. Elsevier, p. 549–583.

Gurobi Optimization, LLC (2023). Gurobi Optimizer Reference Manual. URL: https://www.gurobi.com.

Haesaert, S., Cauchi, N., and Abate, A. (2017a). Certified policy synthesis for general Markov decision processes: An application in building automation systems. *Performance Evaluation*, 117:75–103.

Haesaert, S., Soudjani, S. E. Z., and Abate, A. (2017b). Verification of general Markov decision processes by approximate similarity relations and policy refinement. *SIAM Journal on Control and Optimization*, 55(4):2333–2367.

Haesaert, S., Nilsson, P., Vasile, C. I., Thakker, R., Agha-mohammadi, A.-a., Ames, A. D., and Murray, R. M. (2018). Temporal Logic Control of POMDPs via Label-based Stochastic Simulation Relations. *IFAC-PapersOnLine*, 51(16):271–276.

Haesaert, S. and Soudjani, S. E. Z. (2020). Robust dynamic programming for temporal logic control of stochastic systems. *IEEE Transactions on Automatic Control*, 66(6):2496–2511.

Haesaert, S., Van den Hof, P. M., and Abate, A. (2017c). Data-driven and model-based verification via Bayesian identification and reachability analysis. *Automatica*, 79:115–126.

Haghighi, M. M. (2013). Controlling energy-efficient buildings in the context of smart grid: A cyber physical system approach. PhD thesis. University of California, Berkeley.

Hartmanns, A. and Hermanns, H. (2014). The Modest Toolset: An Integrated Environment for Quantitative Modelling and Verification. *In Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. vol. 8413. Lecture Notes in Computer Science. Springer, p. 593–598.

Henriksen, J. G., Jensen, J., Jørgensen, M., Klarlund, N., Paige, R., Rauhe, T., and Sandholm, A. (1995). Mona: Monadic second-order logic in practice. *In First International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Springer, p. 89–110.

Herceg, M., Kvasnica, M., Jones, C. N., and Morari, M. (2013). Multi-parametric toolbox 3.0. *In Proceedings of 2013 IEEE European Control Conference (ECC)*. `http://control.ee.ethz.ch/~mpt`. IEEE, p. 502–510.

Hinchey, M. G. and Bowen, J. P. (2012). *Industrial-strength formal methods in practice*. Springer Science & Business Media.

Hinchey, M. G., Jackson, M., Cousot, P., Cook, B., Bowen, J. P., and Margaria, T. (2008). Software engineering and formal methods. *Communications of the ACM*, 51(9):54–59.

Hollander, F. den (2012). *Probability theory: The coupling method (lecture notes)*. Mathematics Institute Leiden University, The Netherlands.

Hsu, K., Majumdar, R., Mallik, K., and Schmuck, A. (2018). Multi-layered abstraction-based controller synthesis for continuous-time systems. *In Proceedings of the 21st international conference on Hybrid Systems: Computation and Control*

(HSCC), p. 120–129.

Huang, C., Chen, X., Lin, W., Yang, Z., and Li, X. (2017). Probabilistic safety verification of stochastic hybrid systems using barrier certificates. *ACM Transactions on Embedded Computing Systems*, 16(5s):1–19.

Huijgevoort, B. C. van and Haesaert, S. (2022). Similarity quantification for linear stochastic systems: A coupling compensator approach. *Automatica*, 144:110476.

Hüls, J., Niehaus, H., and Remke, A. (2020). Hpnmg: A C++ Tool for Model Checking Hybrid Petri Nets with General Transitions. *In 12th International NASA Formal Methods Symposium, NFM 2020*. Springer.

Jagtap, P., Soudjani, S. E. Z., and Zamani, M. (2020). Formal synthesis of stochastic systems via control barrier certificates. *IEEE Transactions on Automatic Control*, 66(7):3097–3110.

Jaynes, E. T. and Kempthorne, O. (1976). Confidence intervals vs Bayesian intervals. *In Foundations of Probability Theory, Statistical Inference, and Statistical Theories of Science*. Springer, p. 175–257.

Johansson, M. (1999). Piecewise linear control systems. PhD thesis. Lund Institute of Technology, Sweden.

Julius, A. A. and Pappas, G. J. (2009). Approximations of stochastic hybrid systems. *IEEE Transactions on Automatic Control*, 54(6):1193–1203.

Kalagarla, K. C., Jain, R., and Nuzzo, P. (2021). Model-free reinforcement learning for optimal control of Markov decision processes under signal temporal logic specifications. *In Proceedings of 60th IEEE Conference on Decision and Control (CDC)*. IEEE, p. 2252–2257.

Kalra, N. and Paddock, S. M. (2016). Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193.

Kapoor, P., Balakrishnan, A., and Deshmukh, J. V. (2020). Model-based reinforcement learning from signal temporal logic specifications. *arXiv preprint*.

Kariotoglou, N., Kamgarpour, M., Summers, T. H., and Lygeros, J. (2017). The linear programming approach to reach-avoid problems for Markov decision processes. *Journal of Artificial Intelligence Research*, 60:263–285.

Kazemi, M., Majumdar, R., Salamati, M., Soudjani, S. E. Z., and Wooding, B. (2022). Data-driven abstraction-based control synthesis. *arXiv preprint*.

Kazemi, M. and Soudjani, S. E. Z. (2020). Formal policy synthesis for continuous-state systems via reinforcement learning. *In Integrated Formal Methods: 16th International Conference, IFM 2020*. Springer, p. 3–21.

Keesman, K. J. (2011). *System identification: An introduction*. Vol. 2. Springer.

Knapp, A. W. (2016). Chapter VI. Measure Theory for Euclidean Space. *Basic Real Analysis*. Project Euclid (distributor), p. 334–394.

Koch, A., Berberich, J., and Allgöwer, F. (2021). Provably robust verification of dissipativity properties from data. *IEEE Transactions on Automatic Control*, 67(8):4248–4255.

Kochdumper, N., Gruber, F., Schürmann, B., Gaßmann, V., Klischat, M., and Althoff, M. (2021). AROC: A toolbox for automated reachset optimal controller synthesis. *In Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control (HSCC)*, p. 1–6.

Koelewijn, P. J. W. (2023). Analysis and Control of Nonlinear Systems with Stability and Performance Guarantees: A Linear Parameter-Varying Approach.

Kong, Z., Jones, A., Medina Ayala, A., Aydin Gol, E., and Belta, C. (2014). Temporal logic inference for classification and prediction from data. *In Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (HSCC)*, p. 273–282.

Kreiker, J., Tarlecki, A., Vardi, M. Y., and Wilhelm, R. (2011). Modeling, analysis, and verification-the formal methods manifesto 2010 (dagstuhl perspectives workshop 10482). *Dagstuhl Manifestos*, 1(1).

Kupferman, O. and Vardi, M. Y. (2001). Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314.

Kushner, H. J. (1967). *Stochastic stability and control*. Tech. rep. Brown University Providence RI.

Kwiatkowska, M., Norman, G., and Parker, D. (2002). PRISM: Probabilistic symbolic model checker. *In International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*. Springer, p. 200–204.

Kwiatkowska, M., Norman, G., and Parker, D. (2005). Probabilistic model checking in practice: Case studies with PRISM. *ACM SIGMETRICS Performance Evaluation Review*, 32(4):16–21.

Kwiatkowska, M., Norman, G., and Parker, D. (2007). Stochastic model checking. *Formal Methods for Performance Evaluation: 7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2007*:220–270.

Labit, Y., Peaucelle, D., and Henrion, D. (2002). SeDuMi interface 1.02: a tool for solving LMI problems with SeDuMi. *In Proceedings of IEEE International Symposium on Computer Aided Control System Design*. IEEE, p. 272–277.

Lavaei, A., Khaled, M., Soudjani, S. E. Z., and Zamani, M. (2020a). AMYTISS: Parallelized automated controller synthesis for large-scale stochastic systems. *International Conference on Computer Aided Verification*:461–474.

Lavaei, A., Soudjani, S. E. Z., and Zamani, M. (2019). Compositional construction of infinite abstractions for networks of stochastic control systems. *Automatica*, 107:125–137. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2019.05.043.

Lavaei, A., Soudjani, S. E. Z., and Zamani, M. (2021). Compositional abstraction-

based synthesis of general MDPs via approximate probabilistic relations. *Nonlinear Analysis: Hybrid Systems*, 39:100991.

Lavaei, A., Somenzi, F., Soudjani, S. E. Z., Trivedi, A., and Zamani, M. (2020b). Formal controller synthesis for continuous-space MDPs via model-free reinforcement learning. *In Proceedings of 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, p. 98–107.

Lavaei, A., Soudjani, S. E. Z., and Zamani, M. (2020c). Compositional (in) finite abstractions for large-scale interconnected stochastic systems. *IEEE Transactions on Automatic Control*, 65(12):5280–5295.

Lavaei, A., Soudjani, S. E. Z., Abate, A., and Zamani, M. (2022a). Automated verification and synthesis of stochastic hybrid systems: A survey. *Automatica*, 146:110617.

Lavaei, A., Soudjani, S. E. Z., Frazzoli, E., and Zamani, M. (2022b). Constructing MDP abstractions using data with formal guarantees. *IEEE Control Systems Letters*, 7:460–465.

Lavaei, A., Soudjani, S. E. Z., Majumdar, R., and Zamani, M. (2017). Compositional abstractions of interconnected discrete-time stochastic control systems. *In Proceedings of 56th Annual Conference on Decision and Control (CDC)*. IEEE, p. 3551–3556.

Lee, E. A. and Seshia, S. A. (2016). *Introduction to embedded systems: A cyber-physical systems approach*. MIT Press.

Lesser, K. and Oishi, M. M. M. K. (2016). Approximate safety verification and control of partially observable stochastic hybrid systems. *IEEE Transactions on Automatic Control*, 62(1):81–96.

Lima, P. F., Pereira, G. C., Mårtensson, J., and Wahlberg, B. (2018). Experimental validation of model predictive control stability for autonomous driving. *Control Engineering Practice*, 81:244–255.

Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y., and Shi, W. (2019). Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716.

Lofberg, J. (2004). YALMIP: A toolbox for modeling and optimization in MATLAB. *In 2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*. IEEE, p. 284–289.

Lopez, B. T. and Slotine, J.-J. E. (2023). Unmatched control barrier functions: Certainty equivalence adaptive safety. *In Proceedings of 2023 American Control Conference (ACC)*. IEEE, p. 3662–3668.

Lou, D. (Nov. 2021). Parameterized Model Order Reduction with Applications to Thermal Systems. English. PhD thesis. Electrical Engineering. ISBN: 978-90-386-5402-7.

Majumdar, R., Mallik, K., Schmuck, A.-K., and Soudjani, S. E. Z. (2021). Symbolic qualitative control for stochastic systems via finite parity games. *IFAC-*

PapersOnLine, 54(5):127–132.

Majumdar, R., Mallik, K., and Soudjani, S. E. Z. (2020). Symbolic Controller Synthesis for Büchi Specifications on Stochastic Systems. *In Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control (HSCC)*. Association for Computing Machinery.

Makdesi, A., Girard, A., and Fribourg, L. (2021). Efficient data-driven abstraction of monotone systems with disturbances. *IFAC-PapersOnLine*, 54(5):49–54.

Maler, O. and Nickovic, D. (2004). Monitoring temporal properties of continuous signals. *In International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. Springer, p. 152–166.

Markovsky, I. and Dörfler, F. (2021). Behavioral systems theory in data-driven analysis, signal processing, and control. *Annual Reviews in Control*, 52:42–64.

Markovsky, I. and Rapisarda, P. (2007). On the linear quadratic data-driven control. *In Proceedings of 2007 European Control Conference (ECC)*. IEEE, p. 5313–5318.

Markovsky, I. and Rapisarda, P. (2008). Data-driven simulation and control. *International Journal of Control*, 81(12):1946–1959.

Mesbah, A. (2016). Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44.

Nejati, A., Soudjani, S. E. Z., and Zamani, M. (2020). Compositional Construction of Control Barrier Certificates for Large-Scale Stochastic Switched Systems. *IEEE Control Systems Letters*, 4(4):845–850.

Nejati, A., Soudjani, S. E. Z., and Zamani, M. (2022). Compositional construction of control barrier functions for continuous-time stochastic hybrid systems. *Automatica*, 145:110513.

Nilsson, P., Haesaert, S., Thakker, R., Otsu, K., Vasile, C.-I., Agha-Mohammadi, A.-A., Murray, R. M., and Ames, A. D. (2018). Toward specification-guided active mars exploration for cooperative robot teams.

Ou, R., Pan, G., and Faulwasser, T. (2022). Data-driven multiple shooting for stochastic optimal control. *IEEE Control Systems Letters*, 7:313–318.

Pan, G., Ou, R., and Faulwasser, T. (2022a). On a stochastic fundamental lemma and its use for data-driven optimal control. *IEEE Transactions on Automatic Control*.

Pan, G., Ou, R., and Faulwasser, T. (2022b). Towards data-driven stochastic predictive control. *arXiv preprint*.

Pannu, A. (2015). Artificial intelligence and its application in different areas. *Artificial Intelligence*, 4(10):79–84.

Papachristodoulou, A., Anderson, J., Valmorbida, G., Prajna, S., Seiler, P., Parrilo, P., Peet, M. M., and Jagt, D. (2013). SOSTOOLS version 4.00 sum of squares optimization toolbox for MATLAB. *arXiv preprint*.

Pappas, T., Laub, A., and Sandell, N. (1980). On the numerical solution of the discrete-time algebraic Riccati equation. *IEEE Transactions on Automatic Control*, 25(4):631–641.

Pearl, T. H. (2018). Hands on the wheel: a call for greater regulation of semi-autonomous cars. *Ind. LJ*, 93:713.

Pilch, C. and Remke, A. (2017). HYPEG: Statistical Model Checking for hybrid Petri nets: Tool Paper. *In Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*. VALUETOOLS 2017. Venice, Italy: ACM, p. 186–191.

Pnueli, A. (1977). The Temporal Logic of programs. *18th Annual Symposium on Foundations of Computer Science*:46–57.

Qi, S., Zhang, Z., Haesaert, S., and Sun, Z. (2023). Automated Formation Control Synthesis from Temporal Logic Specifications. *arXiv preprint*.

Rabin, M. O. and Scott, D. (1959). Finite automata and their decision problems. *IBM journal of research and development*, 3(2):114–125.

Raman, V., Donzé, A., Maasoumy, M., Murray, R. M., Sangiovanni-Vincentelli, A., and Seshia, S. A. (2014). Model predictive control with signal temporal logic specifications. *In Proceedings of 53rd IEEE Conference on Decision and Control (CDC)*. IEEE, p. 81–87.

Rausand, M. (2014). *Reliability of safety-critical systems: theory and applications*. John Wiley & Sons.

Ren, W. and Dimarogonas, D. V. (2019). Dynamic Quantization based Symbolic Abstractions for Nonlinear Control Systems. *In Proceedings of the 58th IEEE Conference on Decision and Control (CDC)*. IEEE, p. 4343–4348.

Rodrigues, L. and Boyd, S. (2005). Piecewise-affine state feedback for piecewise-affine slab systems using convex optimization. *Systems & Control Letters*, 54(9):835–853.

Rodrigues, L. and How, J. P. (2003). Synthesis of piecewise-affine controllers for stabilization of nonlinear systems. *In Proceedings of the 42nd IEEE International Conference on Decision and Control (CDC)*. vol. 3. IEEE, p. 2071–2076.

Romer, A., Berberich, J., Köhler, J., and Allgöwer, F. (2019). One-shot verification of dissipativity properties from input–output data. *IEEE Control Systems Letters*, 3(3):709–714.

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 (HSCC)*, p. 99–104.

Santoyo, C., Dutreix, M., and Coogan, S. (2021). A barrier function approach to finite-time stochastic system verification and control. *Automatica*, 125:109439.

Sarhadi, P. and Yousefpour, S. (2015). State of the art: hardware in the loop modeling and simulation with its applications in design, development and imple-

mentation of system and control software. *International Journal of Dynamics and Control*, 3:470–479.

Schupp, S., Ábrahám, E., Chen, X., Ben Makhlouf, I., Frehse, G., Sankaranarayanan, S., and Kowalewski, S. (2015). Current challenges in the verification of hybrid systems. *In Cyber Physical Systems. Design, Modeling, and Evaluation: 5th International Workshop, CyPhy 2015*. Springer, p. 8–24.

Schuppan, V., Latvala, T., Junttila, T., Heljanko, K., and Biere, A. (2006). Linear encodings of bounded LTL model checking. *Logical Methods in Computer Science*, 2.

Schwarting, W., Alonso-Mora, J., and Rus, D. (2018). Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210.

Segala, R. and Lynch, N. (1994). Probabilistic simulations for probabilistic processes. *International Conference on Concurrency Theory*:481–496.

Sheng, S., Pakdamanian, E., Han, K., Kim, B., Tiwari, P., Kim, I., and Feng, L. (2019). A case study of trust on autonomous driving. *In Proceedings of 2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, p. 4368–4373.

Shmarov, F. and Zuliani, P. (2015). ProbReach: Verified Probabilistic δ-Reachability for Stochastic Hybrid Systems. *In Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control (HSCC)*. ACM, p. 134–139.

Skjetne, R. and Egeland, O. (2006). Hardware-in-the-loop testing of marine control system.

Soudjani, S. E. Z. 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.

Soudjani, S. E. Z., Gevaerts, C., and Abate, A. (2015). FAUST^2: Formal Abstractions of Uncountable-STate STochastic Processes. *TACAS*. LNCS:272–286.

Soudjani, S. E. Z. and Abate, A. (2013b). Probabilistic reach-avoid computation for partially degenerate stochastic processes. *IEEE Transactions on Automatic Control*, 59(2):528–534.

Steinhardt, J. and Tedrake, R. (2012). Finite-time regional verification of stochastic non-linear systems. *The International Journal of Robotics Research*, 31(7):901–923.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tabuada, P. (2008). An approximate simulation approach to symbolic control. *IEEE Transactions on Automatic Control*, 53(6):1406–1418.

Tabuada, P. (2009). *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media.

Tazaki, Y. and Imura, J.-i. (2010). Approximately bisimilar discrete abstractions of nonlinear systems using variable-resolution quantizers. *In Proceedings of the 2010 American Control Conference (ACC)*. IEEE, p. 1015–1020.

Thorpe, A. J., Ortiz, K. R., and Oishi, M. M. M. K. (2021). SReachTools Kernel Module: Data-Driven Stochastic Reachability Using Hilbert Space Embeddings of Distributions. *In Proceedings of 60th IEEE Conference on Decision and Control (CDC)*. IEEE, p. 5073–5079.

Tkachev, I. and Abate, A. (2014). On approximation metrics for linear temporal model-checking of stochastic systems. *Proceedings of the 17th international conference on Hybrid systems: Computation and Control (HSCC)*:193–202.

Tkachev, I., Mereacre, A., Katoen, J. P., and Abate, A. (2013). Quantitative automata-based controller synthesis for non-autonomous stochastic hybrid systems. *In Proceedings of 16th international conference on Hybrid Systems: Computation and Control (HSCC)*, p. 293–302.

Tkachev, I. and Abate, A. (2011). On infinite-horizon probabilistic properties and stochastic bisimulation functions. *In Proceedings of 50th IEEE Conference on Decision and Control (CDC) and European Control Conference (ECC)*. IEEE, p. 526–531.

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.

Tóth, R. (2010). *Modeling and identification of linear parameter-varying systems*. Vol. 403. Springer.

Van Den Hof, P. M. and Schrama, R. J. (1995). Identification and control–closed-loop issues. *Automatica*, 31(12):1751–1770.

Van Waarde, H. J., Camlibel, M. K., Rapisarda, P., and Trentelman, H. L. (2022). Data-driven dissipativity analysis: application of the matrix S-lemma. *IEEE Control Systems Magazine*, 42(3):140–149.

Verhoek, C., Abbas, H. S., Tóth, R., and Haesaert, S. (2021a). Data-driven predictive control for linear parameter-varying systems. *IFAC-PapersOnLine*, 54(8):101–108.

Verhoek, C., Tóth, R., and Abbas, H. S. (2022). Direct Data-Driven State-Feedback Control of Linear Parameter-Varying Systems. *Submitted to IEEE Transactions on Automatic Control*:arXiv preprint.

Verhoek, C., Tóth, R., Haesaert, S., and Koch, A. (2021b). Fundamental lemma for data-driven analysis of linear parameter-varying systems. *In Proceedings of the 60th IEEE Conference on Decision and Control (CDC)*. IEEE, p. 5040–5046.

Vinod, A., Gleason, J., and Oishi, M. M. K. (2019). SReachTools: A MATLAB stochastic reachability toolbox. *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*:33–38.

Wang, X., Nair, S., and Althoff, M. (2020). Falsification-based robust adversarial

reinforcement learning. *In Proceedings of 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, p. 205–212.

Weisstein, E. W. (2014). Taylor's Inequality. https://mathworld.wolfram.com/.

Wells, A. M., Lahijanian, M., Kavraki, L. E., and Vardi, M. Y. (2020). LTLf synthesis on probabilistic systems. *arXiv preprint*.

Wieland, P. and Allgöwer, F. (2007). Constructive safety using control barrier functions. *IFAC Proceedings Volumes*, 40(12):462–467.

Willems, J. C. and Polderman, J. W. (1997). *Introduction to mathematical systems theory: a behavioral approach*. Vol. 26. Springer Science & Business Media.

Willems, J. C., Rapisarda, P., Markovsky, I., and De Moor, B. L. (2005). A note on persistency of excitation. *Systems & Control Letters*, 54(4):325–329.

Wolff, E. M., Topcu, U., and Murray, R. M. (2014). Optimization-based control of nonlinear systems with linear temporal logic specifications. *In Proceedings of International Conference on Robotics and Automation (ICRA)*, p. 5319–5325.

Wongpiromsarn, T., Topcu, U., and Lamperski, A. (2015). Automata theory meets barrier certificates: Temporal logic verification of nonlinear systems. *IEEE Transactions on Automatic Control*, 61(11):3344–3355.

Woodcock, J., Larsen, P. G., Bicarregui, J., and Fitzgerald, J. (2009). Formal methods: Practice and experience. *ACM computing surveys (CSUR)*, 41(4):1–36.

Yates, R. D. and Goodman, D. J. (1999). Probability and stochastic processes. *John Willey & Sons*.

Zamani, M., Esfahani, P. M., Majumdar, R., Abate, A., and Lygeros, J. (2014). Symbolic control of stochastic systems via approximately bisimilar finite abstractions. *IEEE Transactions on Automatic Control*, 59(12):3135–3150.

Zhang, S., Deng, W., Zhao, Q., Sun, H., and Litkouhi, B. (2013). Dynamic trajectory planning for vehicle autonomous driving. *In 2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, p. 4161–4166.

Zhong, B., Lavaei, A., Zamani, M., and Caccamo, M. (2023a). Automata-based controller synthesis for stochastic systems: A game framework via approximate probabilistic relations. *Automatica*, 147:110696.

Zhong, B., Zamani, M., and Caccamo, M. (2023b). Formal Synthesis of Controllers for Uncertain Linear Systems against ω-Regular Properties: A Set-based Approach. *IEEE Transactions on Automatic Control*.

Zugaj, M. and Narkiewicz, J. (2009). Autopilot for reconfigurable flight control system. *Journal of Aerospace Engineering*, 22(1):78–84.

List of abbreviations

AI	Artificial Intelligence
CBC	Control Barrier Certificate
cf.	Compare (confer or conferatur)
DB	Discretization-Based
DF	Discretization-Free
DFA	Deterministic Finite-state Automaton
DP	Dynamic Programming
e.g.	For example (exempli gratia)
et al.	And others (et alia)
GB	Gigabyte
GHz	Gigahertz
gMDP	General Markov Decision Process
i.e.	That is (id est)
i.i.d	Independently and Identically Distributed (when describing a noise signal)
LMI	Linear Matrix Inequality
LTL	Linear Temporal Logic
LTI	Linear Time Invariant
MB	Megabyte
MDP	Markov Decision Process
MHz	Mehahertz
nD	n-Dimensional
PWA	Piecewise-Affine
s, sec	Seconds
scLTL	Syntactically co-safe Linear Temporal Logic
s.t.	Subject To or Such That
SSR	$(\epsilon, \delta)-$sub-simulation relation
STL	Signal Temporal Logic

List of symbols

General

$\mathbf{d}_{\mathbb{Y}}$	Metric on the space $\mathbb{Y}$
$\mathbf{1}_Q$	Indicator function for set Q
$vert(P)$	Vertices of polytope P
$\mathcal{B}(\mathbb{X})$	Borel σ- algebra on space $\mathbb{X}$
β	Deviation caused by state mapping from a continuous-state space to a finite-state space
$\mathcal{D}$	Data consisting of data points
$(1-\alpha)$	Confidence bound
$\delta_a(A)$	Dirac measure concentrated at a point a

Coupling compensators

$\mathcal{W}$	Coupling compensator as a probability measure that couples the probability measures of the disturbances (c.f. Def. 2.2)
$\bar{\mathcal{W}}$	Coupling compensator as a Borel measurable stochastic kernel over the probability measures of disturbances
$\mathcal{W}^x$	Coupled probability measure that couples the probability measures of states
$\bar{\mathcal{W}}^x$	Coupled Borel measurable stochastic kernel over the probability measures of states
$\mathcal{W}^x_{sub}$	Sub-probability coupling that couples the probability measures of the states
$\bar{\mathcal{W}}^x_{sub}$	Sub-probability coupling as a Borel measurable stochastic kernel over the probability measures of states

Models and automata

$\mathbf{M}$	Mathematical model
$\hat{\mathbf{M}}$	Abstract model (finite-state)
$\hat{\mathbf{M}}\|\|\mathbf{M}$	Composed model as in Def. 2.3
$\mathbf{M}(\theta)$	Mathematical model with parametric uncertainty
$\widetilde{\mathbf{M}}$	Abstract model (nominal parameter)
$\mathbf{M}_w$	Abstract model (waypoint model)

$\mathcal{A}_\phi$	Deterministic finite-state automaton with respect to specification ϕ
$\tau_{\mathcal{A}}$	Transition function of automaton $\mathcal{A}$

Control

$\boldsymbol{\mu}$	Control policy
$\mathbf{C}$	Controller
$\hat{\mathbf{C}}$	Abstract (finite-state) controller
$\widetilde{\mathbf{C}}$	Abstract (nominal) controller
$\mathbf{M} \times \mathbf{C}$	Controlled systems
$\mathcal{U}_v$	Interface function
$\mathcal{X}_v$	State mapping

Relations

$\mathcal{R}$	Measurable relation
$\boldsymbol{\mathcal{R}}$	Multi-layered simulation relation
$\preceq_\epsilon^\delta$	(ϵ, δ)-stochastic simulation relation, in its most basic form defined in Def. 2.4
$\preceq_\epsilon^\delta$	(ϵ, δ)-sub-simulation relation
ϵ	Output deviation or precision (scalar)
$\boldsymbol{\epsilon}$	Output deviation or precision (matrix)
δ	Probability deviation or confidence (scalar)
$\boldsymbol{\delta}$	Probability deviation or confidence (matrix)
$\boldsymbol{\delta}(\cdot)$	Probability deviation or confidence (function)

Dynamic programming

$V(\cdot)$	Value function
$\mathbf{T}(\cdot)$	Bellman operator
$\mathbf{L}(\cdot)$	Truncation function

Probability and statistics

$\mathbb{P}$	Probability measure
$\mathcal{P}(\mathbb{X})$	Set of probability measures on the measurable space $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$
$\mathbb{P}(\mathbf{M} \times \mathbf{C} \models \phi)$	Satisfaction probability, i.e. the probability that the controlled systems satisfies the specification ϕ
$\mathrm{cdf}(\cdot)$	Cumulative distribution function of a one-dimensional standard Gaussian distribution
$\mathrm{idf}(\cdot)$	Inverse distribution function of a one-dimensional standard Gaussian distribution
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean μ and co-variance matrix Σ
$\mathcal{N}(\cdot \vert \mu, \Sigma)$	Gaussian stochastic kernel with mean μ and co-variance matrix Σ
$\chi(\cdot \mid v)$	Chi-squared distribution with v degrees of freedom
$\mathbf{t}(\cdot \mid \cdot)$	Probability kernel describing a state transition
$\mathbb{E}(\cdot)$	Expected value operator

Spaces

$\mathbb{X}$	State space
$(\mathbb{X}, \mathcal{B}(\mathbb{X}))$	Borel measurable space
$\mathbb{U}$	Input space
$\mathbb{Y}$	Output space
$\mathbb{W}$	Disturbance space
$\hat{\mathbb{X}}, \hat{\mathbb{U}}, \hat{\mathbb{Y}}$	State, input, and output space of abstract model $\hat{\mathbf{M}}$
$\tilde{\mathbb{X}}, \tilde{\mathbb{U}}$	State, and input space of abstract model $\widetilde{\mathbf{M}}$
$\mathbb{X}_w$	State space of abstract model $\mathbf{M}_w$

Signal, trajectories and variables

x	State associated with state-space representation
u	Input
w	Disturbance
y	Output
n_x	Dimension of variable x
$x(t+1) = x_{t+1} = x^+$	Time update of signal x
$x(0) = x_0$	initial state
$\hat{x}, \hat{u}, \hat{w}, \hat{y}$	State, input, disturbance, and output associated with abstract model $\hat{\mathbf{M}}$
$\tilde{x}, \tilde{u}, \tilde{w}, \tilde{y}$	State, input, disturbance, and output associated with abstract model $\widetilde{\mathbf{M}}$
x_w, y_w	State and output associated with abstract model $\mathbf{M}_w$
q	State of DFA $\mathcal{A}$
$\boldsymbol{x}$	State trajectory
$\mathbf{u}$	Input trajectory
$\mathbf{y}$	Output trajectory
$\boldsymbol{z}$	Infinite sequence of signals z_t starting at $t = 0$, that is, $\boldsymbol{z} = \{z_0, z_1, \dots\}$
$\boldsymbol{z}_t$	Infinite sequence of signals z_t, that is, $\boldsymbol{z} = \{z_t, z_{t+1}, \dots\}$
$\boldsymbol{z}_{[0,N]}$	Finite sequence of signals z_t, that is, $\boldsymbol{z}_{[0,N]} = \{z_0, z_1, \dots, z_N\}$
θ	Vector consisting of the unknown parameters of model $\mathbf{M}(\theta)$
θ^*	True parameters
$\tilde{\theta}$	Estimated parameters

Set theory

$x \in \mathbb{X}$	Set membership, that is x belongs to $\mathbb{X}$
$\subseteq, (\subset)$	(strict) subset
$A \times B$	Cartesian product of sets A and B
$A \oplus B$	Minkowski sum of two sets A and B
$A \cup B$	Union of two sets A and B
$\bigcup_i A_i$	Union of a finite number of sets A_i
$A \cap B$	Intersection of two sets A and B

$\emptyset$	Empty set
$\mathbb{R}^+$	Set of positive real numbers
$\mathbb{N}$	Set of natural numbers (excluding zero)
$\mathbb{Z}$	Set of integers

Temporal logic

ϕ	co-safe temporal logic specification
AP	Set of atomic propositions
p_i	Atomic proposition in AP
2^{AP}	Alphabet corresponding to atomic propositions AP
π	Letter (in the alphabet, that is $\pi \in 2^{\text{AP}}$)
$\boldsymbol{\pi}$	Word (consists of letters)
L	Labeling function assigning letters to outputs
$\neg$	Negation
$\wedge$	And operator
$\vee$	Or operator
$\bigcirc$	Next operator
U	Until operator
$\Box$	Always operator
$\Diamond$	Eventually operator
true	True

Linear algebra and calculus

I_n	n-dimensional identity matrix
$\|\|x\|\|$	Two-norm
$\|\|x\|\|_D$	Weighted two-norm
$\|u\|$	Absolute value of scalar u
$x^\top$	Transpose of x
A^{-1}	Inverse of matrix A
$\det(A)$	Determinant of matrix A
$A \otimes B$	Kronecker product of matrices A and B
$\nabla f(x)$	Gradient of function $f(x)$

Sets

$\mathbb{R}$	Set of real numbers
$\mathbb{R}^+$	Set of positive real numbers
$\mathbb{N}$	Set of natural numbers (excluding zero)
$\mathbb{Z}$	Set of integers
Θ	Credible set (c.f. Def. 5.2)
$\mathcal{E}(x)$	Ellipsoidal set centered around x
$\mathbb{T}$	Time-axis in the behavioral framework

Behaviors

$\boldsymbol{\Sigma}$	Dynamical system
$\mathfrak{B}$	Behavior of a dynamical system
$\mathfrak{B}\|_A$	Behavior of a dynamical system restricted to a specific subset A

General mathematics

min	Minimum
max	Maximum
arg	Argument of
inf	Infimum
sup	Supremum
$\geq, (>)$	(strict) Inequality
$\succ, (\succeq)$	Positive (semi-)definite
$\prec, (\preceq)$	Negative (semi-) definite
$\forall$	For all
$\implies$	Implication
$\equiv$	Equivalence
$:=$	Equal by definition
$::=$	Recursively defined
$\approx$	Approximate equality

Publiekssamenvatting

Automatische Synthese van Regelaars met Temporele Logica Eisen

Stochastische, Onzekere, en Niet-Lineaire Systemen

De invloed die technische systemen hebben op het dagelijks leven is aan het groeien. Ze spelen een steeds grotere rol, worden *slimmer* en nemen meer beslissingen *zelf*. Hierdoor is ook de *betrouwbaarheid* van deze hightech systemen steeds belangrijker. Dit geldt vooral voor systemen die in een situatie gebruikt worden waarbij de veiligheid kritiek is. Bijvoorbeeld systemen die direct in contact komen met mensen.

In de praktijk is het niet mogelijk om ervoor te zorgen dat er *nooit* wat fout gaat, maar we kunnen wel fouten zo veel mogelijk *voorkomen*. In deze thesis kijk ik naar het verbeteren van de betrouwbaarheid van veiligheidskritische systemen door garanties te geven op het gedrag. Ik maak hierbij gebruik van het onderzoeksgebied van *automatische, correct-door-ontwerp regelaar synthese* voor systemen met een continu toestandsruimte en ik ontwikkel formele methodes om regelaars te synthetiseren waarvoor je kunt bewijzen dat het regelsysteem aan temporele logica eisen voldoet.

Als we kijken naar het gedrag van veiligheidskritische systemen, dan krijgen we te maken met meerdere *onzekerheden*, zoals bijvoorbeeld een stochastische invloed op de dynamica. Het is cruciaal om deze onzekerheden mee te nemen in het automatisch synthetiseren van regelaars en op die manier de kans te berekenen dat een veiligheidskritisch systeem gewenst gedrag vertoond. Indien de onzekerheden genegeerd worden is het onmogelijk om accurate garanties te geven en blijft de betrouwbaarheid van dit soort systemen te laag.

Om om te gaan met deze onzekerheden, ontwikkel ik een formele methode met bijbehorend `MATLAB` programma voor stochastische systemen. Via deze methode is het mogelijk om het verschil in gedrag van twee modellen te kwantificeren via 1) het verschil in de uitkomst/observaties en 2) het verschil in de probabilistische transities. Deze methode zorgt zowel voor een efficiënte berekening, als een accurate berekening van de kans op gewenst gedrag. Hierna breid ik deze formele methode uit in twee richtingen. Eerst verbeter ik de schaalbaarheid en rekensnelheid van deze methode door verschillende aanpakken te combineren. Hierdoor hoeven we alleen een langzame, moeilijke, maar wel precieze berekening te doen waar nodig en kunnen we een minder precieze, maar wel snelle en makkelijkere berekening gebruiken waar

mogelijk. In een internationaal samenwerkingsverband met Newcastle University (Verenigd Koninkrijk), heb ik deze methode uitgebreid door modelonzekerheid mee te nemen. We gaan dan uit van een stochastisch model waarvan de parameters onbekend zijn. Door data van het systeem te gebruiken, kunnen we een schatting maken van de verzameling waarin deze parameters zitten. Deze onzekerheid nemen we vervolgens mee in het automatisch synthetiseren van onze regelaar en in de berekening van de kans op gewenst gedrag. Daarna ga ik nog een stapje verder door ervan uit te gaan dat ons gehele systeem onbekend is. In plaats van een model te gebruiken, synthetiseer ik direct aan de hand van data een regelaar. Deze eerste stap maak ik voor deterministische systemen en laat de stochastiek dus even buiten beschouwing.

Concluderend, in mijn thesis ontwikkel ik formele methodes voor het automatisch synthetiseren van regelaars waarbij verschillende onzekerheden mee worden genomen. Als we deze methodes blijven verbeteren, dan kunnen we uiteindelijk garanties geven op het gedrag van autonome systemen in veiligheidskritische omstandigheden. Hierdoor maken we deze technologische ontwikkeling niet alleen mogelijk, maar ook betrouwbaar.

Acknowledgement

My journey as a PhD student has been influenced by the people around me and I will try to acknowledge and thank all of them for their presence and support.

Sofie, I like to think that we started this journey together, me as a PhD student and you as a PhD supervisor for the first time. The beginning of the journey was a bit of a struggle, but together we found a way to work together in a very efficient manner. I am very grateful for how easily I could reach you and how quickly you came up with (multiple) solutions to my problems. I admire your mathematical knowledge, programming skills, and the way you can think multiple steps ahead. You helped me with writing in a convincing manner and emphasizing the importance of my research, for which I am very grateful. I particularly enjoyed setting up and teaching (part of) the CPES course with you.

Next, I would like to thank Siep. You started supervising me during my internship and my graduation project, and then you agreed to be the Promotor of my PhD project. In total, I enjoyed your supervision for 6 years! You have the gift of conveying your enthusiasm to others and I always felt inspired and motivated after our meetings. I consider this as one of the most amazing characteristics that you have. Next to, obviously, being an amazing mathematician with a broad knowledge of control theory in general.

Besides having Sofie as my supervisor and Siep as my promotor, Sadegh also supervised me for a couple of months. Furthermore, I worked closely with him and his PhD student Oliver during the last couple of years of my PhD. Sadegh, you are a talented and kind supervisor with an inspiring ability to explain the most complicated things in an understandable manner. Thank you for the interesting discussions we had during my PhD and for your involvement in my research that led to this thesis.

I would also like to thank the members of my defense committee, Erika Ábrahám, Dimos Dimarogonas, Michel Reniers, and Sadegh Soudjani. I would like to thank all of you for reviewing my thesis and providing me with helpful suggestions.

Next, I would like to thank some people I have collaborated with over the years. Chris, thank you for your insight while writing a paper together. Oliver, I am so glad Sofie and Sadegh decided to put us together. You are a very kind person and the fact that you never stop asking questions is an important characteristic of a

PhD student. Also, thanks to you and Ben my visit to Newcastle University was very nice.

Next, I would like to thank everybody in the Control Systems group. Especially, Feye, Mannes, Patrick, and Tom with whom I did multiple DISC courses. Lizan, I really enjoyed helping out with the Mathematics II course together. Carlos and Ilja, thanks for being amazing neighbors and always being open for a chat. Clarisse, you have become a very close friend and I hope we will continue having our coffee/tea and cake moments occasionally. I would also like to thank all the PhDs who welcomed me back into the group after working from home for a very long time due to the coronavirus pandemic and maternity leave. Besides that, I would like to thank my colleagues from the FM4Control group. We started with very few people, but have grown substantially over the years. I am grateful for our interesting discussions and I was very happy to finally have other PhD students interested in working with formal methods.

I would also like to thank my friends. Especially, Lisa and Niels. Lisa, thanks for the relaxing walks and chats. Niels, you have been my friend since we joined the board of e.t.s.v. Thor (study Association of Electrical Engineering) together, more than 9 years ago and I am very glad that we both started our PhD journey in 2018. It was very refreshing to discuss the struggles and joys of doing a PhD while enjoying a “chocomel”.

Lastly, I would like to thank my family and friends for always supporting me. Mum and Dad, thank you for supporting me throughout my studies, for believing in me, and for offering advice when life gets tough. Rob, I could not have done this without you. My PhD has been a rollercoaster of emotions and you helped me through the most difficult times. Thanks for helping me with (or completely doing parts of) programming tasks, reviewing emails, practicing my presentations, and listening to me complain about something. I love you. You and Hugo mean everything to me.

About the author

Birgit van Huijgevoort was born on 10 August 1994, in Breda, The Netherlands. She completed high school in 2012 at the Cambreur College, Dongen. She received her Bachelor's degree in Automotive and her Master's degree in Systems and Control from the Eindhoven University of Technology, both Cum Laude, in 2016 and 2018 respectively.

During her Bachelor's degree, she spent a year as President of the Study Association of Electrical Engineering, *Thor*, and a year as chief of the electrical engineering team at *Team FAST*. During her Master's degree, she spent three months at Canterbury University, Christchurch, New Zealand, working on parameter tuning of snake robots. For her Master's thesis, she worked on the topic of structure-preserving discretization of port-Hamiltonian distributed parameter systems under the supervision of Prof. Dr. Siep Weiland and Prof. Dr. Hans Zwart.

In 2018, she started her PhD project at the Control Systems group at the Eindhoven University of Technology under the supervision of Dr. Ir. Sofie Haesaert and Prof. Dr. Siep Wieland. The topic of this research was Formal methods for uncertain cyber-physical systems. During her PhD project, she also followed graduate courses at the Dutch Institute for Systems and Control (DISC) and received her DISC certificate. In 2021, she and her husband, Rob Sanders, welcomed their son Hugo.