UCLouvain
Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

Abstraction Relations over Dynamical Systems: Theory and Algorithms

Julien Calbert

Thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor in Applied Sciences

Dissertation committee:

Prof. Raphaël Jungers (UCLouvain, *Advisor*)
Prof. Julien Hendrickx (UCLouvain, *Reader*)
Prof. Antoine Girard (Université Paris-Saclay, *Reader*)
Prof. Majid Zamani (University of Colorado Boulder, *Reader*)
Prof. Pierre-Antoine Absil (UCLouvain, *Chair*)

Version of November 2024.

I am a part of all that I have met;
Yet all experience is an arch wherethrough
Gleams that untravelled world, whose margin fades
For ever and for ever when I move.

Alfred Tennyson, Ulysses

Acknowledgments

Writing this thesis has been a challenging yet incredibly rewarding journey, and I am deeply grateful to those who have supported me throughout this process. Their encouragement, insights, and guidance have been invaluable in helping me reach this milestone.

First of all, I would like to thank my mother, Sylvie, for her unwavering love, which has without a doubt been the primary force bringing me to where I am today.

I am also profoundly grateful to my advisor, Raphaël Jungers, for his support over the past four years. His dedication, responsiveness, and insightful advice have been essential to my progress. I also thank him for creating and seizing every opportunity that came our way. I greatly appreciated the experience of being a teaching assistant for "Linear Algebra" and "Matrix Computations" courses, and I am thankful for the trust he placed in me, allowing me to deliver some lectures.

Alongside Raphaël, I extend my gratitude to the members of my thesis committee, Prof. Pierre-Antoine Absil, Prof. Julien Hendrickx, Prof. Majid Zamani, and especially Prof. Antoine Girard, for the warm and intense visit I experienced at L2S, which I thoroughly appreciated. I am sincerely honored to have had such a distinguished committee for my thesis.

I would also like to warmly thank the colleagues I had the opportunity to collaborate with, especially Benoît Legat and Lucas Egidio. Every discussion with them has been of inestimable value, greatly aiding me in completing my thesis successfully.

The exceptional support of the administrative and technical staff—Marie-Christine, Pascale, Astrid, Marine, François, and John—has been invaluable. I am truly grateful for their constant availability and willingness to assist us.

The friendships and connections I've built with everyone in the Euler building have sincerely been the most rewarding part of these years.

I leave with wonderful memories—from (far too many) coffee breaks in our office "Cafet 2.0," to countless afterworks, research retreats, (more than enough) sports activities, and all the events we shared together. For all this, I would like to thank the many doctoral students and postdoctoral fellows I had the pleasure of meeting at Euler: Adrien, Alexis, Anne, Antoine A., Antoine Q., Astrid, Bastien, Benoît, Charles, Clémence, Dânâ, Erwan, Florentin, Guillaume B., John, Lucas, Martina, Matteo, Michel, Nizar, Olivier, Philémon, Rémi, Renato, Sébastien C., Sébastien M., Simon M., Simon V., Sofiane, Thomas, Thiago, Virginie, Wouter, Yassine, and Zheming. Seeing you each day has been one of my main motivations to come into the office, and I am truly happy to continue in this working environment a little longer, especially within the warm atmosphere of the ground floor.

I would especially like to thank Rémi, with whom I spent most of my time as office mates and who was also my adventure companion in the desert; Yassine, with whom we initiated countless afterworks; Renato, with whom I ran my first marathon in Athens; and Virginie, for her unfailing support. Thank you for all those endless discussions philosophizing about the world, the countless moments of laughter, and for becoming such close friends.

I am fortunate to have many good friends who supported me throughout this thesis. I am especially grateful to Charlotte for her encouragement during the intense six-week writing rush.

Finally, I would like to thank my family for their unwavering support and for providing me with a warm environment in which I had the opportunity to grow. I am especially grateful to my grandfather Jacques, who inspired my early curiosity for science and history, and to my brother Simon, for the enriching conversations that have further nurtured my interest in science.

Abstract

The increasing complexity of modern control systems—spanning applications from smart grids to autonomous vehicles—has underscored the limitations of traditional control techniques. These cyber-physical systems (CPS) require advanced frameworks to ensure both efficiency and safety. This thesis explores abstraction-based control as a promising approach for CPS, where system behaviors are captured by finite-state models, transforming control tasks into combinatorial problems that enable systematic, correct-by-design controller synthesis.

First, we develop a theoretical framework to systematically classify and characterize system relations according to their concretization procedure. We demonstrate how the abstraction relation can be tailored based on the desired properties of the control architecture of the original system, irrespective of the specification we aim to enforce. Meanwhile, we introduce key system relations, highlighting their impact on control design and their role in mitigating the inherent non-determinism caused by discretization.

Second, we introduce algorithms to co-design the abstraction and abstract controller, focusing on relevant regions of the state space specific to the control task and thereby reducing computational complexity. For that, we integrate dynamic programming techniques within the abstraction framework, providing methods to transfer value functions from the abstract system to the original system, and we propose a framework for constructing non-uniform cells optimized for the system dynamics and control objectives. Finally, we implement these algorithms in `Dionysos.jl`, a modular package that solves control problems using advanced techniques from symbolic control, optimization, and learning.

Contents

III Algorithms

Introduction

In our modern world, the control systems are increasingly complex (think of smart grids, autonomous cars, robots and the internet of things). These complex systems are at the center of a paradigm shift coined as *the cyber-physical revolution* by the industrial and academic communities [KK12, Alu15, LS16]. The industry currently applies classical control techniques even if their requirements are no longer met for these systems. This causes a loss of efficiency and a lack of guarantees that are crucial in view of the importance of safety in such systems [BK08]. As a consequence, many key technological applications run nowadays at sub-optimal regimes.

Generalizing classical control techniques based on frequency analysis or convex optimization to cyber-physical systems (CPS), such as hybrid systems [HKPV95], is challenging. In the literature, some works achieve this for specific classes of hybrid systems but this is usually limited to systems for which the discrete part is not too complex [Cla97]. However, the increasing interconnection between computers and dynamical systems demands more general frameworks, which should be able to match the current challenges of these cyber-physical systems in a seamless way. For example, a common method to test the correct behavior of a system is through simulation [Don07, DM07, GP06]. The key advantage of simulation is its ability to potentially produce a counterexample, i.e., a trajectory that reaches an undesired set, thereby proving that the system is unsafe. However, if no counterexample is found, it does not guarantee the system's safety, as there are infinitely many possible trajectories due to uncertainties in initial states, inputs, and parameters. Formal verification frameworks have been developed using barrier certificates [Pra06, PJ04] that system trajectories cannot cross, or using reachability analysis [Alt10] to identify potential states that a system can reach. Reachability analysis involves

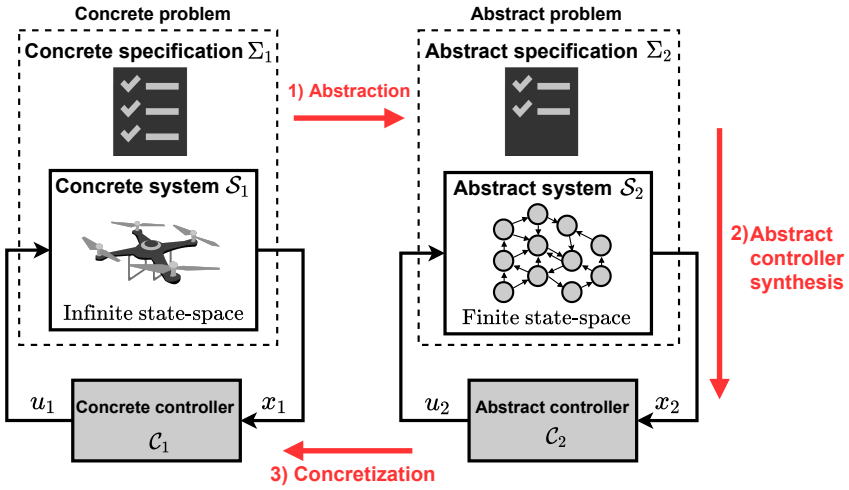


Fig. 1 The three steps of abstraction-based control.

computing over-approximations of the set of reachable states. However, these techniques are mainly focused on verification, and are restricted to some classes of dynamical systems, as the exact reachable set can only be computed in specific cases [AD94, LPY01].

More often than not with cyber-physical systems, the only sensible way of developing a controller is by discretizing the different variables, thus transforming the control task into a purely combinatorial problem on a finite-state mathematical object, called an *abstraction* (also known as *symbolic model*) of this system. This renowned approach, termed *abstraction-based control* (a.k.a. *symbolic control*) [Tab09], involves a correct-by-design synthesis, whereby a finite-state model approximates the behavior of the original (a.k.a. *concrete*) system that, instead, evolves in a continuous (or even hybrid) state space. This is achieved by defining mathematical relations between the finite state machine and the original dynamics [AHKV98, Tab09, RWR16, BPDB18]. Most abstraction-based approaches [Tab09, Rei11, RWR16, ZPMT11, Gir13, Gir14] for designing a controller $\mathcal{C}_1$ that enforces the desired specifications Σ_1 on the original system $\mathcal{S}_1$ follow a systematic three-step procedure, as shown in Figure 1. First, both the original system $\mathcal{S}_1$ and the specifications Σ_1 are transposed into an abstract domain, resulting in an abstract system $\mathcal{S}_2$ and corresponding abstract specifications Σ_2. Next, an abstract controller $\mathcal{C}_2$ is synthesized to solve this abstract control

problem $(\mathcal{S}_2, \Sigma_2)$. Finally, in the third step, called *concretization* as opposed to abstraction, a controller $\mathcal{C}_1$ that enforces Σ_1 for $\mathcal{S}_1$ is derived from the abstract controller $\mathcal{C}_2$.

The *effectiveness* of this approach stems from replacing the concrete system, often characterized by an infinite number of states, with a finite state system. This allows the use of advanced computational tools from abstract control synthesis [BYG17, KV01], discrete-event systems [KG12, CL08] and games on automata [DAHM01, MNA03], to synthesize controllers ensuring specifications often difficult to enforce with classical control methods.

The *correctness* of this three-step approach is ensured by relating the concrete system with its abstraction in terms of a system relation. The notion of *alternating simulation relation* (ASR) [Tab09, Definition 4.19] is crucial there; indeed, it is proved in the seminal work of [AHKV98] that the ASR is a sufficient condition for a system $\mathcal{S}_2$ to correctly represent a system $\mathcal{S}_1$, such that any controller $\mathcal{C}_2$ for $\mathcal{S}_2$ can be concretized into a valid controller $\mathcal{C}_1$ for $\mathcal{S}_1$.

Although this approach offers a safety-critical framework, the available techniques suffer important scalability issues. Typically, these methods require discretizing both the state and input spaces using uniform hyperrectangles (*cells*) [RWR16]. To account for the quantization error between the actual and quantized states, the forward image of the cells must be over-approximated under constant discretized inputs. A major drawback of this approach is that, without *incremental stability* [Kha96], over-approximation increases the level of non-determinism in the symbolic system, potentially leading to intractable or unsolvable problems. Furthermore, the curse of dimensionality heavily impacts the uniform discretization, as the number of states grows exponentially with the system's dimension. In order to render these techniques practical, it is necessary to construct smarter abstractions that differ from classical techniques by partitioning the state and input space in a non trivial way.

In this thesis, we aim to address these two key challenges: the non-determinism in abstractions resulting from discretization and the curse of dimensionality caused by predefined uniform grid-based discretization. Our approach combines theoretical and algorithmic developments, along with practical implementation tools.

First, in order to mitigate the non-determinism arising from state discretization, we propose avoiding input space discretization. Instead, we exploit the full input space to design state-dependent controllers that either reduce non-determinism or ensure deterministic transitions using lo-

cal controllers. This approach necessitates a detailed analysis of the link between steps 1 and 3 of the abstraction procedure (Figure 1), i.e., between system relations and the concrete control architecture. This enables us to design more flexible templates for concrete controllers. To this end, we develop a theoretical framework that classifies and characterizes abstraction relations based on their concretization procedures.

Second, in order to mitigate the computational challenges posed by high-dimensional systems, we propose a problem-specific approach to abstraction construction. By merging steps 1 and 2 of the abstraction-based procedure (Figure 1), we co-design the abstraction and abstract controller, guided by the optimal control problem. This allows to construct the abstraction incrementally, focusing only on the relevant regions of the state space, thus reducing computational complexity. We establish a connection between dynamic programming techniques and alternating simulations, leveraging these methods to construct smart abstractions.

We review the outline of the thesis below, and briefly summarize each section.

⋆ Outline

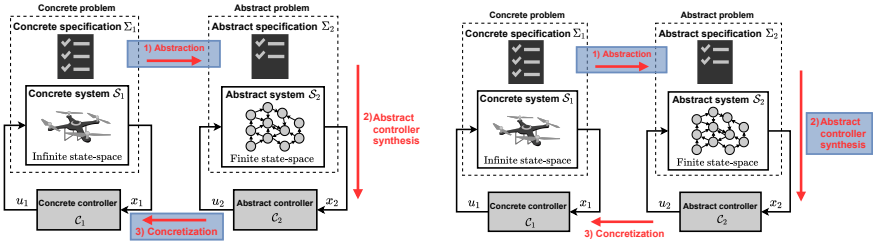
Before delving into the core of this thesis, let us briefly outline the content of this thesis, which is organized around the three key steps of the abstraction-based approach in Figure 1. Our main contributions are divided into two parts (Part II and Part III), which focus on the results obtained from exploring the connections between steps 1 and 3, and steps 2 and 3, respectively, as summarized in Figure 2.

Part I: Background

We review classical notions that will be essential throughout this thesis, including the control framework and a description of the limitations of classical abstraction-based techniques, which are addressed in this thesis.

Chapter 1: Preliminaries

This chapter introduces some definitions and notations that will be used throughout the thesis.



(a) Part II. **(b)** Part III.

Fig. 2 Structure of the contributions of the thesis. Part II focuses on characterizing the concretization procedure (step 3) with respect to the system relation between $\mathcal{S}_1$ and $\mathcal{S}_2$ (step 1). We show that one can tailor the abstraction relation depending on the required properties of the concretization step and of the finally obtained concrete controller, independently of the abstract controller synthesis process (step 2). Part III focuses on merging the abstraction step (step 1) and the synthesis of the abstract controller (step 2). We show that one can tailor the abstraction construction depending on the specific control problem considered.

Chapter 2: Control framework

In this chapter, we present the control framework used in this thesis, including the definitions of systems, system compositions and control problems such as specification and optimal control. We also introduce dynamic programming concepts that will be adapted in the abstraction-based control framework in Chapter 6. These concepts will be leveraged in Chapter 7 and Chapter 8 to construct abstractions on a reduced portion of the state space, solving an optimal control problem.

Chapter 3: Abstraction-based control

We formally define the abstraction-based approach within the given control framework. We then explore the classical abstraction-based method, which we will later compare with the methods that we introduce in this thesis, highlighting its limitations.

Part II: Theory

Motivated by the numerous system relations proposed in the literature and the limitations of restricting concrete controllers to specific templates, we present a framework to systematically classify and characterize system

relations in terms of their concretization procedure. Specifically, we show how the abstraction relation can be tailored based on the desired properties of the concretization step and the final concrete controller, independently of the abstract controller synthesis process (Figure 2a). The results presented in this part are partially published in [CGJ24] and were developed in collaboration with Antoine Girard.

Chapter 4: Simulation relations

We provide a systematic characterization of simulation relations within a plug-and-play control architecture. Our main contribution is a theorem (Corollary 4.19) that links these simulation relations to the control architecture inherited by the concrete system. Meanwhile, we introduce key system relations, such as the *predictive simulation relation* (PSR), *feedforward abstraction relation* (FAR), and the *memoryless concretization relation* (MCR), showing their impact on the control design. We also provide explicit constructions of these relations, highlighting the strengths and limitations of their respective concretization procedures.

Chapter 5: Characterization of simulation relations

We extend the results from the previous chapter by proving a converse result for Corollary 4.19, showing that if a concretization procedure works for any abstract controller, the original system must be related to its abstraction by the corresponding system relation. To achieve this, we introduce a universal system transformation, referred to as the *augmented system*, which is in feedback refinement relation (FRR) [RWR16, Definition V.2] with the abstract system, enabling a more flexible class of controllers than the piece-wise constant controllers of the FRR.

Part III: Algorithms

In this part, we propose tailoring the abstraction construction to the specific control problem by integrating abstraction and controller synthesis (steps 1 and 2 from Figure 2b). By co-designing the abstraction and the abstract controller based on the optimal control problem, the abstraction can be constructed lazily, focusing only on the necessary regions of the state space.

Chapter 6: Dynamic programming on abstractions

Building on the proven success of dynamic programming for solving optimal control problems for finite systems, this chapter establishes the connection between dynamic programming techniques and alternating simulations. Specifically, we focus on demonstrating how value functions that satisfy Bellman inequalities, computed efficiently in the abstract system, can be transferred to the concrete system. In Chapter 7 and Chapter 8, we will demonstrate how these dynamic programming surrogate functions, though suboptimal, can be leveraged to construct smart abstractions. The results presented in this chapter are based on preliminary work published in [CLEJ21].

Chapter 7: Lazy and hierarchical abstractions

In this chapter, we build on the modular framework between alternating simulations and the Bellman operator introduced in Chapter 6 to transfer optimal cost bounds from abstractions across different levels of nested partitions. The chapter is organized into three sections, each presenting an algorithm that builds upon the previous one. We develop a branch-and-bound algorithm that leverages the bounds obtained from the abstractions to compute the abstraction over a reduced portion of the state space. The results presented in this chapter are based on preliminary work published in [CLEJ21].

Chapter 8: Lazy abstraction based on non-uniform optimized cells

This chapter presents a novel approach to overcome the limitations of classical abstraction techniques, which are constrained by the curse of dimensionality and the non-determinism introduced by discretization. The proposed algorithm combines a Rapidly-Exploring Random Trees (RRT) method with ellipsoidal coverings, integrating local controllers within a high-level controller. This hierarchical control structure eliminates the need for input space discretization and guarantees deterministic transitions, providing formal guarantees even in the presence of noise. Moreover, the use of non-uniform cells and a lazy construction approach significantly reduces the complexity of the abstraction, while addressing specific control problems. This chapter gathers the results that have been published in [CEJ24, CEJ23] based on a collaboration with Lucas N. Egidio.

Chapter 9: Dionysos.jl: a modular platform for smart symbolic control

In this chapter, we present `Dionysos.jl`, a modular package designed

to solve optimal control problems for complex dynamical systems, using both state-of-the-art and experimental techniques from symbolic control, optimization, and learning. Notably, all the algorithms and results discussed in this thesis are implemented within the `Dionysos.jl` package. This chapter gathers the results that have been published in [CBLJ24a, CBLJ24b].

List of publications

HERE is the list of abstract, conference and journal papers written during this PhD thesis, including those whose content might not be included in this manuscript. It also includes a repeatability package that has accompanied the submission of a conference paper.

⋆ Abstracts

- Benoît Legat, Guillaume Berger, Julien Calbert, Raphaël M. Jungers. *Dionysos.jl: Optimal Control of Cyber-Physical Systems*. Set Propagation Methods in Julia: Techniques and Applications, JuliaCon2021, 2021.
- Julien Calbert, Raphaël M. Jungers. *Heuristic symbolic models via importance sampling and application to limit-cycle detection*. Symposium on Algorithmic Information Theory and Machine Learning, Alan Turing Institute, 2022.
- Julien Calbert, Lucas N. Egidio, Raphaël M. Jungers. *Smart abstraction based on iterative ellipsoidal covering*. In Proceedings of the 42th Benelux meeting on systems and control, Benelux Meeting, 2023.
- Julien Calbert, Adrien Banse, Raphaël M. Jungers. *Dionysos.jl: a Modular Platform for Smart Symbolic Control*. In Julia and Optimization Days 2023, Paris, 2023.
- Julien Calbert, Adrien Banse, Raphaël M. Jungers. *Dionysos.jl: a Modular Platform for Smart Symbolic Control*. In JuliaCon Local Eindhoven 2023, Eindhoven, 2023.

- Julien Calbert, Sébastien Mattenet, Antoine Girard, and Raphaël M. Jungers. *Memoryless concretization relation*. In Proceedings of the 43th Benelux meeting on systems and control, Benelux Meeting, 2024.

⋆ Conference proceedings

[RCJ21] Wei Ren, Julien Calbert, and Raphaël M. Jungers. *Zonotope-based controller synthesis for ltl specifications*. In 2021 60th IEEE Conference on Decision and Control (CDC), 2021.

[CLEJ21] Julien Calbert, Benoît Legat, Lucas N. Egidio, and Raphaël M. Jungers. *Alternating simulation on hierarchical abstractions*. In 2021 60th IEEE Conference on Decision and Control (CDC), 2021.

[CJ23] Julien Calbert and Raphaël M. Jungers. *Data-driven heuristic symbolic models and application to limit-cycle detection*. In 2023 American Control Conference (ACC), 2023.

[CEJ23] Julien Calbert, Lucas N. Egidio, and Raphaël M. Jungers. *An efficient method to verify the inclusion of ellipsoids*. IFAC-PapersOnLine, 56(2):1958–1963, 2023.

[CMGJ24] Julien Calbert, Sébastien Mattenet, Antoine Girard, and Raphaël M. Jungers. *Memoryless concretization relation*. In Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control, 2024.

⋆ Journal papers

[CEJ24] Julien Calbert, Lucas N. Egidio, and Raphaël M. Jungers. *Smart abstraction based on iterative cover and non-uniform cells*. IEEE Control Systems Letters, 2024.

[CBLJ24a] Julien Calbert, Adrien Banse, Benoît Legat, and Raphaël M. Jungers. *Dionysos. jl: a modular platform for smart symbolic control*. In Proceedings of the JuliaCon Conferences, 2024. (*submitted*)

[CGJ24] Julien Calbert, Antoine Girard, and Raphaël M. Jungers. *Classification of simulation relations for symbolic control*. Automatica. (*submitted*)

⋆ Repeatability package

[CBLJ24b] Julien Calbert, Adrien Banse, Benoît Legat, and Raphaël M. Jungers. *Dionysos.jl: a modular platform for smart symbolic control*, 2024. Accessible on https://codeocean.com/capsule/8370135/tree/v2.

List of symbols

The next list compiles the common notations used throughout the thesis. Additional notations are defined within the text as needed.

Basics

$\mathbb{R}, \mathbb{Z}, \mathbb{N}$ — Real, integers and positive integer numbers (p. 19)

$[i; j], [i; j[$ — $[i, j] \cap \mathbb{Z}$, $[i, j[\cap \mathbb{Z}$, $j \geq i$ (p. 19)

$2^{\mathcal{A}}$ — Power set of a set $\mathcal{A}$ (p. 20)

$\varnothing$ — Empty set (p. 19)

$\mathbf{A}\mathcal{A}$ — $\{\mathbf{A}x \mid x \in \mathcal{A}\}$ where $\mathbf{A}$ is a matrix and $\mathcal{A}$ a set (p. 20)

$\mathbf{A}^{-1}\mathcal{A}$ — $\{x \mid \mathbf{A}x \in \mathcal{A}\}$ where $\mathbf{A}$ is a matrix and $\mathcal{A}$ a set (p. 20)

$\operatorname{int}(\mathcal{A}), \partial\mathcal{A}$ — Interior and boundary of $\mathcal{A} \subseteq \mathbb{R}^n$, respectively. (p. 206)

$\mathcal{A} \oplus \mathcal{B}$ — Minkowski sum of two sets $\mathcal{A}, \mathcal{B} \subseteq \mathbb{R}^n$ (p. 19)

$\subseteq, \subset, \not\subseteq$ — Inclusion, strict inclusion and non-inclusion (p. 20)

$\mathscr{F}(\mathcal{X}, \mathcal{U})$ — Set of single-valued functions $f : \mathcal{X} \to \mathcal{U}$ (p. 154)

$\mathcal{K}, \mathcal{K}_\infty, \mathcal{KL}$ — Class $\mathcal{K}$, $\mathcal{K}_\infty$ and $\mathcal{KL}$ functions (p. 42)

Vectors and matrices

$\mathbb{R}^n$ — Vectors of dimension n (p. 19)

$\mathbb{R}^{m \times n}$ — $m \times n$ matrices (p. 19)

$\|\cdot\|_p$ — p-norm (p. 19)

$\mathrm{support}(v)$ — Support of $v \in \mathbb{R}^n$: $\{i \in [1;n] : v_i \neq 0\}$ (p. 205)

$\mathbf{I}_n \subseteq \mathbb{R}^{n \times n}$ — Identity matrix (p. 163)

$\mathbf{A}^\top$ — Transpose of $\mathbf{A}$ (p. 21)

$\mathbf{A}^{-1}$ — Inverse of $\mathbf{A}$ (p. 20)

$\mathbf{A}^\dagger$ — Moore–Penrose inverse of $\mathbf{A}$ (p. 207)

$\lambda_{\mathsf{min}}(\mathbf{A})$ — Smallest eigenvalue in absolute value of $\mathbf{A}$ (p. 204)

$\mathbb{S}^n_{>0}$ — $n \times n$ positive definite matrices (p. 21)

$\mathbf{A} \succ \mathbf{0}, \mathbf{A} \succeq \mathbf{0}$ — Positive definitive and semi-definite matrix (p. 160)

Convex analysis

$\mathrm{conv}(\mathcal{A})$ — Convex hull of $\mathcal{A}$ (p. 160)

$\mathrm{CC}(\mathcal{A})$ — Chebychev center of $\mathcal{A} \subseteq \mathbb{R}^n$ (p. 20)

$\mathrm{H}(c,h)$ — Hyperrectangle of center c and half-length h (p. 21)

$\mathrm{E}(c,\mathbf{P})$ — Ellipsoid of center c and shape matrix $\mathbf{P}$ (p. 21)

$\mathrm{B}(c,r)$ — Euclidean ball of center c and radius r (p. 21)

Abbreviations

CPS — Cyber-physical systems (p. 1)

LMI — Linear matrix inequality (p. 152)

FLOPs — Floating-point operations (p. 200)

δ-GAS — Incrementally globally asymptotically stable (p. 43)

DP — Dynamic programming (p. 35)

ADP — Approximate dynamic programming (p. 35)

RRT — Rapidly-Exploring Random Trees (p. 152)

ASR — Alternating simulation relation (p. 47)

FRR — Feedback refinement relation (p. 49)

S-ASR — Strong alternating simulation relation (p. 79)

PSR _____ Predictive simulation relation (p. 70)

DSR _____ Delayed simulation relation (p. 70)

FAR _____ Feedforward abstraction relation (p. 70)

MCR _____ Memoryless concretization relation (p. 70)

SFR _____ State-feedback abstraction (p. 79)

$\pm$ _____ Should be read twice replacing it by "$+$" and "$-$" (p. 21)

PART I
Background

1

Preliminaries

In order for this thesis to be self-contained, we cover in this chapter preliminaries that are used throughout the text.

1.1 Basics

The sets $\mathbb{R}, \mathbb{Z}, \mathbb{N}$ denote respectively the sets of real, integers and positive integer numbers. For example, we use $[a, b] \subseteq \mathbb{R}$ to denote a closed continuous interval and $[a; b] = [a, b] \cap \mathbb{Z}$ for discrete intervals. The symbol $\varnothing$ denotes the empty set. We denote a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ by a capital letter and in bold, and the element of the i-th row and j-th column of $\mathbf{A}$ (with $1 \leq i \leq m$ and $1 \leq j \leq n$) by A_{ij}.
The Minkowski sum of two sets $\mathcal{A}, \mathcal{B} \subseteq \mathbb{R}^n$ is defined as

$$\mathcal{A} \oplus \mathcal{B} = \{a + b \mid a \in \mathcal{A},\ b \in \mathcal{B}\}.$$

Given a vector $x \in \mathbb{R}^n$, the p-norm is defined by

$$\|x\|_p = \sqrt[p]{\sum_{i=1}^{n} |x_i|^p}.$$

Given a compact convex set $\mathcal{A} \subseteq \mathbb{R}^n$, we define the Chebychev center of $\mathcal{A}$

as the center of the minimal-radius ball enclosing the entire set

$$\mathrm{CC}(\mathcal{A}) = \mathrm{argmin}_{x\in\mathcal{A}} \max_{y\in\mathcal{A}} \|x-y\|_2.$$

The *convex hull* of a set $\mathcal{A} \subseteq \mathbb{R}^n$ is the smallest convex set containing $\mathcal{A}$,

$$\mathrm{conv}(\mathcal{A}) = \{\lambda_1 x_1 + \ldots + \lambda_r x_r \mid x_1, \ldots, x_r \in \mathcal{A},\ \lambda \in \Delta^{r-1}\},$$

where $\Delta^{r-1} = \{\lambda \in \mathbb{R}^r \mid \lambda_1 + \ldots + \lambda_r = 1, \text{ and } \lambda_1, \ldots, \lambda_r \geq 0\}$ denotes the set of coefficient of a *convex combination*, i.e., the $r-1$-dimensional standard simplex.

1.2 Functions and relations

Given two sets $\mathcal{X}, \mathcal{Y}$, we define a *single-valued map* as $f : \mathcal{X} \to \mathcal{Y}$, while a *set-valued map* is defined as $f : \mathcal{X} \to 2^{\mathcal{Y}}$, where $2^{\mathcal{Y}}$ is the power set of $\mathcal{Y}$, i.e., the set of all subsets of $\mathcal{Y}$.

Given a function $f : \mathcal{X} \to \mathcal{Y}$ and two subsets $\mathcal{A} \subseteq \mathcal{X}, \mathcal{B} \subseteq \mathcal{Y}$, we define the following notation

$$f(\mathcal{A}) = \{f(x) \in \mathcal{Y} \mid x \in \mathcal{A}\},$$
$$f^{-1}(\mathcal{B}) = \{x \in \mathcal{X} \mid f(x) \in \mathcal{B}\}.$$

Note that f does not need to be injective in these definitions. By slight abuse of notation, we also use the notation $\mathbf{A}\mathcal{A}$, $\mathbf{A}^{-1}\mathcal{B}$ where $\mathbf{A}$ is a matrix.

The set of maps $\mathcal{A} \to \mathcal{B}$ is denoted $\mathcal{B}^{\mathcal{A}}$, and the set of all signals that take their values in $\mathcal{B}$ and are defined on intervals of the form $[0; N[= [0; N-1]$ is denoted $\mathcal{B}^{\infty}$, $\mathcal{B}^{\infty} = \bigcup_{N\in\mathbb{N}\cup\{\infty\}} \mathcal{B}^{[0;N[}$.

Given a set-valued map $f : \mathcal{X} \to 2^{\mathcal{Y}}$ and $\boldsymbol{x} \in \mathcal{X}^{[0;N[}$, we denote by $f(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathcal{Y}^{[0;N[} \mid \forall k \in [0; N[:\ y(k) \in f(x(k))\}$.

Let $\boldsymbol{v} \in (\mathcal{X} \times \mathcal{Y})^{[0;N[}$ such that $\boldsymbol{v} = ((x(0), y(0)), \ldots, (x(N-1), y(N-1)))$ with $\boldsymbol{x} \in \mathcal{X}^{[0;N[}$ and $\boldsymbol{y} \in \mathcal{Y}^{[0;N[}$. We define the projection onto the first and second components of the sequence as

$$\pi_{\mathcal{X}}(\boldsymbol{v}) = \boldsymbol{x} \text{ and } \pi_{\mathcal{Y}}(\boldsymbol{v}) = \boldsymbol{y}. \tag{1.1}$$

Given two sets $\mathcal{X}_1$ and $\mathcal{X}_2$, we identify a binary *relation* $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ with set-valued maps, i.e.,

$$R(x_1) = \{x_2 \mid (x_1, x_2) \in R\} \text{ and } R^{-1}(x_2) = \{x_1 \mid (x_1, x_2) \in R\}.$$

If $R_{1,2} \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ and $R_{2,3} \subseteq \mathcal{X}_2 \times \mathcal{X}_3$, then their composition

$$R_{2,3} \circ R_{1,2} = \{(x_1, x_3) \in \mathcal{X}_1 \times \mathcal{X}_3 \mid \exists x_2 \in \mathcal{X}_2 \text{ such that } (x_1, x_2) \in R_{1,2} \text{ and } (x_2, x_3) \in R_{2,3}\}. \quad (1.2)$$

1.3 Geometric sets

An n-dimensional *hyperrectangle* centered at $c \in \mathbb{R}^n$ with *half-lengths* $h \in \mathbb{R}^n_{>0}$ is defined as

$$\mathrm{H}(c, h) \coloneqq \{x \in \mathbb{R}^n \mid |x_i - c_i| \leq h_i \text{ for } i \in [1; n]\}. \quad (1.3)$$

The *vertices* of $\mathrm{H}(c, h)$ are the points $c \pm h$, where each coordinate i takes either h_i or $-h_i$. The symbol $\pm$ should be read twice replacing it by "$+$" and "$-$".

An n-dimensional *ellipsoid* centered at $c \in \mathbb{R}^n$ with *shape matrix* $\mathbf{P} \in \mathbb{S}^n_{>0}$ is defined as

$$\mathrm{E}(c, \mathbf{P}) \coloneqq \{x \in \mathbb{R}^n \mid (x - c)^\top \mathbf{P}(x - c) \leq 1\}. \quad (1.4)$$

A n-dimensional Euclidean *ball* centered at c with radius $r > 0$ is

$$\mathrm{B}(c, r) \coloneqq \{x \in \mathbb{R}^n \mid \|x - c\|_2 \leq r\} = \mathrm{E}(c, r^{-2}\mathbf{I}_n). \quad (1.5)$$

2

Control framework

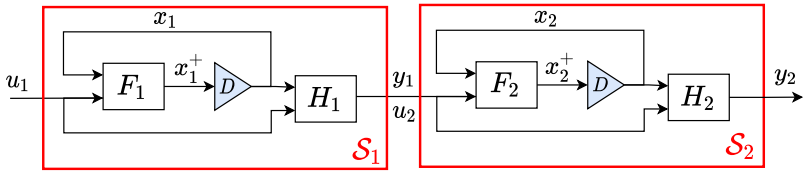
In this chapter, we introduce the control framework employed in this thesis, which provides a common formalism to describe systems, controllers, quantizers, and interconnected systems.

First, in Section 2.1, we motivate the use of the control framework from [RWR16]. We then define systems and their composition in Section 2.2 and Section 2.3, respectively. In Section 2.4, we outline the control problems, including output specification and optimal control. Next, dynamic programming concepts are introduced in Section 2.5 that will be adapted within the abstraction-based control framework in Chapter 6. Finally, Section 2.6 and Section 2.7 cover sampled continuous-time systems used in numerical experiments and the concept of incremental stability, which is essential for abstraction-based control.

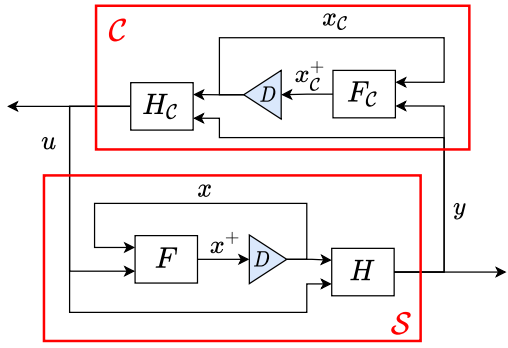
2.1 Motivations

We consider dynamical systems defined by the tuple $\mathcal{S} = (\mathcal{X}, \mathcal{U}, \mathcal{Y}, F, H)$ where $\mathcal{X}, \mathcal{U}$, and $\mathcal{Y}$ are the *state*, *input*, and *output* sets, respectively, and $F : \mathcal{X} \times \mathcal{U} \to 2^{\mathcal{X}}$ and $H : \mathcal{X} \times \mathcal{U} \to 2^{\mathcal{Y}}$ are the *transition* and *output* maps, such that

$$\begin{aligned} x(k+1) &\in F(x(k), u(k)) \\ y(k) &\in H(x(k), u(k)). \end{aligned} \tag{2.1}$$



(a) Serial composition.



(b) Feedback composition.

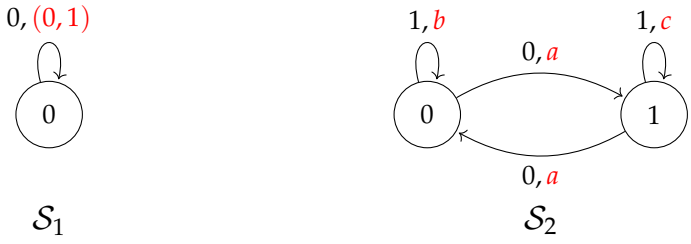
Fig. 2.1 Compositions of two systems of the form (2.1). The blue blocks are *delay blocks* which represent memory elements that store and provide the previous value of a signal, e.g., $D(x_2(k)) = x_2(k-1)$.

The use of a set-valued function to describe the transition and output maps of a system provides a unified formalism for modeling perturbations and other forms of non-determinism.

While this model is widely used in the literature, it has several disadvantages, which we will discuss here.

Firstly, this model introduces subtle issues with interconnected systems, as detailed in [RWR16, Section III.A]. Specifically, the class of systems described by (2.1) is not closed under serial composition as shown in Figure 2.1a, given the natural constraint that the state space of the composed system must equal the product of the state spaces of the individual systems, as proven with the following example.

Example 2.1 ([RWR16, Section III.A]). Consider the serial composition in Figure 2.1a of $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, \mathcal{Y}_1, F_1, H_1)$ and $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, \mathcal{Y}_2, F_2, H_2)$ of the form (2.1) such that $\mathcal{X}_1 = \mathcal{U}_1 = \{0\}$, $\mathcal{Y}_1 = \mathcal{X}_2 = \mathcal{U}_2 = \{0, 1\}$, and $\mathcal{Y}_2 = \{a, b, c\}$. The transition and output maps of these two systems are illustrated graphically by



where the inputs and outputs are given by the black and red labels on the transitions, respectively. We prove that one cannot define a system $\mathcal{S} = (\mathcal{X}, \mathcal{U}_1, \mathcal{Y}_2, F, H)$ of the form (2.1) that recovers the behavior at terminal state u_1 and y_2 under the constraint that $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$. Since $|\mathcal{Y}_2| = 3 > 2 = |\mathcal{X} \times \mathcal{U}_1|$, the map H must be set-valued, which in turn implies that the following properties of the system in Figure 2.1a are not satisfied: (i) between any two appearances of b in y_2 there are an even number of a's, and (ii) between any appearance of b and any appearance of c there are an odd number of a's. △

Secondly, although this model is sufficiently expressive to describe the systems we aim to control, it does not support the implementation of the specific class of controllers considered in this thesis, as demonstrated in the following examples.

Example 2.2. Consider the feedback composition in Figure 2.1b where $\mathcal{S} = (\mathcal{X}, \mathcal{U}, \mathcal{X}, F, H)$ with $H(x,u) = \{x\}$, and $\mathcal{C}_1 = (\mathcal{X}_\mathcal{C}, \mathcal{X}, \mathcal{U}, F_{\mathcal{C}_1}, H_{\mathcal{C}_1})$. It is impossible to implement $F_{\mathcal{C}_1}$ and $H_{\mathcal{C}_1}$ such that the system resulting from the feedback composition of $\mathcal{C}_1$ and $\mathcal{S}$ in Figure 2.1b has the following behavior

$$\begin{aligned} x_\mathcal{C}^+ &\in h_2(x_\mathcal{C}, x) \\ u &\in h_1(x, x_\mathcal{C}^+) \\ x^+ &\in F(x,u) \end{aligned} \tag{2.2}$$

where $h_1 : \mathcal{X} \times \mathcal{X}_\mathcal{C} \to 2^\mathcal{U}$ and $h_2 : \mathcal{X}_\mathcal{C} \times \mathcal{X} \to 2^{\mathcal{X}_\mathcal{C}}$ are set-valued functions. Indeed, the set-valued function h_2 must appear both in $F_{\mathcal{C}_1}$ and $H_{\mathcal{C}_1}$ as in the following implementation

$$\begin{aligned} F_{\mathcal{C}_1}(x_\mathcal{C}, x) &= \{\widetilde{z} \in h_2(x_\mathcal{C}, x)\}, \\ H_{\mathcal{C}_1}(x_\mathcal{C}, x) &= \{u \mid z \in h_2(x_\mathcal{C}, x), u \in h_1(x, z)\}, \end{aligned}$$

which cannot enforce that $\widetilde{z}$ is equal to z due to the non-determinism of h_2. △

Example 2.3. Consider the feedback composition in Figure 2.1b where $\mathcal{S} = (\mathcal{X}, \mathcal{U}, \mathcal{X}, F, H)$ with $H(x,u) = \{x\}$, and $\mathcal{C}_2 = (\mathcal{X}_\mathcal{C}, \mathcal{X}, \mathcal{U}, F_{\mathcal{C}_2}, H_{\mathcal{C}_2})$. It is impossible to implement $F_{\mathcal{C}_2}$ and $H_{\mathcal{C}_2}$ such that the system resulting from the feedback composition of $\mathcal{C}_2$ and $\mathcal{S}$ in Figure 2.1b has the following behavior

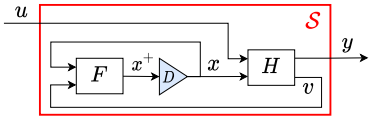
$$\begin{aligned} u &\in h_1(x, x_\mathcal{C}) \\ x_\mathcal{C}^+ &\in h_2(x_\mathcal{C}, x, u) \\ x^+ &\in F(x,u) \end{aligned} \tag{2.3}$$

where $h_1 : \mathcal{X} \times \mathcal{X}_\mathcal{C} \to 2^\mathcal{U}$ and $h_2 : \mathcal{X}_\mathcal{C} \times \mathcal{X} \times \mathcal{U} \to 2^{\mathcal{X}_\mathcal{C}}$ are set-valued functions. Indeed, the set-valued function h_1 must appear both in $F_{\mathcal{C}_2}$ and $H_{\mathcal{C}_2}$ as in the following implementation

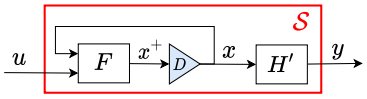
$$\begin{aligned} F_{\mathcal{C}_2}(x_\mathcal{C}, x) &= \{z \mid \widetilde{u} \in h_1(x, x_\mathcal{C}), z \in h_2(x_\mathcal{C}, x, \widetilde{u})\}, \\ H_{\mathcal{C}_2}(x_\mathcal{C}, x) &= \{u \in h_1(x, x_\mathcal{C})\}, \end{aligned}$$

which cannot enforce that $\widetilde{u}$ is equal to u due to the non-determinism of h_1. △

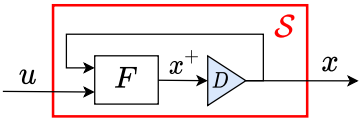
As we will see, the non-deterministic nature of h_1 and h_2 is fundamental for developing non-trivial abstraction-based controllers.



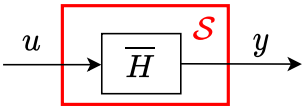
(a) General system (2.4).



(b) Moore system (2.7).



(c) Simple system (2.8).



(d) Static system (2.9).

Fig. 2.2 System representations. For **(b)**, **(c)**, and **(d)**, these are simplified block diagrams of **(a)**.

As illustrated in Example 2.2 and Example 2.3, the formalism in (2.1) does not allow for modeling systems where a quantity, computed non-deterministically in the output map, must be reused in the transition map.

To address these issues, we adopt a more general form of systems proposed in [RWR16, Definition III.1], originally introduced to ensure closedness under the serial composition of two systems (see Example 2.1).

2.2 Systems

Systems and controllers are defined using the following common formalism [RWR16, Definition III.1].

Definition 2.4 (System)**.** A *system* is a sextuple

$$\mathcal{S} = (\mathcal{X}, \mathcal{U}, \mathcal{V}, \mathcal{Y}, F, H), \tag{2.4}$$

where $\mathcal{X}, \mathcal{U}, \mathcal{V}, \mathcal{Y}$ are respectively the sets of *states*, *inputs*, *internal variables* and *outputs*, and $F : \mathcal{X} \times \mathcal{V} \to 2^{\mathcal{X}}$ and $H : \mathcal{X} \times \mathcal{U} \to 2^{\mathcal{Y} \times \mathcal{V}}$ are respectively the *transition* and *output* maps such that

$$x(k+1) \in F(x(k), v(k)) \tag{2.5}$$
$$(y(k), v(k)) \in H(x(k), u(k)). \tag{2.6}$$

A quadruple $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y}) \in \mathcal{U}^{[0;N[} \times \mathcal{V}^{[0;N[} \times \mathcal{X}^{[0;N[} \times \mathcal{Y}^{[0;N[}$ is a *trajectory* of system $\mathcal{S}$ starting at $x(0) \in \mathcal{X}$ if $N \in \mathbb{N} \cup \{\infty\}$, (2.5) holds for all $k \in [0; N-1[$, (2.6) holds for all $k \in [0; N[$. △

In this more general formalism, the inputs u affect the states x via the internal variables v. The internal variable v provides a way to address the issue outlined in the previous section, as detailed below. Specifically, it enables the transmission of a quantity, computed non-deterministically in the output map, to the transition map.

Returning to Example 2.1, the internal variables allow to introduce the constraint $u_2 = y_1$ imposed by the composition in Figure 2.1a and recover the behavior of the serial composed system with a system $\mathcal{S}$ of the form (2.4) given by $\mathcal{X} = \{0,1\}, \mathcal{U} = \{0\}, \mathcal{V} = \mathcal{Y} = \{a,b,c\}$ with $F(0,a) = F(1,c) = \{1\}$, $F(1,a) = F(0,b) = \{0\}$, $H(0,0) = \{(a,a),(b,b)\}$, $H(1,0) = \{(a,a),(c,c)\}$.

Returning to Example 2.2 and Example 2.3, we can use formalism (2.4) to define the controllers $\mathcal{C}_i = (\mathcal{X}_\mathcal{C}, \mathcal{X}, \mathcal{V}_\mathcal{C}, \mathcal{U}, F_{\mathcal{C}_i}, H_{\mathcal{C}_i})$ for $i \in \{1,2\}$ with $\mathcal{V}_\mathcal{C} = \mathcal{X}_\mathcal{C}$, $F_{\mathcal{C}_1}(x_\mathcal{C}, z) = F_{\mathcal{C}_2}(x_\mathcal{C}, z) = \{z\}$ and

$$H_{\mathcal{C}_1}(x_\mathcal{C}, x) = \{(u,z) \mid z \in h_2(x_\mathcal{C}, x), u \in h_1(x,z)\},$$
$$H_{\mathcal{C}_2}(x_\mathcal{C}, x) = \{(u,z) \mid u \in h_1(x, x_\mathcal{C}), z \in h_2(x_\mathcal{C}, x, u)\},$$

which ensure that the closed-loop system behavior follows (2.2) for $\mathcal{C}_1$ and (2.3) for $\mathcal{C}_2$. The internal variable $v_\mathcal{C}$ allows to enforce $\widetilde{z} = z$ for $\mathcal{C}_1$ and $\widetilde{u} = u$ for $\mathcal{C}_2$.

The general system $\mathcal{S}$ in (2.4) is represented by the block diagram in Figure 2.2a. We classify the system (2.4) as follows.

Definition 2.5. The system $\mathcal{S}$ in (2.4) is

(*i*) *Autonomous* if $\mathcal{U}$ is a singleton[1].

(*ii*) *Static* if $\mathcal{X}$ is a singleton.

(*iii*) *Moore*[2] if

$$\mathcal{S} = (\mathcal{X}, \mathcal{U}, \mathcal{U}, \mathcal{Y}, F, H), \tag{2.7}$$

with $\mathcal{V} = \mathcal{U}$, and where $H(x,u) = H'(x) \times \{u\}$ for some set-valued

[1] When $\mathcal{X}, \mathcal{U}, \mathcal{V}$, and $\mathcal{Y}$ are singleton sets, we will consider, without loss of generality, the singleton $\{0\}$.

[2] This definition is slightly stronger than that in [RWR16] and sufficient for this thesis.

function $H' : \mathcal{X} \to 2^{\mathcal{Y}}$. Intuitively, a Moore system has the form

$$x(k+1) \in F(x(k), u(k))$$
$$y(k) \in H'(x(k)),$$

representing a system whose output does not depend directly on the input. For the sake of simplifying notation, we denote the Moore system (2.7) by the tuple $\mathcal{S} = (\mathcal{X}, \mathcal{U}, \mathcal{Y}, F, H')$.

(*iv*) *Simple* if

$$\mathcal{S} = (\mathcal{X}, \mathcal{U}, \mathcal{U}, \mathcal{X}, F, I_{\mathcal{X}\times\mathcal{U}}), \tag{2.8}$$

with $\mathcal{V} = \mathcal{U}$, $\mathcal{Y} = \mathcal{X}$, and $H = I_{\mathcal{X}\times\mathcal{U}}$. Intuitively, a simple system has the form

$$x(k+1) \in F(x(k), u(k)),$$

representing a system without an output. For the sake of simplifying notation, we denote the simple system (2.8) by the tuple $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$. Note that a simple system is Moore.

(*v*) *Static quantizer* if

$$\mathcal{S} = (\{0\}, \mathcal{U}, \{0\}, \mathcal{Y}, F, H), \tag{2.9}$$

with $\mathcal{X} = \mathcal{V} = \{0\}$, $F(0,0) = \{0\}$, and where $H(0,u) = \{(y,0) \mid y \in \overline{H}(u)\}$ for some set-valued function $\overline{H} : \mathcal{U} \to 2^{\mathcal{Y}}$. Intuitively, a static quantizer has the form

$$y(k) \in \overline{H}(u(k)),$$

representing a static system characterized solely by a set-valued function. For the sake of simplifying notation, we denote the static quantizer (2.9) by the set-valued map $\overline{H}$.

We illustrate systems (2.4), (2.7), (2.8) and (2.9) in Figure 2.2. △

For a Moore system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, \mathcal{Y}, F, H)$, we define the set-valued operator of *available inputs* as $\mathcal{U}_{\mathcal{S}}(x) = \{u \in \mathcal{U} \mid F(x,u) \neq \varnothing\}$, which gives the set of inputs u available at a given state x. By extension, we define the operator of available inputs on a set $\mathcal{Z} \subseteq \mathcal{X}$ as $\mathcal{U}_{\mathcal{S}}(\mathcal{Z}) = \bigcap_{x\in\mathcal{Z}} \mathcal{U}_{\mathcal{S}}(x)$. Additionally, we define the pre operator for a set $\mathcal{V} \subseteq \mathcal{X}$, which provides the set of states that can robustly reach $\mathcal{V}$ in one time step despite non-

determinism

$$\mathrm{pre}_{\mathcal{S}}(\mathcal{V}, u) = \{x \in \mathcal{X} \mid F(x,u) \subseteq \mathcal{V}\}, \tag{2.10}$$
$$\mathrm{pre}_{\mathcal{S}}(\mathcal{V}) = \{x \in \mathcal{X} \mid \exists u \in \mathcal{U}_{\mathcal{S}}(x) : \ F(x,u) \subseteq \mathcal{V}\}. \tag{2.11}$$

We say that a simple system is *deterministic* if for every state $x \in \mathcal{X}$ and control input $u \in \mathcal{U}$, $F(x,u)$ is either empty or a singleton. Otherwise, we say that it is *non-deterministic*. The system is *finite* if both $\mathcal{X}$ and $\mathcal{U}$ are finite.

Given two sets $\mathcal{U}$ and $\mathcal{Y}$, we define a binary *relation* $R \subseteq \mathcal{U} \times \mathcal{Y}$. Depending on the context, the symbol R will be used interchangeably to represent the binary relation itself, the corresponding set-valued map $R : \mathcal{U} \to 2^{\mathcal{Y}}$, or the associated static quantizer system (2.9), commonly referred to as a quantizer. Additionally, the relation R or its associated quantizer is said to be *strict* if $R(u) \neq \varnothing$ for every $u \in \mathcal{U}$, and *deterministic* if $R(u)$ is a singleton for every $u \in \mathcal{U}$.

We define the *behavior* of a system as its set of *maximal* trajectories.

Definition 2.6 (Behavior)**.** Given the system $\mathcal{S}$ in (2.4), the set

$$\mathcal{B}(\mathcal{S}) = \{\boldsymbol{y} \mid \exists \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x} \text{ such that } (\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y}) \text{ is a trajectory of } \mathcal{S} \text{ on } [0; N[, \\ \text{and if } N < \infty \text{ then } F(x(N-1), v(N-1)) = \varnothing\}$$

is called the *behavior* of $\mathcal{S}$. △

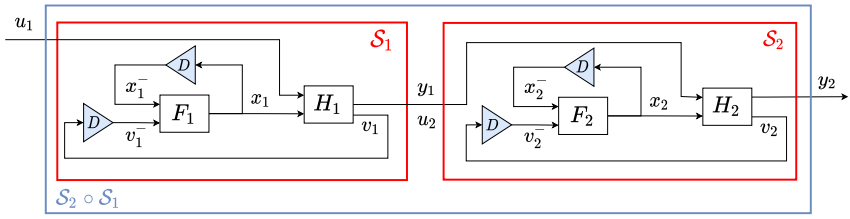
This notion of behavior encompasses both finite and infinite trajectories, allowing to include blocking systems, which typically appear in the context of terminating programs.

2.3 System compositions

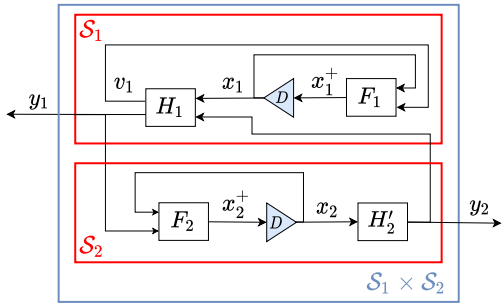
In this section, we define the *serial* and the *feedback composition* of two systems.

We start by defining the serial composition of two systems [RWR16, Definition III.2] as illustrated in Figure 2.3a.

Definition 2.7 (Serial composition)**.** Let $\mathcal{S}_i = (\mathcal{X}_i, \mathcal{U}_i, \mathcal{V}_i, \mathcal{Y}_i, F_i, H_i)$ be systems, $i \in \{1,2\}$. Then, $\mathcal{S}_1$ is *serial composable* with $\mathcal{S}_2$ if $\mathcal{Y}_1 \subseteq \mathcal{U}_2$. The *serial*



(a) Serial composition $\mathcal{S}_2 \circ \mathcal{S}_1$ of two systems of the form (2.4) (see Definition 2.7).



(b) Feedback composition $\mathcal{S}_1 \times \mathcal{S}_2$ of a system $\mathcal{S}_1$ of the form (2.4) and a Moore system $\mathcal{S}_2$ of the form (2.7) where $H_2'(x) = \{y_2 \mid \exists u_2 \in \mathcal{U}_2 : \ (y_2, u_2) \in H_2(x_2, u_2)\}$ (see Definition 2.8).

Fig. 2.3 Composition of two systems of the form (2.4).

composition of $\mathcal{S}_1$ and $\mathcal{S}_2$, denoted by $\mathcal{S}_2 \circ \mathcal{S}_1$, is defined as

$$\mathcal{S}_2 \circ \mathcal{S}_1 = (\mathcal{X}_{12}, \mathcal{U}_1, \mathcal{V}_{12}, \mathcal{Y}_2, F_{12}, H_{12}), \tag{2.12}$$

where $\mathcal{X}_{12} = \mathcal{X}_1 \times \mathcal{X}_2$, $\mathcal{V}_{12} = \mathcal{V}_1 \times \mathcal{V}_2$, $F_{12} : \mathcal{X}_{12} \times \mathcal{V}_{12} \to 2^{\mathcal{X}_{12}}$, and $H_{12} : \mathcal{X}_{12} \times \mathcal{U}_1 \to 2^{\mathcal{Y}_2 \times \mathcal{V}_{12}}$ satisfy

$$\begin{aligned} F_{12}(x, v) &= F_1(x_1, v_1) \times F_2(x_2, v_2), \\ H_{12}(x, u_1) &= \{(y_2, v) \mid \exists (y_1, v_1) \in H_1(x_1, u_1) \wedge (y_2, v_2) \in H_2(x_2, y_1)\}, \end{aligned}$$

where $x = (x_1, x_2)$ and $v = (v_1, v_2)$. △

In practice, the serial composition is used to model the interconnection of a system with a static quantizer (see Definition 2.5.(v)) in two contexts. First, the serial composition $\mathcal{C} \circ R$ of a controller $\mathcal{C} = (\mathcal{X}_\mathcal{C}, \mathcal{U}_\mathcal{C}, \mathcal{V}_\mathcal{C}, \mathcal{Y}_\mathcal{C}, F_\mathcal{C}, H_\mathcal{C})$ with an input quantizer $R : \mathcal{U}' \to 2^{\mathcal{U}_\mathcal{C}}$ is given by

$$\mathcal{C} \circ R = (\mathcal{X}_\mathcal{C}, \mathcal{U}', \mathcal{V}_\mathcal{C}, \mathcal{Y}_\mathcal{C}, F_\mathcal{C}, H'), \tag{2.13}$$

where $H' : \mathcal{X}_\mathcal{C} \times \mathcal{U}' \to 2^{\mathcal{Y}_\mathcal{C} \times \mathcal{V}_\mathcal{C}}$ is defined as $H'(x_\mathcal{C}, u') = H_\mathcal{C}(x_\mathcal{C}, R(u'))$. Second, the serial composition $R \circ \mathcal{S}$ of a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$ with a state quantizer $R : \mathcal{X} \to 2^{\mathcal{X}'}$ is given by

$$R \circ \mathcal{S} = (\mathcal{X}, \mathcal{U}, \mathcal{U}, \mathcal{X}', F, H'), \tag{2.14}$$

where $H' : \mathcal{X} \times \mathcal{U} \to 2^{\mathcal{X}' \times \mathcal{U}}$ is defined as $H'(x, u) = R(x) \times \{u\}$. Note that the system $R \circ \mathcal{S}$ is Moore.

We define the feedback composition of two systems [RWR16, Definition III.3] as illustrated in Figure 2.3b.

Definition 2.8 (Feedback composition)**.** Let $\mathcal{S}_i = (\mathcal{X}_i, \mathcal{U}_i, \mathcal{V}_i, \mathcal{Y}_i, F_i, H_i)$ be systems, $i \in \{1, 2\}$. Then, $\mathcal{S}_1$ is *feedback composable* with $\mathcal{S}_2$, denoted $\mathcal{S}_1$ is f.c. with $\mathcal{S}_2$, if $\mathcal{S}_2$ is Moore, $\mathcal{Y}_2 \subseteq \mathcal{U}_1$, $\mathcal{Y}_1 \subseteq \mathcal{U}_2$, and the following condition holds

$$\begin{aligned} &\text{If } (y_1, v_1) \in H_1(x_1, y_2),\ (y_2, v_2) \in H_2(x_2, y_1) \text{ and} \\ &\qquad F_2(x_2, v_2) = \varnothing, \text{ then } F_1(x_1, v_1) = \varnothing. \end{aligned} \tag{2.15}$$

The *feedback composition* of $\mathcal{S}_1$ and $\mathcal{S}_2$, denoted by $\mathcal{S}_1 \times \mathcal{S}_2$, is defined as

$$\mathcal{S}_1 \times \mathcal{S}_2 = (\mathcal{X}_{12}, \{0\}, \mathcal{V}_{12}, \mathcal{Y}_{12}, F_{12}, H_{12}), \tag{2.16}$$

where $\mathcal{X}_{12} = \mathcal{X}_1 \times \mathcal{X}_2$, $\mathcal{V}_{12} = \mathcal{V}_1 \times \mathcal{V}_2$, $\mathcal{Y}_{12} = \mathcal{Y}_1 \times \mathcal{Y}_2$, $F_{12} : \mathcal{X}_{12} \times \mathcal{V}_{12} \to 2^{\mathcal{X}_{12}}$, and $H_{12} : \mathcal{X}_{12} \times \{0\} \to 2^{\mathcal{Y}_{12} \times \mathcal{V}_{12}}$ satisfy

$$\begin{aligned} F_{12}(x, v) &= F_1(x_1, v_1) \times F_2(x_2, v_2), \\ H_{12}(x, 0) &= \{(y, v) \mid (y_1, v_1) \in H_1(x_1, y_2) \wedge (y_2, v_2) \in H_2(x_2, y_1)\}, \end{aligned}$$

where $x = (x_1, x_2)$, $v = (v_1, v_2)$ and $y = (y_1, y_2)$. △

In our context, the system $\mathcal{S}_2$ represents the system we aim to control, while $\mathcal{S}_1$ represents the controller. Note that the system $\mathcal{S}_1 \times \mathcal{S}_2$ is autonomous. The assumption that $\mathcal{S}_2$ is a Moore system in Definition 2.8 prevents the closed-loop system from containing a delay-free cycle [Vid81] by introducing a one-step delay in the feedback loop. The compatibility condition (2.15) ensures that if the concrete closed loop is non-blocking, then the abstract closed loop will also be non-blocking.

Given $\mathcal{S}_1$ and $\mathcal{S}_2$ in Definition 2.8, we have $\mathcal{B}(\mathcal{S}_1 \times \mathcal{S}_2) \subseteq (\mathcal{Y}_1 \times \mathcal{Y}_2)^\infty$. To simplify further developments, we define the set $\mathcal{B}_\times(\mathcal{S}_1 \times \mathcal{S}_2)$, which contains the output trajectories of $\mathcal{S}_2$ under feedback composition with $\mathcal{S}_1$, as follows

$$\mathcal{B}_\times(\mathcal{S}_1 \times \mathcal{S}_2) = \pi_{\mathcal{Y}_2}(\mathcal{B}(\mathcal{S}_1 \times \mathcal{S}_2)), \tag{2.17}$$

where $\pi_{\mathcal{Y}_2}(\cdot)$ is defined in (1.1).

2.4 Control problem

In this section, we define the system specifications and introduce the optimal control problem, which includes a transition cost function. We also cover key dynamic programming concepts that will be used to address this optimal control problem.

2.4.1 Output specifications

We now introduce the concepts of *specification* and *control problem*.

Definition 2.9 (Specification)**.** Consider the system $\mathcal{S}$ in (2.4). A *specification* Σ for $\mathcal{S}$ is defined as any subset $\Sigma \subseteq \mathcal{Y}^\infty$. A system $\mathcal{S}$ together with a specification Σ constitute a *control problem* $(\mathcal{S}, \Sigma)$. Additionally, a controller $\mathcal{C}$ is said to *solve* the control problem $(\mathcal{S}, \Sigma)$ if $\mathcal{C}$ is f.c. with $\mathcal{S}$ and $\mathcal{B}_\times(\mathcal{C} \times \mathcal{S}) \subseteq \Sigma$. △

Linear Temporal Logic (LTL) [BK08, Chapter 5] can be used to define specification Σ for a system $\mathcal{S}$. We explicitly introduce two types of specifications that will be considered throughout this thesis: *reach-avoid* and *safety* specifications.

Definition 2.10. We consider a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$.
A *reach-avoid* specification associated with sets $\mathcal{I}, \mathcal{T}, \mathcal{O} \subseteq \mathcal{X}$ such that $\mathcal{T} \cap \mathcal{O} = \varnothing$ is defined as

$$\Sigma = \{\boldsymbol{x} \in \mathcal{X}^\infty \mid x(0) \in \mathcal{I} \Rightarrow \exists N \in \mathbb{Z}_{\geq 0} : (x(N) \in \mathcal{T} \wedge \forall k \in [0; N[: x(k) \notin \mathcal{O})\}, \quad (2.18)$$

which enforces that all states in the initial set $\mathcal{I}$ will reach the target $\mathcal{T}$ in finite time while avoiding obstacles in $\mathcal{O}$. We use the abbreviated notation $\Sigma^{\text{Reach}} = [\mathcal{I}, \mathcal{T}, \mathcal{O}]$ to denote the specification (2.18).
A *safety* specification associated with sets $\mathcal{I} \subseteq \mathcal{Z} \subseteq \mathcal{X}$ is defined as

$$\Sigma = \{\boldsymbol{x} \in \mathcal{X}^\infty \mid x(0) \in \mathcal{I} \Rightarrow \forall k \in [0; \infty[: x(k) \in \mathcal{Z}\}, \quad (2.19)$$

which enforces that all states in the initial set $\mathcal{I}$ will remain in the safety set $\mathcal{Z}$ indefinitely. We use the abbreviated notation $\Sigma^{\text{Safe}} = [\mathcal{I}, \mathcal{Z}]$ to denote the specification (2.19). △

2.4.2 Optimal control

In addition to specifying the output behavior of the closed-loop system, transition costs can be incorporated to address optimal control problems. We focus on the *optimal control problem* of designing a controller that enforces a reach-avoid specification while minimizing the worst-case cost.

Definition 2.11 (Optimal control problem)**.** Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, a reach-avoid specification $\Sigma^{\text{Reach}} = [\mathcal{I}, \mathcal{T}, \mathcal{O}]$, and a transition cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{>0} \cup \{\infty\}$, the *optimal control problem* involves designing an *optimal controller* $\mathcal{C}^*$ that is f.c. with $\mathcal{S}$ and minimizes the *cost function*

$$\mathcal{L}(\mathcal{C}) = \begin{cases} \sup_{x_0 \in \mathcal{I}} l_{\mathcal{C}}(x_0), & \text{if } \mathcal{C} \text{ solves } (\mathcal{S}, \Sigma^{\text{Reach}}), \\ \infty, & \text{otherwise.} \end{cases} \quad (2.20)$$

Here, $l_{\mathcal{C}}(x) : \mathcal{I} \to \mathbb{R}_{\geq 0}$ is defined as

$$l_{\mathcal{C}}(x_0) = \sup_{(\boldsymbol{u},\boldsymbol{x}) \in \mathcal{B}(\mathcal{C}\times\mathcal{S}) | x(0)=x_0} \sum_{i=0}^{N(\boldsymbol{x})-1} c(x(i), u(i)), \tag{2.21}$$

where $N(\boldsymbol{x}) = \min_{k \in \mathbb{Z}_{\geq 0} | x(k) \in \mathcal{T}} k$, i.e., $x(N)$ is the first occurrence in the sequence $\boldsymbol{x}$ that belongs to $\mathcal{T}$. △

The function $l_{\mathcal{C}}(x_0)$ represents the worst-case cumulative cost to reach the target set among the closed-loop trajectories starting in state x_0. The overall cost of a controller that solves the reach-avoid specification, $\mathcal{L}(\mathcal{C})$, is the worst-case cumulative cost among trajectories starting in the initial set. Note that $\mathcal{L}(\mathcal{C}^*) = \infty$ if and only if the control problem $(\mathcal{S}, \Sigma^{\text{Reach}})$ is infeasible.

2.5 Dynamic programming

Dynamic programming (DP) [Ber05, Chapter 1] is a general method for solving stochastic control problems. It relies on characterizing the value function, which represents the expected cost incurred by following an optimal policy from a given state. The optimal policy is then derived by solving an optimization problem that involves both the stage cost and the value function.

In many cases, however, computing or even representing the value function is computationally intractable. A common alternative is approximate dynamic programming (ADP) [Ber05, Chapter 6][BT96], where the value function is replaced with an *approximate value function*. The aim of ADP is to find an approximation that makes the policy evaluation feasible (e.g., through convex optimization) while ensuring near-optimal performance. ADP techniques often involve parameterizing a family of lower and upper bounds on the true value function. Ensuring these bounds is typically achieved using iterated Bellman inequalities [OWB13, WOB15], which are based on Bellman inequalities (2.24) [Ber05, Ber07, WOB15, Section 7.2] and are related to the 'linear programming approach' to ADP [Man60, SS85, DFVR03].

Since in this thesis our focus is on robust abstraction-based control, we provide in the next sections the worst-case scenario setting rather than the stochastic one.

2.5.1 Bellman value function

Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, a *value function* is a function $v : \mathcal{X} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$. Given a *transition cost function* $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{>0} \cup \{\infty\}$ and a value function v, we denote by $\mathcal{T}_c$ the *Bellman operator* defined as

$$[\mathcal{T}_c(v)](x) = \min_{u \in \mathcal{U}_{\mathcal{S}}(x)} \left(c(x,u) + \max_{x^+ \in F(x,u)} v(x^+) \right). \tag{2.22}$$

A value function v solving the Bellman equation [Bel57]

$$v(x) = [\mathcal{T}_c(v)](x), \tag{2.23}$$

is commonly referred to as a *Bellman value function*.

We can derive a static controller from a value function as follows.

Definition 2.12. Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, a transition cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{>0} \cup \{\infty\}$, and a value function $v : \mathcal{X} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$, we define the *associated controller* with v as the static system $\mathcal{C} = (\{0\}, \mathcal{X}, \mathcal{X}, \mathcal{U}, F_{\mathcal{C}}, H_{\mathcal{C}})$ such that

$$F_{\mathcal{C}}(0,x) = \begin{cases} \{0\} & \text{if } v(x) < \infty, \\ \emptyset & \text{otherwise.} \end{cases}$$

and

$$H_{\mathcal{C}}(0,x) = \{(u^*, x) \mid u^* \in \operatorname{argmin}_{u \in \mathcal{U}_{\mathcal{S}}(x)} \left(c(x,u) + \max_{x^+ \in F(x,u)} v(x^+) \right)\}.$$

The set of admissible inputs of the controller $\mathcal{C}$ at state 0 satisfies $\mathcal{U}_{\mathcal{C}}(0) = \{x \mid v(x) < \infty\}$. △

A Bellman value function can be used to construct a static optimal controller $\mathcal{C}^*$ that solves an optimal control problem.

Proposition 2.13. *Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, an optimal control problem with reach-avoid specification $\Sigma^{Reach} = [\mathcal{I}, \mathcal{T}, \mathcal{O}]$, and transition cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{>0} \cup \{\infty\}$ such that there exists $b > 0$ with $c(x,u) \geq b$ for all $x \in \mathcal{X}$ and all $u \in \mathcal{U}_{\mathcal{S}}(x)$. Let $v^* : \mathcal{X} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ be a Bellman value function such that $v^*(x) = 0$ if and only if $x \in \mathcal{T}$, $v^*(x) = \infty$ if $x \in \mathcal{O}$, and $v^*(x) > 0$ otherwise.*

The controller $\mathcal{C}^*$ associated with v^* (Definition 2.12) solves the optimal control problem $(\mathcal{S}, [\mathcal{U}_{\mathcal{C}^*}(0), \mathcal{T}, \mathcal{O}])$ with transition cost c. Moreover, if $\mathcal{I} \subseteq \mathcal{U}_{\mathcal{C}^*}(0)$, then $\mathcal{C}^*$ solves the optimal control problem $(\mathcal{S}, \Sigma^{Reach})$ with transition cost c.

Proof. Since v^* is a Bellman value function, it satisfies the Bellman equation $v^*(x) = [\mathcal{T}_c(v^*)](x)$ (2.23). Consequently, for the controller $\mathcal{C}^*$ associated with v^* (Definition 2.12), the worst-case cost $l_{\mathcal{C}^*}(x_0)$ in (2.21) satisfies $l_{\mathcal{C}^*}(x_0) = v^*(x_0)$. Additionally, for all $x \in \mathcal{U}_{\mathcal{C}^*}(0) = \{x \mid v^*(x) < \infty\}$, the controller $\mathcal{C}^*$ minimizes $l_{\mathcal{C}^*}(x)$ since v^* is a Bellman value function. In addition, for $(u, x) \in H_{\mathcal{C}^*}(0, x)$, it holds that $\max_{x^+ \in F(x,u)} v^*(x^+) = v^*(x) - c(x, u) \leq v^*(x) - b$ since $c(x, u) \geq b$. As a result, the function v^* decreases by at least b along trajectories of $\mathcal{C}^* \times \mathcal{S}$ for all $x \in \mathcal{U}_{\mathcal{C}^*}(0)$, converging to the minimum of v^*, which corresponds to the target set $\mathcal{T}$ while avoiding the obstacle $\mathcal{O}$. Therefore, $\mathcal{C}^*$ solves the optimal control problem as long as $\mathcal{I} \subseteq \mathcal{U}_{\mathcal{C}^*}(0)$. □

2.5.2 Suboptimal and superoptimal value functions

While the Bellman value function v solving the Bellman equation $v(x) = [\mathcal{T}_c(v)](x)$ (2.23) is the Grail of optimal control (Proposition 2.13), its computation is challenging. We define two types of value functions, the *suboptimal value functions* ([LBJ21, Definition 3]) and the *superoptimal value functions* ([LBJ21, Definition 7]), that satisfy inequalities instead [Ber05, Section 7.2][Ber07, WOB15],

$$v(x) \leq [\mathcal{T}_c(v)](x) \text{ and } v(x) \geq [\mathcal{T}_c(v)](x), \tag{2.24}$$

respectively.

As mentioned in the introduction of Section 2.5, these surrogate functions that satisfy a Bellman inequality are traditionally used to bound the optimal value function in stochastic control problems. In this thesis, we will show how to apply these functions within the context of abstraction-based control (Chapter 6), specifically how to translate them from the abstract system to the concrete one.

Definition 2.14 (Suboptimal value function)**.** Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, a value function $v : \mathcal{X} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ is a *suboptimal value function* with transition cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{>0} \cup \{\infty\}$, if for all $x \in \mathcal{X}$, $v(x) \leq [\mathcal{T}_c(v)](x)$. △

Definition 2.15 (Superoptimal value function)**.** Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$ and a set $\mathcal{X}_v \subseteq \mathcal{X}$, a value function $v : \mathcal{X} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ is a *superoptimal value function* with transition cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{>0} \cup \{\infty\}$, if for all $x \in \mathcal{X} \setminus \mathcal{X}_v$, $v(x) = \infty$, and for all $x \in \mathcal{X}_v$, $v(x)$ is finite and $v(x) \geq [\mathcal{T}_c(v)](x)$. △

The following propositions establish how suboptimal and superoptimal value functions can be combined.

Proposition 2.16 ([CLEJ21, Proposition 1])**.** *Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$ and two suboptimal value functions v_1 and v_2 with the same transition cost c, the value function $v = \max(v_1, v_2)$ is a suboptimal value function with transition cost c.*

Proof. For any $x \in \mathcal{X}$, by (2.22), we have

$$\begin{aligned}
[\mathcal{T}_c(v)](x) &= \min_{u \in \mathcal{U}_{\mathcal{S}}(x)} \left(c(x,u) + \max_{x^+ \in F(x,u)} v(x^+) \right) \\
&= \min_{u \in \mathcal{U}_{\mathcal{S}}(x)} \left(c(x,u) + \max_{x^+ \in F(x,u)} \max(v_1(x^+), v_2(x^+)) \right) \\
&= \min_{u \in \mathcal{U}_{\mathcal{S}}(x)} \left(c(x,u) + \max_{i \in \{1,2\}} \max_{x^+ \in F(x,u)} v_i(x^+) \right) \\
&= \max([\mathcal{T}_c(v_1)](x), [\mathcal{T}_c(v_2)](x)) \\
&\geq \max(v_1(x), v_2(x)) \\
&= v(x).
\end{aligned}$$

□

Proposition 2.17 ([CLEJ21, Proposition 2])**.** *Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$ and two superoptimal value functions v_1 and v_2 with the same transition cost c in $\mathcal{X}_v \subseteq \mathcal{X}$, the value function $v = \min(v_1, v_2)$ is a superoptimal value function with transition cost c in $\mathcal{X}_v$.*

Proof. For any $x \in \mathcal{X}_v$, by (2.22) and the max-min inequality, we have

$$\begin{aligned}
[\mathcal{T}_c(v)](x) &= \min_{u \in \mathcal{U}_S(x)} \left(c(x,u) + \max_{x^+ \in F(x,u)} v(x^+) \right) \\
&= \min_{u \in \mathcal{U}_S(x)} \left(c(x,u) + \max_{x^+ \in F(x,u)} \min(v_1(x^+), v_2(x^+)) \right) \\
&\leq \min_{u \in \mathcal{U}_S(x)} \left(c(x,u) + \min_{i \in \{1,2\}} \max_{x^+ \in F(x,u)} v_i(x^+) \right) \\
&= \min([\mathcal{T}_c(v_1)](x), [\mathcal{T}_c(v_2)](x)) \\
&\leq \min(v_1(x), v_2(x)) \\
&= v(x).
\end{aligned}$$

□

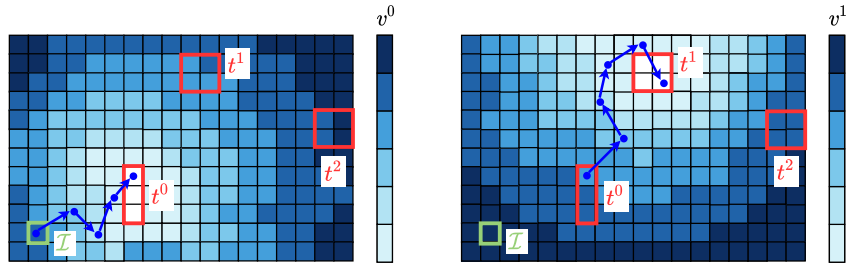
Just as a Bellman value function is used to construct an optimal controller (Proposition 2.13), a superoptimal value function can be employed to construct a static controller that *non-optimally* solves an optimal control problem.

Theorem 2.18. *Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, an optimal control problem with reach-avoid specification $\Sigma^{Reach} = [\mathcal{I}, \mathcal{T}, \mathcal{O}]$, and transition cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{>0} \cup \{\infty\}$ such that there exists $b > 0$ with $c(x,u) \geq b$ for all $x \in \mathcal{X}$ and all $u \in \mathcal{U}_S(x)$. Let $v : \mathcal{X} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ be a superoptimal value function such that $v(x) = 0$ if and only if $x \in \mathcal{T}$, $v(x) = \infty$ if $x \in \mathcal{O}$, and $v(x) > 0$ otherwise.*

The controller $\mathcal{C}$ associated with v (Definition 2.12) solves $(\mathcal{S}, [\mathcal{U}_\mathcal{C}(0), \mathcal{T}, \mathcal{O}])$. If $\mathcal{I} \subseteq \mathcal{U}_\mathcal{C}(0)$, then $\mathcal{C}$ solves $(\mathcal{S}, \Sigma^{Reach})$. Additionally, the function $v(x)$ provides an upper bound on the optimal worst-case cost from x and $\mathcal{L}(\mathcal{C}) \geq \mathcal{L}(\mathcal{C}^*)$ (Definition 2.11).

Proof. Since v is a superoptimal value function, for all $x \in \mathcal{X}_v$, where $\mathcal{X}_v = \{x \in \mathcal{X} \mid v(x) < \infty\}$, we have $v(x) \geq [\mathcal{T}_c(v)](x)$. For $(u,x) \in H_\mathcal{C}(0,x)$, it holds that $\max_{x^+ \in F(x,u)} v(x^+) \leq v(x) - c(x,u) \leq v(x) - b < v(x)$ since $c(x,u) \geq b$. As a result, the function v decreases by at least b along trajectories of $\mathcal{C} \times \mathcal{S}$ for all $x \in \mathcal{X}_v$, converging to the minimum of v, which corresponds to the target set $\mathcal{T}$ while avoiding the obstacle $\mathcal{O}$. □

Thus, the superoptimal value function provides an upper bound on the optimal worst-case cost of the closed-loop system.



(a) $x_{\mathcal{C}} = 0$. **(b)** $x_{\mathcal{C}} = 1$.

Fig. 2.4 Closed-loop behavior with the dynamical controller $\mathcal{C}$ (Definition 2.19) constructed from a sequence of superoptimal value functions $\boldsymbol{v} = (v^j)_{j=0}^2$ for solving an optimal control problem with reach-avoid specification $\Sigma^{\text{Reach}} = [\mathcal{I}, \mathcal{T}, \varnothing]$. The initial set $\mathcal{I}$ and the local targets t^0, t^1, and $t^2 \subseteq \mathcal{T}$ are shown. The piecewise constant superoptimal value functions v^0 and v^1 are depicted with the blue colormap in **(a)** and **(b)**, respectively. A closed-loop trajectory is illustrated, with the controller $\mathcal{C}$ in state $x_{\mathcal{C}} = 0$ in **(a)** and in state $x_{\mathcal{C}} = 1$ in **(b)**.

In order to reduce the computational cost of designing a controller for an optimal control problem, the overall problem can be divided into subproblems by enforcing the visit of intermediate objectives, as discussed in Section 7.3. To achieve this, we introduce a dynamical controller that generalizes the static controller from Definition 2.12 by associating it with a sequence of value functions rather than a single one.

Definition 2.19. Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, a transition cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$, a sequence of k value functions $\boldsymbol{v} = (v^j)_{j=0}^{k-1}$ for $\mathcal{S}$, and a sequence of local targets $(t^j)_{j=0}^{k-1}$ where $t^j \subseteq \mathcal{X}$, such that $v^j(x) = 0$ if $x \in t^j$, and $v^j(x) > 0$ otherwise. We define the *associated controller* with $\boldsymbol{v}$ as the dynamical system $\mathcal{C} = (\mathcal{X}_{\mathcal{C}}, \mathcal{X}, \mathcal{X}, \mathcal{U}, F_{\mathcal{C}}, H_{\mathcal{C}})$, with $\mathcal{X}_{\mathcal{C}} = \{0, 1, 2, \ldots, k-1\}$, where

$$F_{\mathcal{C}}(j, x) = \begin{cases} \{j\} & \text{if } 0 < v^j(x) < \infty \\ \{j+1\} & \text{if } v^j(x) = 0,\ j < k-1 \\ \varnothing & \text{otherwise} \end{cases}$$

and

$$H_{\mathcal{C}}(j,x) = \{(u^*,x) \mid u^* \in \operatorname{argmin}_{u \in \mathcal{U}_{\mathcal{S}}(x)} \left(c(x,u) + \max_{x^+ \in F(x,u)} v^j(x^+) \right)\}.$$

The set of admissible inputs at state j satisfies $\mathcal{U}_{\mathcal{C}}(j) = \{x \mid v^j(x) < \infty\}$. △

Note that if $k = 1$, Definition 2.19 reduces to the static controller in Definition 2.12.

A sequence of superoptimal value functions can be used to construct a dynamical controller that sub-optimally solves an optimal control problem by enforcing the visit of intermediate objectives, as shown in Figure 2.4.

Theorem 2.20. *Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, an optimal control problem with reach-avoid specification $\Sigma^{Reach} = [\mathcal{I}, \mathcal{T}, \mathcal{O}]$, and a transition cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{>0} \cup \{\infty\}$ such that there exists $b > 0$ with $c(x,u) \geq b$ for all $x \in \mathcal{X}$ and all $u \in \mathcal{U}_{\mathcal{S}}(x)$. Let $\boldsymbol{v} = (v^j)_{j=0}^{k-1}$ be a sequence of superoptimal value functions for $\mathcal{S}$ and $(t^j)_{j=0}^{k-1}$ be a sequence of local targets where $t^j \subseteq \mathcal{X}$, such that $v^j(x) = 0$ if $x \in t^j$, $v^j(x) = \infty$ if $x \in \mathcal{O}$, and $v^j(x) > 0$ otherwise.*

Let $\mathcal{C}$ be the controller associated with $\boldsymbol{v}$ (Definition 2.19). Define $\mathcal{X}_{\boldsymbol{v}}^j = \{x \mid v^j(x) < \infty\}$ for $j \in [0; k-1]$. If $t^j \subseteq \mathcal{X}_{\boldsymbol{v}}^{j+1}$ for all $j \in [0; k-2]$, $\mathcal{I} \subseteq \mathcal{X}_{\boldsymbol{v}}^0$, and $t^{k-1} \subseteq \mathcal{T}$, then the controller $\mathcal{C}$ initialized at state 0 solves $(\mathcal{S}, \Sigma^{Reach})$ by sequentially visiting each local target in $(t^j)_{j=0}^{k-1}$. Additionally, it holds that $\mathcal{L}(\mathcal{C}) \geq \mathcal{L}(\mathcal{C}^*)$ (Definition 2.11).

Proof. Apply Theorem 2.18 to each intermediate reach-avoid problem

$$[\mathcal{I}, t^0, \mathcal{O}] \text{ and } [t^j, t^{j+1}, \mathcal{O}] \text{ for } j \in [0; k-2].$$

□

2.6 Continuous-time systems

Although the system formalism introduced in Section 2.2 is discrete-time, it can also represent various types of systems, including sampled continuous-time systems. Consider a continuous-time system defined by

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), w(t)) \\ y(t) &= h(x(t), \delta(t)) \end{aligned} \tag{2.25}$$

where $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$, and $y(t) \in \mathcal{Y} \subseteq \mathbb{R}^p$ represent the state, control input, and measured output, respectively. The uncertainties are denoted by $w(t) \in \mathcal{W} \subseteq \mathbb{R}^{r_1}$ and $\delta(t) \in \mathcal{D} \subseteq \mathbb{R}^{r_2}$. The functions $f : \mathbb{R}^n \times \mathcal{U} \times \mathcal{W} \to \mathbb{R}^n$ and $h : \mathbb{R}^n \times \mathcal{D} \to \mathcal{Y}$ describe the system dynamics and output mapping, respectively.

As it will be illustrated in Section 3.2.2, we will focus on controlling continuous-time systems by measuring the state at regular time intervals and applying constant control inputs between these sampling times. Therefore, given $\tau \in \mathbb{R}_{>0}$ and an interval $I \subseteq [0, \tau]$, a *solution of* (2.25) *on* I *with constant input* $u \in \mathcal{U}$, i.e., $u(t) = u$ for all $t \in I$, is defined as an absolutely continuous function $\xi : I \to \mathbb{R}^n$ that satisfies (2.25) for almost every (a.e.) $t \in I$.

Definition 2.21 (τ-sampled system)**.** Given the continuous-time system in (2.25) and $\tau \in \mathbb{R}_{>0}$, the *τ-sampled system* of (2.25) is the Moore system $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, \mathcal{Y}_1, F_1, H_1)$ with $\mathcal{X}_1 = \mathcal{X}$, $\mathcal{U}_1 = \mathcal{U}$, $\mathcal{Y}_1 = \mathcal{Y}$, $H_1(x_1) = \{h(x_1, \delta) \mid \delta \in \mathcal{D}\}$, and $x_1^+ \in F_1(x_1, u_1)$ if and only if there exists a solution ξ of (2.25) on $[0, \tau]$ with input u_1 satisfying $\xi(0) = x_1$ and $\xi(\tau) = x_1^+$. △

Assuming that both $\mathcal{W}$ and $\mathcal{D}$ contain the origin, we define the *nominal system* of (2.25) as the unperturbed system

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), 0) \\ y(t) &= h(x(t), 0). \end{aligned} \tag{2.26}$$

Assuming that $f(\cdot, u, 0)$ is locally Lipschitz, given $x_0 \in \mathbb{R}^n$ and $u \in \mathcal{U}$, we denote by $\varphi_0(\cdot, x_0, u)$ the unique solution of (2.26) with constant input u starting at $x(0) = x_0$ [Har02].

2.7 Stability notions

Stability properties are described through the use of comparison functions. A continuous function $\gamma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\gamma(0) = 0$. It is said to belong to class $\mathcal{K}_\infty$ if it is a $\mathcal{K}$ function and $\gamma(r) \to +\infty$ when $r \to +\infty$. Let $\mathcal{L}$ denote the set of continuous strictly decreasing functions $\gamma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with $\gamma(r) \to 0$ when $r \to +\infty$. Then, we can denote $\beta \in \mathcal{KL}$ as functions where $\beta(\cdot, t) \in \mathcal{K}$ for all $t \geq 0$ and $\beta(r, \cdot) \in \mathcal{L}$ for all $r > 0$.

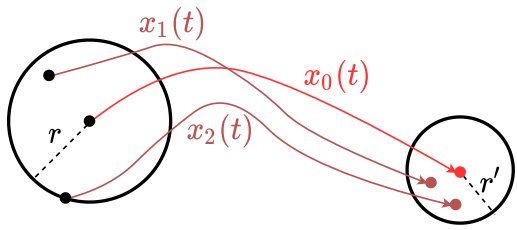


Fig. 2.5 Incremental stability. If the continuous-time system (2.25) is incrementally stable, the distance between trajectories decreases over time. Specifically, for any trajectories $x_0(t)$, $x_1(t)$, and $x_2(t)$, the inequality $\|x_0(t) - x_i(t)\|_2 \leq \beta(\|x_0(0) - x_i(0)\|_2, t)$ holds, where $i \in \{1, 2\}$.

The *incremental stability* is a property of dynamical systems that concerns the behavior of pairs of trajectories with respect to each other, rather than just the behavior of a single trajectory with respect to an equilibrium point. It ensures that the distance between any two trajectories decreases over time or remains bounded, regardless of their initial conditions, as illustrated in Figure 2.5.

We define the incremental stability for continuous-time systems (Section 2.6), but the concept can be similarly defined for discrete-time systems ([TRK16, Definition 1]) or switched systems ([GPT09, Definition 2.6]).

Definition 2.22 ([Ang02, Definition 2.1][LS98])**.** Consider a continuous-time dynamical system of the form (2.25), where the disturbance takes values in a closed set $\mathcal{W} \subseteq \mathbb{R}^{n_1}$. The system is said to be *incrementally globally asymptotically stable* (δ-GAS) if there exists a function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ of class $\mathcal{KL}$ such that for all $w \in \mathcal{W}$, and for any initial conditions $x_1(0), x_2(0) \in \mathcal{X}$, the solutions corresponding to the same input $u(t)$ satisfy

$$\|x_1(t) - x_2(t)\|_2 \leq \beta(\|x_1(0) - x_2(0)\|_2, t) \tag{2.27}$$

for all $t \geq 0$. △

In other words, the distance between the trajectories $x_1(t)$ and $x_2(t)$ can be bounded by a function that depends only on their initial distance and the time elapsed, and tends to zero as t increases.

Note that in the above definition, the condition holds for any pair of initial conditions in the entire state space. However, if this property is restricted to initial conditions within a specific region, the system is said to be *locally incrementally stable*.

3

Abstraction-based control

In this chapter, we formally define within our control formalism the abstraction-based approach that was briefly introduced earlier. We then delve into what we term the classical abstraction-based method, along with its inherent weaknesses. Finally, we outline our proposed solutions and explain how the thesis is organized to address these challenges.

3.1 Definition and properties

In this thesis, we mainly focus on performing *state-feedback* control of dynamical systems using abstraction-based techniques. Consequently, while the controllers and interconnected systems will be described using general systems of the form (2.4), the systems we aim to control and their abstractions are both described as *simple* systems (Definition 2.5).

We now translate the three steps of abstraction-based control (see Figure 1) described in the introduction into our mathematical framework.

Consider two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ and two controllers $\mathcal{C}_1$ and $\mathcal{C}_2$

$$\mathcal{S}_i = (\mathcal{X}_i, \mathcal{U}_i, F_i), \tag{3.1}$$
$$\mathcal{C}_i = (\mathcal{X}_{\mathcal{C}_i}, \mathcal{X}_i, \mathcal{V}_{\mathcal{C}_i}, \mathcal{U}_i, F_{\mathcal{C}_i}, H_{\mathcal{C}_i}). \tag{3.2}$$

In this context, $\mathcal{S}_1$ is referred to as the concrete system, while $\mathcal{S}_2$ is its abstraction. To solve the concrete control problem $(\mathcal{S}_1, \Sigma_1)$, the abstraction-based control procedure proceeds as follows.

Step 1: Abstract the system $\mathcal{S}_1$ into a system $\mathcal{S}_2$, establishing a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ between their states. Additionally, abstract the concrete specification into an abstract specification Σ_2, ensuring that

$$R^{-1}(\Sigma_2) \subseteq \Sigma_1. \tag{3.3}$$

Step 2: Design a controller $\mathcal{C}_2$ that solves the abstract problem $(\mathcal{S}_2, \Sigma_2)$, i.e., such that

$$\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2) \subseteq \Sigma_2. \tag{3.4}$$

Step 3: Construct a controller $\mathcal{C}_1$ that satisfies the *reproducibility* condition, as defined in [CMGJ24, Property 1], ensuring that

$$\mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2)). \tag{3.5}$$

This means that for every trajectory $\boldsymbol{x}_1$ defined on $[0; N[$ with $N \in \mathbb{N} \cup \{\infty\}$ of the closed-loop system $\mathcal{C}_1 \times \mathcal{S}_1$, there exists a trajectory $\boldsymbol{x}_2$ defined on $[0; N[$ of the closed-loop system $\mathcal{C}_2 \times \mathcal{S}_2$ satisfying

$$\forall k \in [0; N[\colon\ (x_1(k), x_2(k)) \in R. \tag{3.6}$$

The validity of this approach is established by the following inclusions, which guarantee that the controller $\mathcal{C}_1$ solves the concrete problem $(\mathcal{S}_1, \Sigma_1)$

$$\mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2)) \subseteq R^{-1}(\Sigma_2) \subseteq \Sigma_1,$$

which follows directly from (3.5), (3.4), (3.3), and the monotonicity of R^{-1}.

In order to be able to construct $\mathcal{C}_1$ from $\mathcal{C}_2$ satisfying (3.5), the relation R must impose conditions on the local dynamics of the systems in the associated states, accounting for the impact of different input choices on state transitions. The *alternating simulation relation* [Tab09, Definition 4.19] is a comprehensive definition of such a relation.

Definition 3.1 (Alternating simulation relation)**.** Given two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ is an *alternating simulation*

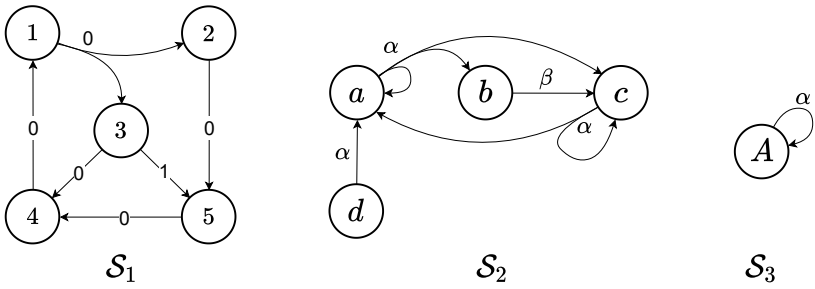


Fig. 3.1 Three simple systems $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$, $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$ and $\mathcal{S}_3 = (\mathcal{X}_3, \mathcal{U}_3, F_3)$. Specifically, $\mathcal{X}_1 = \{1,2,3,4,5\}$, $\mathcal{U}_1 = \{0,1\}$, $\mathcal{X}_2 = \{a,b,c,d\}$, $\mathcal{U}_2 = \{\alpha,\beta\}$, $\mathcal{X}_3 = \{A\}$, $\mathcal{U}_3 = \{\alpha\}$, the transition maps F_1, F_2 and F_3 and the available inputs maps $\mathcal{U}_1(.)$, $\mathcal{U}_2(.)$ and $\mathcal{U}_3(.)$ are clear from the illustration.

relation from $\mathcal{S}_1$ to $\mathcal{S}_2$, denoted $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$, if

$$\forall (x_1, x_2) \in R \; \forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) \; \exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1) \\ \forall x_1^+ \in F_1(x_1, u_1) \; \exists x_2^+ \in F_2(x_2, u_2) : \; (x_1^+, x_2^+) \in R. \quad (3.7)$$

△

Intuitively, an alternating simulation relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ describes which states of $\mathcal{S}_1$ are *simulated* by which states of $\mathcal{S}_2$. Thus, if $(x_1, x_2) \in R$, state x_1 of system $\mathcal{S}_1$ is simulated by state x_2 of system $\mathcal{S}_2$. Note that there is a clear order between the two systems as system $\mathcal{S}_2$ simulates system $\mathcal{S}_1$. Informally speaking, condition (3.7) requires the existence of inputs enforcing desired transitions robustly to the non-determinism of $\mathcal{S}_1$. This can also be given a game theoretic interpretation. In this two-player game, given inputs u_1 and u_2, system $\mathcal{S}_1$ selects a transition $x_1^+ \in F_1(x_1, u_1)$ that $\mathcal{S}_2$ must match with a transition $x_2^+ \in F_2(x_2, u_2)$, such that $(x_1^+, x_2^+) \in R$. If, for any input u_2, there exists an input u_1 allowing $\mathcal{S}_2$ to match all transition choices of player $\mathcal{S}_1$, then R is an alternating simulation relation.

Example 3.2. Consider the simple systems $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ as defined in Figure 3.1. We can verify that the relations $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ and $Q \subseteq \mathcal{X}_1 \times \mathcal{X}_3$, with

$$R = \{(1,a),(2,c),(3,b),(3,d),(4,a),(5,c)\},$$
$$Q = \{(1,A),(2,A),(3,A),(4,A),(5,A)\},$$

satisfy $\mathcal{S}_1 \preceq_R^{\mathrm{ASR}} \mathcal{S}_2$ and $\mathcal{S}_1 \preceq_Q^{\mathrm{ASR}} \mathcal{S}_3$. As we will see in the next section, abstraction-based control often entails designing abstractions that group concrete states together to reduce the number of states to manage, as illustrated by $\mathcal{S}_2$ and $\mathcal{S}_3$. However, while both $\mathcal{S}_2$ and $\mathcal{S}_3$ are abstractions of $\mathcal{S}_1$, $\mathcal{S}_3$ is likely not a useful abstraction due to its high level of aggregation. Typically, for a given specification Σ_1 for $\mathcal{S}_1$, it would be impossible to find a specification Σ_3 for $\mathcal{S}_3$ that meets condition (3.3). △

Note that condition (3.7) can equivalently be rewritten as

$$\forall (x_1, x_2) \in R \, \forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) \, \exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1) \\ \forall x_1^+ \in F_1(x_1, u_1) : \; R(x_1^+) \cap F_2(x_2, u_2) \neq \varnothing. \quad (3.8)$$

Note that Definition 3.1 slightly differs from [Tab09, Definition 4.19], where there is an additional requirement regarding the outputs of related states. This requirement is useful for the concept of approximate alternating simulation relation [Tab09, Definition 9.6] which we do not discuss here.

The relation $\preceq^{\mathrm{ASR}}$ is reflexive and transitive, which justifies the use of the pre-order symbol $\preceq$.

Proposition 3.3 ([Tab09, Proposition 4.23])**.** *Let $\mathcal{S}_1, \mathcal{S}_2$ and $\mathcal{S}_3$ be simple systems, and $R_{1,2}, R_{2,3}$ be relations. Then*

1. $\mathcal{S}_1 \preceq_I^{\mathrm{ASR}} \mathcal{S}_1$;
2. *If* $\mathcal{S}_1 \preceq_{R_{1,2}}^{\mathrm{ASR}} \mathcal{S}_2$ *and* $\mathcal{S}_2 \preceq_{R_{2,3}}^{\mathrm{ASR}} \mathcal{S}_3$, *then* $\mathcal{S}_1 \preceq_{R_{2,3} \circ R_{1,2}}^{\mathrm{ASR}} \mathcal{S}_3$.

The alternating simulation relation guarantees that any control applied to $\mathcal{S}_2$ can be concretized into a controller for $\mathcal{S}_1$ that maintains the relation between the closed-loop trajectories. This is formalized by the following theorem.

Theorem 3.4 ([Tab09, Proposition 8.7])**.** *Given two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$, if $\mathcal{S}_1 \preceq_R^{\mathrm{ASR}} \mathcal{S}_2$, then for every controller $\mathcal{C}_2$ f.c. with $\mathcal{S}_2$, there exists a controller $\mathcal{C}_1$ satisfying* (3.5).

Although the alternating simulation relation provides a safety-critical framework, it has several practical drawbacks in the implementation of the concrete controller $\mathcal{C}_1$, which we will discuss later (see Section 4.6). To address these issues, various alternative relations have been introduced in the literature. Among these, the *feedback refinement relation* [RWR16, Definition V.2] is central to the further developments of this thesis.

Definition 3.5 (Feedback refinement relation)**.** Given two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ is a *feedback refinement relation* from $\mathcal{S}_1$ to $\mathcal{S}_2$, denoted $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$, if for every $(x_1, x_2) \in R$

$$\begin{aligned} &(i)\ \mathcal{U}_{\mathcal{S}_2}(x_2) \subseteq \mathcal{U}_{\mathcal{S}_1}(x_1); \\ &(ii)\ \forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)\ \forall x_1^+ \in F_1(x_1, u_2) :\ R(x_1^+) \neq \varnothing \\ &\qquad \text{and}\ \ R(x_1^+) \subseteq F_2(x_2, u_2). \end{aligned} \tag{3.9}$$

△

The relation $\preceq^{\text{FRR}}$ is reflexive and transitive, which justifies the use of the pre-order symbol $\preceq$.

Proposition 3.6 ([RWR16, Proposition V.3])**.** *Let $\mathcal{S}_1, \mathcal{S}_2$ and $\mathcal{S}_3$ be simple systems, and $R_{1,2}, R_{2,3}$ be relations. Then*

1. $\mathcal{S}_1 \preceq_I^{\text{FRR}} \mathcal{S}_1$;
2. *If* $\mathcal{S}_1 \preceq_{R_{1,2}}^{\text{FRR}} \mathcal{S}_2$ *and* $\mathcal{S}_2 \preceq_{R_{2,3}}^{\text{FRR}} \mathcal{S}_3$, *then* $\mathcal{S}_1 \preceq_{R_{2,3} \circ R_{1,2}}^{\text{FRR}} \mathcal{S}_3$;

where $R_{2,3} \circ R_{1,2}$ is defined in (1.2).

The feedback refinement relation is a refined notion of the alternating simulation relation.

Proposition 3.7. *Given the simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in* (3.1) *and a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$, if $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$ then $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$.*

Proof. Given $(x_1, x_2) \in R$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, it follows from condition (i) that $u_2 \in \mathcal{U}_{\mathcal{S}_1}(x_1)$. For any $x_1^+ \in F_1(x_1, u_2)$, we have $R(x_1^+) \neq \varnothing$ and $R(x_1^+) \subseteq F_2(x_2, u_2)$. This implies that $R(x_1^+) \cap F_2(x_2, u_2) \neq \varnothing$. This establishes condition (3.8). □

As a refined notion of ASR, the feedback refinement relation benefits from a straightforward concretization scheme, involving the serial composition of the quantizer and the abstract controller, as established by the following theorem and illustrated in Figure 3.2.

Theorem 3.8 ([RWR16, Theorem V.4, Theorem V.5])**.** *Consider the simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in* (3.1)*. Given $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$, the following equivalence holds:*

(1) $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$;

(2) For all $\mathcal{C}_2$ f.c. with $\mathcal{S}_2$: $\mathcal{C}_2$ is f.c. with $R \circ \mathcal{S}_1$ (2.14)*, $\mathcal{C}_2 \circ R$ is f.c. with $\mathcal{S}_1$, and $\mathcal{B}_\times(\mathcal{C}_2 \times (R \circ \mathcal{S}_1)) \subseteq \mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2)$.*

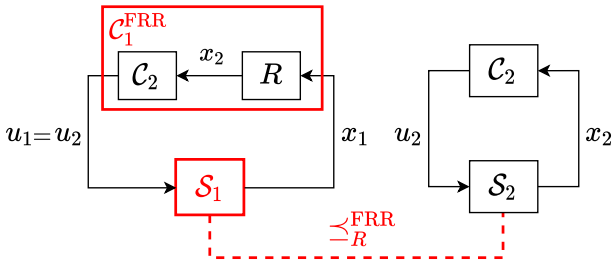


Fig. 3.2 Concretization procedure of the abstract controller for the feedback refinement relation.

Additionally, it implies that

$$\mathcal{C}_1^{\text{FRR}} := \mathcal{C}_2 \circ R, \tag{3.10}$$

satisfies $\mathcal{B}_\times(\mathcal{C}_1^{\text{FRR}} \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2))$.

3.2 Classical abstraction-based approach

Although the abstraction-based technique introduced in Section 3.1 does not theoretically require constructing a finite-state abstraction $\mathcal{S}_2$ (i.e., described by finite state, input, and output alphabets) of the original system $\mathcal{S}_1$, in practice, the usual procedure for abstraction-based controller design involves constructing a finite-state abstraction of the underlying dynamical system. This typically involves two steps: time discretization if the original system operates in continuous-time, and space discretization if the system has a continuous state space. The reasons for this approach are three-fold

1. It provides a common class of systems, namely discrete models, to represent all abstractions of various kinds of original systems, thereby offering a systematic way of designing correct-by-design controllers for diverse systems.
2. It leverages controller synthesis techniques developed for finite systems. Numerous automata-theoretic schemes, known as reactive synthesis, exist to algorithmically synthesize controllers that enforce complex specifications, possibly formulated in some temporal logic (see e.g. [Var95, BJP+12]). This allows for efficient resolution of step 2, i.e.,

abstract synthesis problem.

3. Time discretization facilitates the implementation of digital controllers that are easily manipulable by computers.

3.2.1 Classical abstraction

In this section, we discuss what we refer to as the classical abstraction-based approach (Algorithm 3.1) described in [RWR16]. This technique involves computing a feedback refinement relation (Definition 3.5) using a uniform grid discretization of the state space and a sampling of the input space.

In order to construct a finite state abstraction $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$ of $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$, we discretize the state space and sample the input space. The classical technique involves discretizing the state space using a uniform grid defined by

$$[\eta\mathbb{Z}^n] = \{c \in \mathbb{R}^n \mid \exists_{k\in\mathbb{Z}^n} \forall_{i\in[1;n]} c_i = k_i\eta_i\}, \tag{3.11}$$

with grid parameter $\eta \in \mathbb{R}^n_{>0}$. The abstract states correspond to the vertices of this lattice

$$\mathcal{X}_2 = [\eta\mathbb{Z}^n] \cap \mathcal{X}_1, \tag{3.12}$$

and the strict relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ simply defines the inclusion relation

$$R = \{(x_1, x_2) \in \mathcal{X}_1 \times \mathcal{X}_2 \mid x_1 \in \mathrm{H}(x_2, \tfrac{\eta}{2})\}. \tag{3.13}$$

The abstract input space $\mathcal{U}_2 \subseteq \mathcal{U}_1$ is a finite subset of the input space of $\mathcal{S}_1$.

The abstraction $\mathcal{S}_2$ is constructed such that $\mathcal{S}_1 \preceq_R^{\mathrm{FRR}} \mathcal{S}_2$ over the entire state space (Line 2), independently of the specification. Then, the specification is abstracted according to (3.3) (Line 3), and the abstract problem is solved (Line 4). Finally, a controller $\mathcal{C}_1$ for the concrete system is returned (Line 5).

Theorem 3.9. *Algorithm 3.1 returns a controller $\mathcal{C}_1$ that solves $(\mathcal{S}_1, \Sigma_1)$ if $\mathcal{C}_2$ solves $(\mathcal{S}_2, \Sigma_2)$.*

Proof. We have $\mathcal{S}_1 \preceq_R^{\mathrm{FRR}} \mathcal{S}_2$ and $R^{-1}(\Sigma_2) \subseteq \Sigma_1$. By Theorem 3.8, $\mathcal{C}_1 = \mathcal{C}_2 \circ R$ ensures that $\mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2))$. Therefore, if $\mathcal{C}_2$ solves $(\mathcal{S}_2, \Sigma_2)$, i.e., $\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2) \subseteq \Sigma_2$, then

$$\mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2)) \subseteq R^{-1}(\Sigma_2) \subseteq \Sigma_1,$$

Algorithm 3.1: Classical abstraction-based algorithm.

```
1 Function ClassicalAbstraction(S_1, Σ_1, R):
2     compute S_2;
3     Σ_2 ← get_abstract_specification(Σ_1, R);
4     C_2 ← solve(S_2, Σ_2);
5     C_1 ← C_2 ∘ R;
6     return C_1;
  end
```

i.e., $\mathcal{C}_1$ solves $(\mathcal{S}_1, \Sigma_1)$. □

3.2.2 Specific implementation

To illustrate the full process, we present a specific implementation of Line 2 of Algorithm 3.1 for a continuous-time system with continuous state and input spaces.

Consider the uncertain continuous-time system

$$\dot{x} = f(x(t), u(t)) + w(t), \tag{3.14}$$

where $f : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^n$, $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$, and $w(t) \in \mathcal{W} \subseteq \mathbb{R}^n$. The set $\mathcal{W}$ contains the origin and is bounded.

Since we are interested in controlling system (3.14) through a digital controller, the controls are assumed to be piecewise constant functions from $\mathbb{R}_{\geq 0}$ to $\mathcal{U}$. Given $\tau > 0$, $u(t) = u(k\tau)$ for all $t \in [k\tau, (k+1)\tau)$ and $k \in \mathbb{N}_0$. Let $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$ be the τ-sampled system of (3.14) (see Definition 2.21).

To apply Algorithm 3.1, we have to construct an abstraction related to the original system through a feedback refinement relation R. According to Definition 3.5, this requires adding to the transition map $F_2(x_2, u_2)$ all cells that intersect with the image $F_1(R^{-1}(x_2), u_2)$. However, computing the exact attainable set can be challenging or costly due to the nonlinear nature of f. Therefore, abstraction-based techniques often compute an over-approximation of the attainable sets for a particular cell $R^{-1}(x_2)$.

A variety of over-approximation methods have been proposed for different classes of systems and sets, such as those found in [Osi06, Alt10, Rei11, WR14, RWR16].

Using a grid-based discretization, a natural approach involves comput-

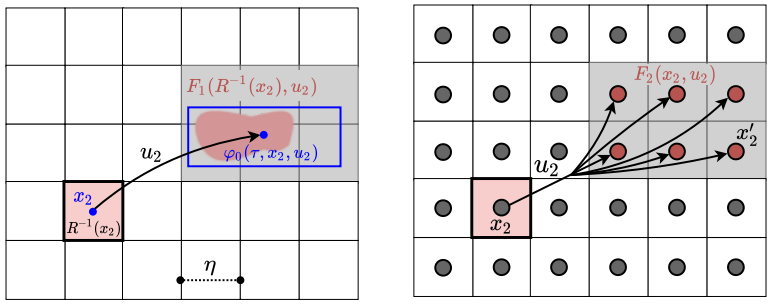


Fig. 3.3 Illustration of the transition function of an abstraction for system (3.14).

ing hyperrectangle over-approximations of the image under the dynamics (3.14) for each cell $R^{-1}(x_2)$ and abstract input $u_2 \in \mathcal{U}_2$. This is achieved using a *growth-bound function* [RWR16, Definition VIII.2][ZPMT11], which provides a bound on the distance between two trajectories based on the distance of their initial conditions.

Definition 3.10 (Growth-bound function)**.** Given sets $\mathcal{X}' \subseteq \mathbb{R}^n$, $\mathcal{U}' \subseteq \mathcal{U}$, and the sampling time τ, a map $\beta : \mathbb{R}^n_{\geq 0} \times \mathcal{U}' \to \mathbb{R}^n_{\geq 0}$ is a *growth-bound function* on $\mathcal{X}'$, $\mathcal{U}'$ associated with the τ-sampled system of (3.14) if the following conditions hold:

(i) $\beta(r, u) \geq \beta(r', u)$ whenever $r \geq r'$ and $u \in \mathcal{U}'$;

(ii) If ξ is a solution of (3.14) on $[0, \tau]$, with input $u \in \mathcal{U}'$ and $\xi(0), p \in \mathcal{X}'$, then

$$|\xi(\tau) - \varphi_0(\tau, p, u)| \leq \beta(|\xi(0) - p|, u) \tag{3.15}$$

holds component-wise. △

Condition (i) ensures that the closer the initial conditions of two trajectories, the smaller the bound provided by the growth-bound function β. Condition (ii) provides an upper bound on the final distance between two trajectories based on the distance between their initial conditions. However, unlike the concept of incremental stability (Definition 2.22), we do not require that trajectories converge to the same reference trajectory regardless of their initial conditions.

As an example, in [RWR16, Theorem VIII.5], the authors propose a growth-bound function for the system (3.14) based on a matrix-valued Lipschitz inequality.

Given a growth-bound function β, an abstract state $x_2 \in \mathcal{X}_2$, and an abstract input $u_2 \in \mathcal{U}_2$, one can compute over-approximation of the cell related to x_2 as follows

$$F_1(R^{-1}(x_2), u_2) \subseteq \mathrm{H}(\varphi_0(\tau, x_2, u_2), \beta(\tfrac{\eta}{2}, u_2)). \tag{3.16}$$

This requires computing the unique solution of the nominal system with the initial condition corresponding to the center of the cell and then using the growth-bound function to obtain a bound of trajectories at time τ, given the bound on the initial condition $x_1 \in R^{-1}(x_2) \Leftrightarrow |x_2 - x_1| \leq \frac{\eta}{2}$, as illustrated in Figure 3.3.

The following theorem establishes the validity of the abstraction-based approach described in this section.

Theorem 3.11 ([RWR16, Theorem VIII.4])**.** *Consider the system $\mathcal{S}_1$, which is the τ-sampled system of (3.14) with $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$, where $\mathcal{X}_1$ is a compact set. Define the system $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$, where $\mathcal{X}_2$ is given by (3.11), $\mathcal{U}_2$ is a finite subset of $\mathcal{U}_1$, and the transition map F_2 satisfies $\forall x_2 \in \mathcal{X}_2 \; \forall u_2 \in \mathcal{U}_2$:*

$$\mathrm{H}(\varphi_0(\tau, x_2, u_2), \beta(\tfrac{\eta}{2}, u_2)) \cap R^{-1}(x_2') \neq \emptyset \Rightarrow x_2' \in F_2(x_2, u_2). \tag{3.17}$$

This construction ensures that $\mathcal{S}_2$ is a finite-state system and $\mathcal{S}_1 \preceq_R^{\mathrm{FRR}} \mathcal{S}_2$.

Once the abstraction is constructed, the concrete specification Σ_1 must also be abstracted into Σ_2 to satisfy (3.3) (Line 3). The following proposition provides the conditions for reach-avoid and safety specifications (Definition 2.10).

Proposition 3.12. *Given simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ as in (3.1) and a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$, consider the reach-avoid specification $\Sigma_1^{Reach} = [\mathcal{I}_1, \mathcal{T}_1, \mathcal{O}_1]$ and the safety specification $\Sigma_1^{Safe} = [\mathcal{I}_1, \mathcal{Z}_1]$ for $\mathcal{S}_1$. The corresponding abstract specifications for $\mathcal{S}_2$ are*

- $\Sigma_2^{Reach} = [\mathcal{I}_2, \mathcal{T}_2, \mathcal{O}_2]$ *with* $\mathcal{I}_1 \subseteq R^{-1}(\mathcal{I}_2)$, $R^{-1}(\mathcal{T}_2) \subseteq \mathcal{T}_1$, *and* $\mathcal{O}_1 \subseteq R^{-1}(\mathcal{O}_2)$;
- $\Sigma_2^{Safe} = [\mathcal{I}_2, \mathcal{Z}_2]$ *with* $\mathcal{I}_1 \subseteq R^{-1}(\mathcal{I}_2)$, *and* $R^{-1}(\mathcal{Z}_2) \subseteq \mathcal{Z}_1$.

These abstract specifications satisfy (3.3), ensuring that $R^{-1}(\Sigma_2^{Reach}) \subseteq \Sigma_1^{Reach}$ and $R^{-1}(\Sigma_2^{Safe}) \subseteq \Sigma_1^{Safe}$.

Algorithm 3.2:
Symbolic Reach-Avoid Solver.

$\mathcal{Y} \leftarrow \emptyset$;
$\mathcal{Y}' \leftarrow G(\mathcal{Y})$;
while $\mathcal{Y}' \neq \mathcal{Y}$ **do**
  for $x_2 \in \mathcal{Y}'$ **do**
    $H_{\mathcal{C}_2}(0, x_2) \coloneqq \{(u_2, x_2) \mid u_2 \in \mathcal{U}_S(x_2), F_2(x_2, u_2) \subseteq \mathcal{Y}\}$;
  end
  $\mathcal{Y} \leftarrow \mathcal{Y}'$;
  $\mathcal{Y}' \leftarrow G(\mathcal{Y})$;
end
$\mathcal{Y}_\infty \leftarrow \mathcal{Y}$;
$F_{\mathcal{C}_2}(0, x_2) \coloneqq \begin{cases} \{0\} & \text{for } x_2 \in \mathcal{Y}_\infty \\ \emptyset & \text{for } x_2 \in \mathcal{X}_2 \setminus \mathcal{Y}_\infty \end{cases}$
return
$\mathcal{C}_2 = (\{0\}, \mathcal{X}_2, \mathcal{X}_2, \mathcal{U}_2, F_{\mathcal{C}_2}, H_{\mathcal{C}_2})$;

Algorithm 3.3:
Symbolic Safety Solver.

$\mathcal{Y} \leftarrow \mathcal{X}_2$;
$\mathcal{Y}' \leftarrow G'(\mathcal{Y})$;
while $\mathcal{Y}' \neq \mathcal{Y}$ **do**
  $\mathcal{Y} \leftarrow \mathcal{Y}'$;
  $\mathcal{Y}' \leftarrow G'(\mathcal{Y})$;
end
for $x_2 \in \mathcal{Y}$ **do**
  $H_{\mathcal{C}_2}(0, x_2) \coloneqq \{(u_2, x_2) \mid u_2 \in \mathcal{U}_S(x_2), F_2(x_2, u_2) \subseteq \mathcal{Y}\}$;
end
$\mathcal{Y}_\infty \leftarrow \mathcal{Y}$;
$F_{\mathcal{C}_2}(0, x_2) \coloneqq \begin{cases} \{0\} & \text{for } x_2 \in \mathcal{Y}_\infty \\ \emptyset & \text{for } x_2 \in \mathcal{X}_2 \setminus \mathcal{Y}_\infty \end{cases}$
return
$\mathcal{C}_2 = (\{0\}, \mathcal{X}_2, \mathcal{X}_2, \mathcal{U}_2, F_{\mathcal{C}_2}, H_{\mathcal{C}_2})$;

Fig. 3.4 Fixed point algorithms for reach-avoid and safety specifications. Given a finite-system $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$, it returns a static controller $\mathcal{C}_2$. **Algorithm 3.2:** Such that $\mathcal{U}_{\mathcal{C}_2}(0)$ is the largest subset of $\mathcal{X}_2$ such that $\mathcal{C}_2$ solves the abstract problem $(\mathcal{S}_2, \Sigma_2^{\text{Reach}})$ with $\Sigma_2^{\text{Reach}} = [\mathcal{U}_{\mathcal{C}_2}(0), \mathcal{T}_2, \mathcal{O}_2]$. **Algorithm 3.3:** Such that $\mathcal{U}_{\mathcal{C}_2}(0)$ is the largest subset of $\mathcal{X}_2$ such that $\mathcal{C}_2$ solves the abstract problem $(\mathcal{S}_2, \Sigma_2^{\text{Safe}})$ with $\Sigma_2^{\text{Safe}} = [\mathcal{U}_{\mathcal{C}_2}(0), \mathcal{U}_{\mathcal{C}_2}(0)]$. The functions G and G' are defined in (3.18) and (3.19), respectively. Note that in both cases, $\mathcal{U}_{\mathcal{C}_2}(0) = \mathcal{Y}_\infty$, which is the solution of the fixed-point equation.

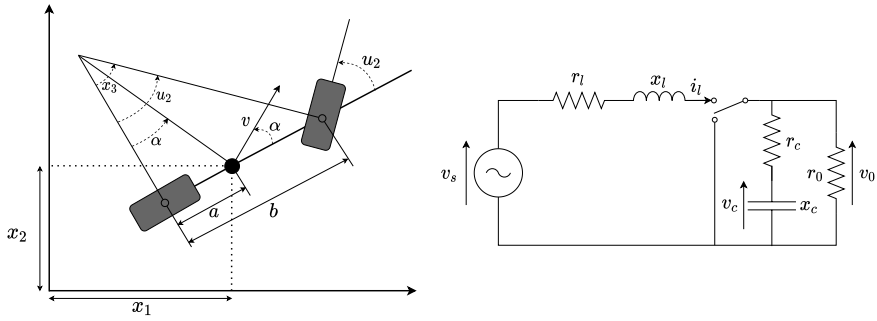
Next, Line 4 of Algorithm 3.1 involves solving the abstract problem $(\mathcal{S}_2, \Sigma_2)$. This can be approached using graph-theoretic tools since the abstract system is a hypergraph.

For reach-avoid specifications (Definition 2.10), the abstract problem $(\mathcal{S}_2, \Sigma_2^{\text{Reach}})$ with $\Sigma_2^{\text{Reach}} = (\mathcal{I}_2, \mathcal{T}_2, \mathcal{O}_2)$, can be solved by finding the fixed point of the following set-valued function

$$G(\mathcal{Y}) = (\text{pre}_{\mathcal{S}_2}(\mathcal{Y}) \cap (\mathcal{X}_2 \setminus \mathcal{O}_2)) \cup \mathcal{T}_2, \tag{3.18}$$

where $\text{pre}_{\mathcal{S}_2}$ is defined in (2.11). The abstract problem admits a solution $\mathcal{C}_2$ if and only if the static controller $\mathcal{C}_2$ returned by Algorithm 3.2 satisfies $\mathcal{I}_2 \subseteq \mathcal{U}_{\mathcal{C}_2}(0)$, and then $\mathcal{C}_2$ solves the abstract problem [RZ16].

For safety specifications (Definition 2.10), the abstract problem $(\mathcal{S}_2, \Sigma_2^{\text{Safe}})$ with $\Sigma_2^{\text{Safe}} = (\mathcal{I}_2, \mathcal{Z}_2)$, can be solved by finding the fixed point of the fol-



(a) Bicycle model [ÅM07, Example 2.8]. **(b)** DC-DC converter [BPM05, Section II].

Fig. 3.5 Modeling of Example 3.13 and Example 3.14, respectively.

lowing set-valued function

$$G'(\mathcal{Y}) = \mathrm{pre}_{\mathcal{S}_2}(\mathcal{Y}) \cap \mathcal{Z}_2. \tag{3.19}$$

The abstract problem admits a solution $\mathcal{C}_2$ if and only if the controller $\mathcal{C}_2$ returned by Algorithm 3.3 satisfies $\mathcal{I}_2 \subseteq \mathcal{U}_{\mathcal{C}_2}(0) \subseteq \mathcal{Z}_2$, and then $\mathcal{C}_2$ solves the abstract problem [RZ16].

Finally, if the abstract problem has a solution $\mathcal{C}_2$, then the controller $\mathcal{C}_1 = \mathcal{C}_2 \circ R$ (Line 5) solves the concrete problem $(\mathcal{S}_1, \Sigma_1)$.

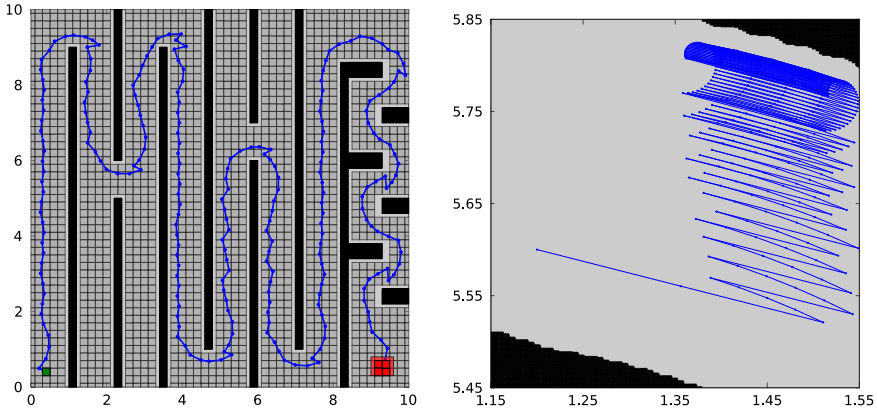
3.2.3 Examples

We illustrate the classical control approach (Algorithm 3.1) presented in the previous section using two control problems as running examples: a reach-avoid problem for a bicycle [RZ16, Section 4.1] and a safety problem for a DC-DC converter [RZ16, Section 4.2].

We start with the following reach-avoid problem for a nonlinear continuous-time system.

Example 3.13 (Path planning problem). We consider a bicycle vehicle (Figure 3.5a with $a = 0.5$, $b = 1$) modeled by the system (3.14), where $f : \mathbb{R}^3 \times \mathcal{U} \to \mathbb{R}^3$ is given by

$$f(x, (u_1, u_2)) = \begin{pmatrix} u_1 \cos(\alpha + x_3) \cos(\alpha)^{-1} \\ u_1 \sin(\alpha + x_3) \cos(\alpha)^{-1} \\ u_1 \tan(u_2) \end{pmatrix},$$



(a) Example 3.13. **(b)** Example 3.14.

Fig. 3.6 Algorithm 3.1 for Example 3.13 and Example 3.14. A closed-loop trajectory of the concrete sampled system is represented in blue. Left: Projection of the states of $\mathcal{S}_1$ and $\mathcal{S}_2$ to $\mathbb{R}^2 \times \{0\}$. The initial, target, and obstacle sets are represented in green, red, and black, respectively. Right: The largest invariant set $\mathcal{Z}_1 = \mathcal{I}_1$ is represented in light gray, while the non-invariant part is illustrated in black.

with $\mathcal{X} = [0,10]^2$, $\mathcal{U} = [-1,1]^2$ and $\alpha = \arctan(\tan(u_2)/2)$. There is no perturbation ($\mathcal{W} = \{(0,0,0)^\top\}$). Here, (x_1, x_2) represents the position of the center of mass and x_3 the orientation of the vehicle in the two-dimensional plane. The control inputs $u_1 = \|v\|_2$ and u_2 are the rear wheel velocity and the steering angle. We consider the reach-avoid specification $\Sigma_1^{\text{Reach}} = [\mathcal{I}_1, \mathcal{T}_1, \mathcal{O}_1]$ (Definition 2.10), with $\mathcal{I}_1 = \{(0.4, 0.4, 0)\}$, $\mathcal{T}_1 = \mathrm{H}((9.25, 9.25, 0), (0.25, 0.25, \mathbb{R}))$, and $\mathcal{O}_1$ illustrated in green, red, and black respectively in Figure 3.6a. △

It can be shown that the function $\beta(r,u) = e^{\mathbf{L}(u)\tau} r$, where $\mathbf{L}$ is defined by $L_{1,3}(u_1,u_2) = L_{2,3}(u_1,u_2) = |u_1\sqrt{\tan^2(u_2)/4+1}|$ and $L_{i,j}(u_1,u_2) = 0$ for $(i,j) \notin \{(1,3),(2,3)\}$ (see [RWR16, Examples IX.A]), is a growth-bound function associated with the τ-sampled system $\mathcal{S}_1$ from Example 3.13.

Consequently, the previously described approach can be applied to construct an abstraction $\mathcal{S}_2$ of $\mathcal{S}_1$ such that $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$, where R is the relation defined by the grid parameters η (3.13). The abstract specification Σ_2^{Reach} is constructed according to Proposition 3.12. A controller $\mathcal{C}_2$ solving $(\mathcal{S}_2, \Sigma_2^{\text{Reach}})$ can be computed with Algorithm 3.2. Finally, the controller $\mathcal{C}_1 = \mathcal{C}_2 \circ R$ solves $(\mathcal{S}_1, \Sigma_1^{\text{Reach}})$ (Theorem 3.9). The resulting abstraction is illustrated in Figure 3.6a[1].

We pursue with the following safety problem for a switch linear system.

Example 3.14 (DC-DC boost converter). We consider a DC-DC boost converter (Figure 3.5b) modeled by the system (3.14), where $f : \mathbb{R}^2 \times \mathcal{U} \to \mathbb{R}^2$ is given by

$$f(x,u) = \mathbf{A}_u x + b_u$$

with $x = (i_l, v_c)^\top$, $\mathcal{U} = \{1,2\}$, $\mathcal{W} = \{(0,0)\}$, $b_1 = b_2 = \left(\frac{v_s}{x_l}, 0\right)^\top$, and

$$\mathbf{A}_1 = \begin{pmatrix} -\frac{r_l}{x_l} & 0 \\ 0 & -\frac{1}{x_c}\frac{1}{r_0+r_c} \end{pmatrix}, \mathbf{A}_2 = \begin{pmatrix} -\frac{1}{x_l}\left(r_l + \frac{r_0 r_c}{r_0+r_c}\right) & -\frac{1}{x_l}\frac{r_0}{r_0+r_c} \\ \frac{1}{x_c}\frac{r_0}{r_0+r_c} & -\frac{1}{x_c}\frac{1}{r_0+r_c} \end{pmatrix}.$$

The control action $u \in \{1,2\}$ is the position of the switch, which affects the system's dynamics. We consider the safety specification $\Sigma_1^{\text{safe}} = [\mathcal{I}_1, \mathcal{Z}_1]$ (Definition 2.10), where we aim to determine the largest invariant set $\mathcal{Z}_1 = \mathcal{I}_1$ within the domain $\mathcal{X} = [1.15, 1.55] \times [5.45, 5.85]$ illustrated in Figure 3.6b. We use the numerical values from [BPM05], i.e., $x_c = 70$, $x_l = 3$, $r_c = 0.005$, $r_l = 0.05$, $r_0 = 1$ and $v_s = 1$. △

[1] These numerical experiments are available in the `Dionysos.jl` Julia package https://github.com/dionysos-dev/Dionysos.jl/releases/tag/2024_LCSS in the subfolder `docs/src/examples/solvers`.

The function $\beta(r, u) = e^{\mathbf{L}(u)\tau} r$, where $\mathbf{L}(1) = \mathbf{A}_1$ and $\mathbf{L}(2) = \mathbf{A}_2$, is a growth-bound function associated with the τ-sampled system $\mathcal{S}_1$ from Example 3.14.

Consequently, the previously described approach can be applied to construct an abstraction $\mathcal{S}_2$ of $\mathcal{S}_1$ such that $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$, where R is the relation defined by the grid parameters η (3.13). The abstract specification Σ_2^{Safe} is constructed according to Proposition 3.12. A controller $\mathcal{C}_2$ solving $(\mathcal{S}_2, \Sigma_2^{\text{Safe}})$ can be computed with Algorithm 3.3. Finally, the controller $\mathcal{C}_1 = \mathcal{C}_2 \circ R$ solves $(\mathcal{S}_1, \Sigma_1^{\text{Safe}})$ (Theorem 3.9). The resulting abstraction is illustrated in Figure 3.6b.

It can be shown that the switched system $\mathcal{S}_1$ is incrementally stable ([GPT09, Definition 2.6]). This stability property can be exploited to construct a deterministic feedback refinement abstraction $\mathcal{S}_2$ of $\mathcal{S}_1$, which allows for faster computation, as highlighted in Section 9.5.

The two examples are of different nature: Example 3.13 addresses a reach-avoid problem for a continuous-time non-linear system with a continuous input space, while Example 3.14 addresses a safety problem for a switched linear system with a discrete input space.

3.3 Weaknesses

As mentioned in Section 3.2, the classical abstraction-based approach relies on constructing a feedback refinement relation R using a predefined uniform partition of the entire state space. However, this method has two significant weaknesses:

1. **Non-determinism of the abstraction**
   Abstractions are constructed on a partition of the state space using non-empty compact hyper-intervals and over-approximations of the attainable sets under the system dynamics. While effective computation of these over-approximations is possible—typically involving simulation (e.g., via Runge-Kutta methods) at the center of each cell and bounding with a growth-bound function—this approach may yield conservative abstractions. Though over-approximating the attainable set ensures that all behaviors of $\mathcal{S}_1$ are captured within the abstraction (3.5), it may also introduce non-deterministic trajectories in the abstraction that do not correspond to any real trajectory in the

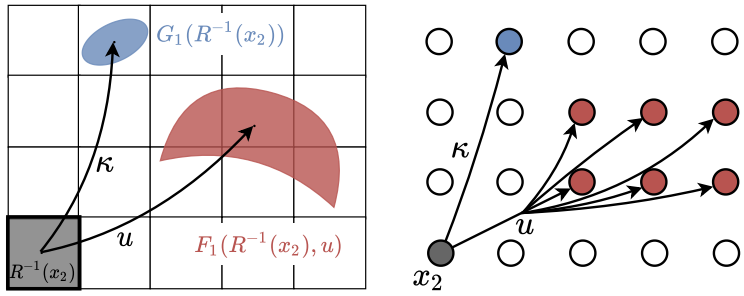


Fig. 3.7 Comparison of piecewise constant and state-dependent controllers. The red region illustrates $F_1(x_1, u)$ for all $x_1 \in R^{-1}(x_2)$, where $u \in \mathcal{U}_{S_2}(x_2)$ and $u \in \mathcal{U}_{S_1}(R^{-1}(x_2))$. The blue region shows $G_1(x_1) = F_1(x_1, \kappa(x_1))$ for all $x_1 \in R^{-1}(x_2)$, where $\kappa \in \mathcal{U}_{S_2}(x_2)$ is a local state-dependent controller $\kappa : \mathcal{X}_1 \to \mathcal{U}_1$. Left: The two-dimensional concrete system with its state space discretization. Right: The corresponding abstract system, highlighting the non-deterministic transition $F_2(x_2, u)$ and the deterministic transition $F_2(x_2, \kappa)$.

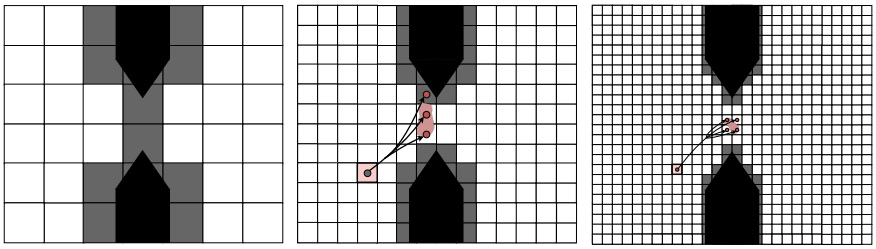


Fig. 3.8 Three levels of discretization for a two-dimensional system, ranging from coarse to fine. The concrete obstacles $\mathcal{O}_1$ are shown in black, while the abstract obstacles $\mathcal{O}_2$ are depicted in grey. Additionally, the image of a cell is illustrated.

original system. For example, a transition from x_2 to x_2' under input u_2 in Figure 3.3 does not match any transition from $R^{-1}(x_2)$ to $R^{-1}(x_2')$ under input u_2 in $\mathcal{S}_1$.

Thus, there is a trade-off: a fine over-approximation, while potentially more accurate, can be computationally expensive, whereas a coarser and faster over-approximation may introduce additional non-determinism into the abstraction, as illustrated in Figure 3.3. If the system lacks local incremental stability (Definition 2.22) within a cell, even precise over-approximations may still result in significant non-determinism when using piecewise constant controllers, leading to a high cardinality of the set $F_2(x_2, u_2)$, as shown in Figure 3.7.

2. **Curse of dimensionality**
The use of uniform grids leads to an exponential increase in the number of cells as the state-space dimension grows, making abstraction construction potentially intractable. Additionally, some regions of the state space may require finer discretization, leading to a rapid increase in the number of cells. As illustrated in Figure 3.8, obstacles can divide the state space into disjoint regions with coarse discretization. While medium discretization might suggest possible transitions between regions, the non-determinism introduced by discretization could still prevent guaranteed safe transitions. Only the finest partition allows safe transitions, but at the cost of greatly increased computational effort.

In summary, the combination of piecewise constant controllers and a uniform, predefined partition of the entire state space can lead to intractable oreven unsolvable abstract problems $(\mathcal{S}_2, \Sigma_2)$.

3.4 Solutions

This thesis aims to address the identified challenges through both theoretical and algorithmic developments, as well as practical implementation tools, as outlined below.

1. **Non-determinism of the abstraction**
To mitigate the non-determinism resulting from the discretization of the state, we propose to avoid discretizing the input space. Instead,

we leverage the entire input space to design state-dependent controllers that can either reduce non-determinism or ensure deterministic transitions, as is illustrated in Figure 3.7 with the local controller $\kappa \in \mathcal{U}_2(x_2)$. This approach requires a comprehensive analysis of the link between steps 1 and 3 of the abstraction procedure (see Figure 1), i.e., between abstraction relations and the resulting concrete control architecture, enabling the design of more general templates for concrete controllers.

This approach is detailed in Part II, in Chapters 4 and 5, which provide a theoretical framework for classifying and characterizing abstraction relations in terms of their concretization procedures.

2. **Curse of dimensionality**
To mitigate the computational challenges posed by high-dimensional systems, we propose a tailored approach to abstraction construction that directly addresses the specific control problem. This involves integrating steps 1 and 2 of the traditional abstraction-based procedure (see Figure 1). By co-designing the abstraction and the abstract controller guided by the optimal control problem, we can construct the abstraction incrementally, focusing only on the necessary regions of the state space, thereby reducing computational complexity.

   This solution is explored in Part III. Chapters 6 to 8 introduce algorithms for constructing smart abstractions, which are computed over a reduced state space with non-uniform cells optimized for the system dynamics and control objectives. Chapter 9 presents a modular toolbox designed to facilitate the benchmarking and implementation of these smart abstraction techniques.

PART II
Theory

4

Simulation relations

ABSTRACTION-based control approaches involve synthesizing a correct-by-construction controller through a systematic three-step procedure illustrated in Figure 1. The correctness of this three-step approach is ensured by relating the concrete system with its abstraction in terms of a system relation. The notion of *alternating simulation relation* (Definition 3.1) is crucial there; indeed, it is proved in the seminal work of [AHKV98] that ASR is a sufficient condition for a system $\mathcal{S}_2$ to correctly represent a system $\mathcal{S}_1$, such that any controller $\mathcal{C}_2$ for $\mathcal{S}_2$ can be concretized into a valid controller $\mathcal{C}_1$ for $\mathcal{S}_1$ (Theorem 3.4).

Although the alternating simulation relation offers a safety critical framework, it has several practical drawbacks. The first issue is that this relation provides no complexity guarantee on the concretization procedure, i.e., the concrete controller could contain the entire abstraction (which is typically made up of millions of states and transitions [Rei11, ZPMT11]) as a building block, we refer to this problem as the *concretization complexity issue* (see Section 4.6.1). The second limitation is that, while an alternating simulation relation ensures a controller exists for the concrete system to match the closed-loop behavior of the abstract system, it does not ensure that properties of the abstract controller, such as being static, will be transferred to the concrete controller in the concretization step (see Section 4.6.2).

For this reason, various relations have been introduced in the litera-

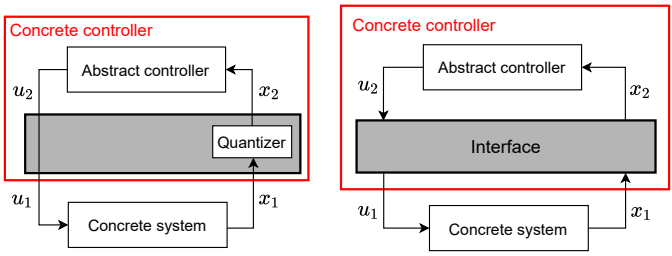


Fig. 4.1 Closed-loop system resulting from the abstraction and concretization approach. Left: Based on the feedback refinement relation and its simple concretization scheme (3.10). Right: Based on the interface proposed in this chapter.

ture, e.g. [RWR16, BPDB18, ELJ22, MOS20], to tackle specific shortcomings of the alternating simulation relation. Some works propose abstraction-based techniques that do not suffer from this concretization complexity issue, but they are either limited to subsets of specification such as safety or reachability [Gir12, DCDVL13, GJ07, HMMS18a], or for a specific class of dynamical systems, e.g. incrementally stable [Gir12], piecewise affine (PWA) dynamics [YTC+11]. Other abstraction-based controller synthesis methods circumvent the concretization complexity issue by constructing abstractions without overlapping cells [Gir13, YTC+11, GJ07, HMMS19, LBJ21, CLEJ21].

An important landmark in the abstraction-based control literature is [RWR16]. There, the authors introduce the stronger *feedback refinement relation* (Definition 3.5), designed to alleviate these limitations. They show that if the two systems enjoy these relations, one may simply concretize the controller by seemlessly plugging the abstract controller (Theorem 3.8), see Figure 4.1 (left).

Recently in [ZAZ24], the authors introduce the *general ϵ-approximate alternating simulation relation* (ϵ-gAAS), which generalizes the alternating simulation relation [ZAZ24, Theorem 3.2]. They show [ZAZ24, Theorem 3.5] that the existence of an ϵ-gAAS is both sufficient and necessary to guarantee the concretization step in (3.5) for some, but not all, abstract controllers. Therefore, it differs from the ASR as it does not permit the design of plug-and-play control architectures between the original system and the abstract controller, i.e., architectures that are independent of the specifications to enforce.

In this chapter and more generally in Part II, we take a step back from these various definitions of relations and ask the following question

> *How can we tailor the abstraction relation based on the desired properties of the concretization step, irrespective of the particular specification we seek to enforce on the original system?*

We start by showing (Example 4.1) that the FRR comes with drawbacks; in particular, we exhibit a system for which there is no FRR allowing to design a valid controller for the abstract system, even though the concrete system is controllable (e.g., with an ASR-type abstraction). Next, we introduce several system relations that we believe are useful in practice for scalable abstraction-based control. We also provide a systematic enumeration of relations that refine the alternating simulation relation, encompassing the different types of simulation relations present in the literature. Then, we characterize these relations based on their concretization procedures, focusing on the resulting control architectures for the concrete system.

Our construction builds on the concept of *interface*, which gathers (and memorizes) the output of the abstract controller $\mathcal{C}_2$, combines it with the observed states from $\mathcal{S}_1$, and outputs a valid control signal, as illustrated in Figure 4.1 (right). The resulting concretization procedure can be viewed as a *hierarchical control architecture*: rather than designing a controller directly for the complex concrete system, one designs an abstract controller (a.k.a. *high-level* controller) over a simpler abstract system, which is then translated into a concrete controller through the interface function (acting as the *low-level controller*). Moreover, this concretization procedure exhibits a *plug-and-play* structure, allowing the abstract controller to be connected to the original system via an interface, irrespective of the particular specification we seek to enforce on the original system.

Our main theoretical contribution is a theorem (Theorem 4.11 and Corollary 4.19) that explicitly links any given simulation relation to the control architecture that the concrete system inherits from the abstract one, as a function of the simulation relation that link them. Thus, the simulation relation becomes one of the elements in the control engineer's toolbox, with the choice of the precise relation to be used depending on the control architecture that (s)he wants to or is able to implement in practice. Finally, we provide explicit constructions of these simulation relations on a common example, highlighting their respective advantages and drawbacks in terms of concretization procedures.

This chapter results from a collaboration with Antoine Girard, and this work has been partially published in [CGJ24].

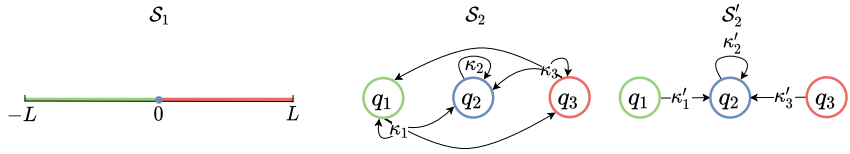
4.1 Motivation

As previously mentioned, the feedback refinement relation (Definition 3.5) benefits from a concretization procedure that is irrespective of the specification we aim to enforce on the original system. This is achieved by interconnecting the abstract controller $\mathcal{C}_2$ with the quantizer R (Theorem 3.8), as illustrated in Figure 4.1 (left). However, this benefit comes with a cost: it constrains the way the abstraction is created, as it limits the concretized controller to be piecewise constant. Specifically, the requirements in (i) and (ii) of Definition 3.5 restrict the class of concrete controllers to those where the control input depends only on the abstract state (3.10). Thus, the feedback refinement relation is well-suited to contexts where the exact state is unknown and only quantified (or symbolic) state information is available. However, when information on the concrete state is available, this becomes a significant limitation. If the system does not exhibit local incremental stability (Definition 2.22) around a cell, the use of piecewise constant controllers introduces significant non-determinism into the abstraction. This could result in an intractable or even unsolvable abstract problem $(\mathcal{S}_2, \Sigma_2)$, as illustrated by the following example from [CMGJ24, Section 4.4].

Example 4.1. We consider the simple systems

$$\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1),\ \mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2) \text{ and } \mathcal{S}_2' = (\mathcal{X}_2, \mathcal{U}_2', F_2')$$

with $\mathcal{X}_1 = [-L, L] \subseteq \mathbb{R}$, $\mathcal{U}_1 = \mathbb{R}$, $\mathcal{X}_2 = \{q_1, q_2, q_3\}$, $\mathcal{U}_2 = \{\kappa_1, \kappa_2, \kappa_3\}$, $\mathcal{U}_2' = \{\kappa_1', \kappa_2', \kappa_3'\}$, $F_1(x_1, u) = \{x_1 + u\}$ and the available inputs $\mathcal{U}_{\mathcal{S}_1}(x_1) = \{u \in \mathcal{U}_1 \mid x_1 + u \in \mathcal{X}_1\}$. The transition maps F_2 and F_2' are illustrated graphically by



Given the sets $\mathcal{X}_{q_1} = [-L, 0)$, $\mathcal{X}_{q_2} = \{0\}$ and $\mathcal{X}_{q_3} = (0, L]$, we define the relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ such that $(x_1, q) \in R \Leftrightarrow x_1 \in \mathcal{X}_q$. The colors indicate the related states. We consider the reach-avoid problem $\Sigma_1^{\text{Reach}} = [\mathcal{X}_1, \{0\}, \varnothing]$ of controlling $\mathcal{S}_1$, whose dynamics consist of moving under translation on a segment of the real line, to reach 0. The abstract specification $\Sigma_2^{\text{Reach}} = [\mathcal{X}_2, \{q_2\}, \varnothing]$, consisting of driving the system from state q_1, q_2, q_3 to q_2 satisfies (3.3). Any abstraction $\mathcal{S}_2$ such that $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$ is constrained to have non-determinism for the cells q_1 and q_3. The problem here is that by using piecewise constant controllers for the concrete system, there will always be a portion around 0 that overshoot its target. Indeed, we have $\kappa_1 \in [0, L]$, $\kappa_2 = 0$ and $\kappa_3 \in [-L, 0]$, and therefore, for any admissible choice of $\kappa_1, \kappa_2, \kappa_3$, the resulting abstraction $\mathcal{S}_2$ is non-deterministic. As a result, the abstract problem $(\mathcal{S}_2, \Sigma_2^{\text{Reach}})$ has no solution. However, we can build an abstraction $\mathcal{S}_2'$ such that $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2'$ where $\kappa_1'(x_1) = \kappa_3'(x_1) = -x_1$ and $\kappa_2'(x_1) = 0$ are linear controllers. Note that we have the same partition of state-space, but because we can have abstract inputs that are local state-dependent controllers instead of real inputs, we can solve the abstract problem $(\mathcal{S}_2', \Sigma_2^{\text{Reach}})$. For the abstract system $\mathcal{S}_2'$, the inputs κ_1', κ_2' and κ_3' can be interpreted as *move to the right cell*, *do not move* and *move to the left cell*, respectively. △

Therefore, our *goal* is to identify, classify, and characterize refined notions of the alternating simulation relation, denoted as T, that allow the design of more general concrete controllers than the piecewise constant controllers of the feedback refinement relation, while still benefiting from its plug-and-play property.

4.2 System relations

In this section, we enumerate and classify system relations that refine the alternating simulation relation (Definition 3.1).

4.2.1 Key relations

We introduce refined notions of alternating simulation relation, which will be central to the discussions throughout this thesis.

Definition 4.2 (Predictive simulation relation)**.** Given two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ is a *predictive simulation relation* from $\mathcal{S}_1$ to $\mathcal{S}_2$, denoted $\mathcal{S}_1 \preceq_R^{\text{PSR}} \mathcal{S}_2$, if

$$\begin{aligned}\forall (x_1, x_2) \in R\ \forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)\ \exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1) \\ \exists x_2^+ \in F_2(x_2, u_2)\ \forall x_1^+ \in F_1(x_1, u_1) :\ (x_1^+, x_2^+) \in R. \end{aligned} \tag{4.1}$$

△

Definition 4.3 (Delayed simulation relation)**.** Given two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ is a *delayed simulation relation* from $\mathcal{S}_1$ to $\mathcal{S}_2$, denoted $\mathcal{S}_1 \preceq_R^{\text{DSR}} \mathcal{S}_2$, if

$$\begin{aligned}\forall x_2 \in \mathcal{X}_2\ \forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)\ \exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(R^{-1}(x_2))\ \forall x_1 \in R^{-1}(x_2) \\ \exists x_2^+ \in F_2(x_2, u_2)\ \forall x_1^+ \in F_1(x_1, u_1) :\ (x_1^+, x_2^+) \in R. \end{aligned} \tag{4.2}$$

△

Definition 4.4 (Feedforward abstraction relation)**.** Given two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ is a *feedforward abstraction relation* from $\mathcal{S}_1$ to $\mathcal{S}_2$, denoted $\mathcal{S}_1 \preceq_R^{\text{FAR}} \mathcal{S}_2$, if

$$\begin{aligned}\forall x_2 \in \mathcal{X}_2\ \forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)\ \exists x_2^+ \in F_2(x_2, u_2)\ \forall x_1 \in R^{-1}(x_2) \\ \exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)\ \forall x_1^+ \in F_1(x_1, u_1) :\ (x_1^+, x_2^+) \in R. \end{aligned} \tag{4.3}$$

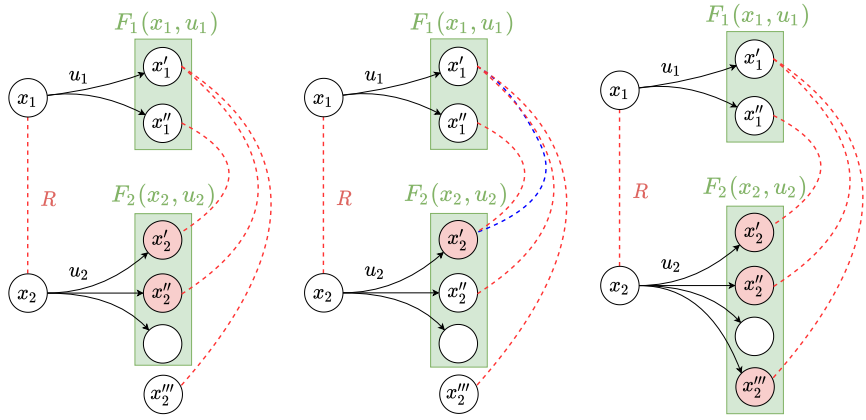
△

Definition 4.5 (Memoryless concretization relation)**.** Given two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ is a *memoryless concretization relation* from $\mathcal{S}_1$ to $\mathcal{S}_2$, denoted $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2$, if

$$\begin{aligned}\forall (x_1, x_2) \in R\ \forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) \exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1) \\ \forall x_1^+ \in F_1(x_1, u_1) : R(x_1^+) \neq \varnothing \text{ and } R(x_1^+) \subseteq F_2(x_2, u_2). \end{aligned} \tag{4.4}$$

△

We define the associated *extended relations* R_e^{ASR}, R_e^{PSR}, $R_e^{\text{MCR}} \subseteq \mathcal{X}_2 \times \mathcal{U}_2 \times \mathcal{X}_1 \times \mathcal{U}_1$ as the sets of tuples (x_2, u_2, x_1, u_1) satisfying (3.8), (4.1), and (4.4) respectively. Similarly, $R_e^{\text{FRR}} \subseteq \mathcal{X}_2 \times \mathcal{U}_2 \times \mathcal{X}_1$ and $R_e^{\text{FAR}} \subseteq \mathcal{X}_2 \times \mathcal{U}_2 \times \mathcal{X}_1 \times \mathcal{U}_1 \times \mathcal{X}_2$ are defined as the set of tuples (x_2, u_2, x_1) and $(x_2, u_2, x_1, u_1, x_2^+)$ satisfying (3.9) and (4.3), respectively.



(a) ASR. **(b)** PSR. **(c)** MCR.

Fig. 4.2 Local conditions for ASR, PSR, and MCR. Related states are connected with a dotted line. The relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ is consistent across all three examples, with the exception of the additional related pair $(x_1', x_2') \in R$ in PSR, illustrated in blue. This relation defines a cover of the state space, as $|R(x_1')| > 1$. The distinction among the examples lies in the definition of the abstract transition map $F_2(x_2, u_2)$. States that must be included in $F_2(x_2, u_2)$ are shown in red, though other states may also be part of it. Note that for ASR, either x_2'' or x_2''' must be included in $F_2(x_2, u_2)$.

Additionally, we define the set-valued maps

$$I_R^{\mathrm{T}}(x_2, u_2, x_1) = \{u_1 \mid (x_2, u_2, x_1, u_1) \in R_e^{\mathrm{T}}\}, \tag{4.5}$$

for $T \in \{\mathrm{ASR}, \mathrm{PSR}, \mathrm{MCR}\}$. Similarly, $I_R^{\mathrm{FRR}} : \mathcal{U}_2 \to \mathcal{U}_2$ and $I_R^{\mathrm{FAR}} : \mathcal{X}_2 \times \mathcal{U}_2 \times \mathcal{X}_1 \times \mathcal{X}_2 \to 2^{\mathcal{U}_1}$ are defined as $I_R^{\mathrm{FRR}}(u_2) = \{u_2\}$ and $I_R^{\mathrm{FAR}}(x_2, u_2, x_1, x_2^+) = \{u_1 \mid (x_2, u_2, x_1, u_1, x_2^+) \in R_e^{\mathrm{FAR}}\}$. These functions map abstract inputs to concrete ones.

An example illustrating the local conditions for ASR, PSR, and MCR is shown in Figure 4.2, highlighting their differences.

For both PSR and FAR, the next abstract state x_2^+ can be identified before applying the concrete input u_1 to the original system. The distinction between PSR and FAR lies in the dependence on x_1 for determining x_2^+ in PSR, whereas in FAR, x_2^+ can be determined solely from abstract information without reference to the specific concrete state $x_1 \in R^{-1}(x_2)$. This is illustrated with the following example.

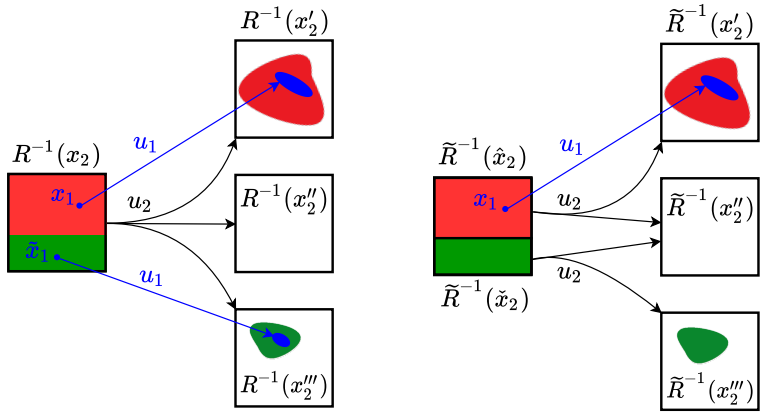


Fig. 4.3 Comparison of PSR and FAR for three simple systems. Let $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$, $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$, and $\widetilde{\mathcal{S}}_2 = (\widetilde{\mathcal{X}}_2, \mathcal{U}_2, \widetilde{F}_2)$, where $\mathcal{X}_1 \subseteq \mathbb{R}^2$, and $\mathcal{X}_2$ and $\widetilde{\mathcal{X}}_2$ are finite sets. Left: Illustration of $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ such that $\mathcal{S}_1 \preceq_R^{\text{PSR}} \mathcal{S}_2$ ($\mathcal{S}_1 \not\preceq_R^{\text{FAR}} \mathcal{S}_2$), highlighting abstract states $x_2, x_2', x_2'', x_2''' \in \mathcal{X}_2$. Right: Illustration of $\widetilde{R} \subseteq \mathcal{X}_1 \times \widetilde{\mathcal{X}}_2$ such that $\mathcal{S}_1 \preceq_{\widetilde{R}}^{\text{FAR}} \widetilde{\mathcal{S}}_2$, with abstract states $\hat{x}_2, \check{x}_2, x_2', x_2'', x_2''' \in \widetilde{\mathcal{X}}_2$. Here, $\widetilde{R}$ refines R by dividing cell x_2 into $\hat{x}_2$ and $\check{x}_2$. In red, we show $\widetilde{R}^{-1}(\hat{x}_2)$ and $F_1(\widetilde{R}^{-1}(\hat{x}_2), u_1)$; in green, $\widetilde{R}^{-1}(\check{x}_2)$ and $F_1(\widetilde{R}^{-1}(\check{x}_2), u_1)$; and in blue, x_1 (resp. $\widetilde{x}_1$) and $F_1(x_1, u_1)$ (resp. $F_1(\widetilde{x}_1, u_1)$).

Example 4.6. Consider the systems shown in Figure 4.3. Given $(x_1, x_2) \in R$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, there exists $u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)$ such that $F_1(x_1, u_1) \subseteq R^{-1}(x_2')$ for some $x_2' \in F_2(x_2, u_2)$. However, as illustrated, the next abstract state satisfying (4.1) depends on the specific concrete state in $R^{-1}(x_2)$; for instance, with $(\widetilde{x}_1, x_2) \in R$, this is x_2''' instead of x_2'. In contrast, for FAR, given $\hat{x}_2$ and $u_2 \in \mathcal{U}_{\widetilde{\mathcal{S}}_2}(\hat{x}_2)$, there exists a common abstract state $x_2' \in \widetilde{F}_2(\hat{x}_2, u_2)$ such that for all $x_1 \in \widetilde{R}^{-1}(\hat{x}_2)$, we have $F_1(x_1, u_1) \subseteq \widetilde{R}^{-1}(x_2')$. △

The following proposition establishes the hierarchy of system relations.

Proposition 4.7. *Given the simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$, the following implications hold*

(1) $\mathcal{S}_1 \preceq_R^{\text{FAR}} \mathcal{S}_2 \Rightarrow \mathcal{S}_1 \preceq_R^{\text{PSR}} \mathcal{S}_2 \Rightarrow \mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$.

(2) $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2 \Rightarrow \mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2 \Rightarrow \mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$.

(3) $\mathcal{S}_1 \preceq_R^{\text{DSR}} \mathcal{S}_2 \Rightarrow \mathcal{S}_1 \preceq_R^{\text{PSR}} \mathcal{S}_2$.

Additionally,

(4) If $\mathcal{S}_2$ is deterministic, then $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2 \Leftrightarrow \mathcal{S}_1 \preceq_R^{\text{FAR}} \mathcal{S}_2$.

(5) If R is deterministic, then $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2 \Leftrightarrow \mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2$.

(6) If $u_1 \in I_R^{\text{MCR}}(x_2, u_2, x_1)$ implies $u_1 = u_2$, then $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2 \Leftrightarrow \mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$.

Proof.

(1) (FAR $\Rightarrow$ PSR) From condition (4.3), we have that for any $x_2 \in \mathcal{X}_2$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, there exists $x_2^+ \in F_2(x_2, u_2)$ such that $\forall x_1 \in R^{-1}(x_2)$, there exists $u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)$ for which $\forall x_1^+ \in F_1(x_1, u_1) : (x_1^+, x_2^+) \in R$. Given $(x_1, x_2) \in R$, we have $x_1 \in R^{-1}(x_2)$. Thus for this x_1, there exists $u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)$ such that $\forall x_1^+ \in F_1(x_1, u_1) : (x_1^+, x_2^+) \in R$. This establishes condition (4.1).

(PSR $\Rightarrow$ ASR) In (3.8), the condition $R(x_1^+) \cap F_2(x_2, u_2) \neq \varnothing$ is equivalent to $\exists x_2^+ \in F_2(x_2, u_2) : (x_1^+, x_2^+) \in R$. Then, the result follows directly from the permutation of the quantifiers $\exists x_2^+ \in F_2(x_2, u_2)$ and $\forall x_1^+ \in F_1(x_1, u_1)$ in (4.1) and (3.8).

(2) (FRR $\Rightarrow$ MCR) Eq. (4.4) is satisfied with $u_1 = u_2$.

(MCR $\Rightarrow$ ASR) The condition (4.4) of MCR, where $R(x_1^+) \neq \varnothing$ and $R(x_1^+) \subseteq F_2(x_2, u_2)$, implies $R(x_1^+) \cap F_2(x_2, u_2) \neq \varnothing$ which corresponds to (3.8).

(3) Directly follows from inverting $\forall x_1 \in R^{-1}(x_2)$ and $\exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)$.

(4) Given that $\mathcal{S}_2$ is deterministic, it follows that for any $x_2 \in \mathcal{X}_2$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, we have $|F_2(x_2, u_2)| = 1$. Consequently, condition (3.8) is equivalent to: $\forall (x_1, x_2) \in R \; \forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) \; x_2^+ \in F_2(x_2, u_2) \; \exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1) \; \forall x_1^+ \in F_1(x_1, u_1) : (x_1^+, x_2^+) \in R$, which in turn is equivalent to (4.3).

(5) The relation R is deterministic implies that for any $(x_1, x_2) \in R$: $|R(x_1)| = 1$. Therefore $R(x_1^+) \cap F_2(x_2, u_2) \neq \varnothing$ if and only if $R(x_1^+) \neq \varnothing$, $R(x_1^+) \subseteq F_2(x_2, u_2)$.

(6) Condition (3.9) is equivalent to (4.4), with the additional requirement that the abstract and concrete inputs in the extended relation are identical.

□

4.2.2 Comprehensive enumeration

In the previous section, we introduced refined notions of the alternating simulation relation. Here, we propose a systematic method to derive relation types T, denoted by $\mathcal{S}_1 \preceq_R^{\mathrm{T}} \mathcal{S}_2$, which satisfy

$$\mathcal{S}_1 \preceq_R^{\mathrm{T}} \mathcal{S}_2 \Rightarrow \mathcal{S}_1 \preceq_R^{\mathrm{ASR}} \mathcal{S}_2. \tag{4.6}$$

The condition for ASR (3.8) can be reformulated as:

$$\boxed{\forall x_2 \in \mathcal{X}_2}\ \boxed{\forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)}\ \boxed{\forall x_1 \in R^{-1}(x_2)}\ \boxed{\exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)}$$
$$\boxed{\forall x_1^+ \in F_1(x_1, u_1)}\ \boxed{\exists x_2^+ \in F_2(x_2, u_2) \cap R(x_1^+)} \tag{4.7}$$

We decompose the logical formula (4.7) into blocks corresponding to the variables x_2, u_2, x_1, u_1, x_1^+, and x_2^+. The notation $x_1^+ < x_2^+$ indicates that the block related to x_1^+ precedes x_2^+. For example, the ordering in the ASR condition (4.7) is $x_2 < u_2 < x_1 < u_1 < x_1^+ < x_2^+$.

To derive these refined relations, we systematically apply logical transformations to the ASR condition (4.7), producing stronger relations.

In the three-step abstraction-based approach, once the abstraction is constructed in step 1 (Figure 1), any abstract controller $\mathcal{C}_2$ designed in step 2 can be concretized into a concrete controller $\mathcal{C}_1$ in step 3, ensuring (3.5). Therefore, the logical formula must start with $\forall x_2 \in \mathcal{X}_2 \forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$.

We apply the following logical operations[1].

1. Replace the block $\exists x_2^+ \in F_2(x_2, u_2) \cap R(x_1^+)$ by one of the following propositions

$$\boxed{\exists x_2^+ \in F_2(x_2, u_2)} : x_2^+ \in R(x_1^+) \tag{4.8}$$
$$\boxed{\forall x_2^+ \in F_2(x_2, u_2)} : x_2^+ \in R(x_1^+) \tag{4.9}$$
$$\boxed{\forall x_2^+ \in R(x_1^+)} : x_2^+ \in F_2(x_2, u_2). \tag{4.10}$$

   Note that (4.8) is equivalent to $\exists x_2^+ \in F_2(x_2, u_2) \cap R(x_1^+)$. Additionally, we have (4.9) $\Rightarrow$ (4.8) and (4.10) $\Rightarrow$ (4.8).

[1] Blue boxes indicate interchangeable elements, while blue content highlights modifiable components.

2. Permute the blue blocks, ensuring $x_1 < x_1^+$, $u_1 < x_1^+$, $x_2 < x_2^+$, and $u_2 < x_2^+$.
3. Impose $u_1 = u_2$ with the condition that if $(x_1, x_2) \in R$, then $\mathcal{U}_{\mathcal{S}_2}(x_2) \subseteq \mathcal{U}_{\mathcal{S}_1}(x_1)$. We do not replace the existential quantifier of u_1 with a universal quantifier, as this would lead to irrelevant relation definitions, allowing any concrete input to preserve the relation independently of abstract variables.

In order to systematically enumerate all valid relations, we use a constructive tree (Figure 4.4). Each blue edge label represents a logical choice, with each leaf node corresponding to a refined system relation of ASR, as shown in Table 4.1. The logical formula for each leaf node is constructed by following the instructions along the path from the root to the leaf. This tree structure not only enables exhaustive enumeration but also meaningful grouping of relation types sharing common properties, as illustrated in Figure 4.5.

There are 17 relations in total, including previously introduced ones like ASR and FRR. References for previously established relations are provided in Table 4.1, while the remaining relations are, to the best of our knowledge, newly introduced in this thesis.

The following theorem, a generalization of Proposition 4.7, validates the enumeration in Figure 4.4 and the hierarchy of system relations in Figure 4.5.

Theorem 4.8. *Figure 4.5 provides the hierarchy of system relations, showing the ordering between the various types of system relations listed in Table 4.1.*

Proof. The implications $7 \Rightarrow 4 \Rightarrow 1$ and $17 \Rightarrow 15 \Rightarrow 1$, along with $1 \Rightarrow 4$ under assumption A_1 and $1 \Rightarrow 15$ under A_2, are established in Proposition 4.7. Similar reasoning applies to the chains $8 \Rightarrow 5 \Rightarrow 2$ and $16 \Rightarrow 6$, with $2 \Rightarrow 8$ under A_1 and $2 \Rightarrow 16$ under A_2. Likewise, $9 \Rightarrow 6 \Rightarrow 3$ and $17 \Rightarrow 3$ hold, with $3 \Rightarrow 9$ under A_1 and $3 \Rightarrow 17$ under A_2. In 2, reversing the order $\forall x_1 \in R^{-1}(x_2) \exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)$ from 1 gives $2 \Rightarrow 1$. In 3, imposing $u_1 = u_2$ strengthens the condition, leading to $3 \Rightarrow 2$. The other implications, $9 \Rightarrow 8 \Rightarrow 7$, $6 \Rightarrow 5 \Rightarrow 4$, $17 \Rightarrow 16 \Rightarrow 15$, $14 \Rightarrow 13 \Rightarrow 12$, and $11 \Rightarrow 10$, follow similarly. The implications $10 \Rightarrow 7$, $11 \Rightarrow 8$, and $14 \Rightarrow 9$ follow by replacing $\forall x_2^+$ with $\exists x_2^+$. Finally, the implications $12 \Rightarrow 10$ and $13 \Rightarrow 11$ follow from switching the positions of $\exists u_1$ and $\forall x_2^+$. □

Most of abstraction-based algorithms found in the literature rely either on MCR for [RZ16, CBLJ24a, CLEJ21] or FAR for [MGG13, ELJ22], because

they typically involve constructing either deterministic quantizer (partition of the state space) or a deterministic abstraction (Theorem 4.8), respectively.

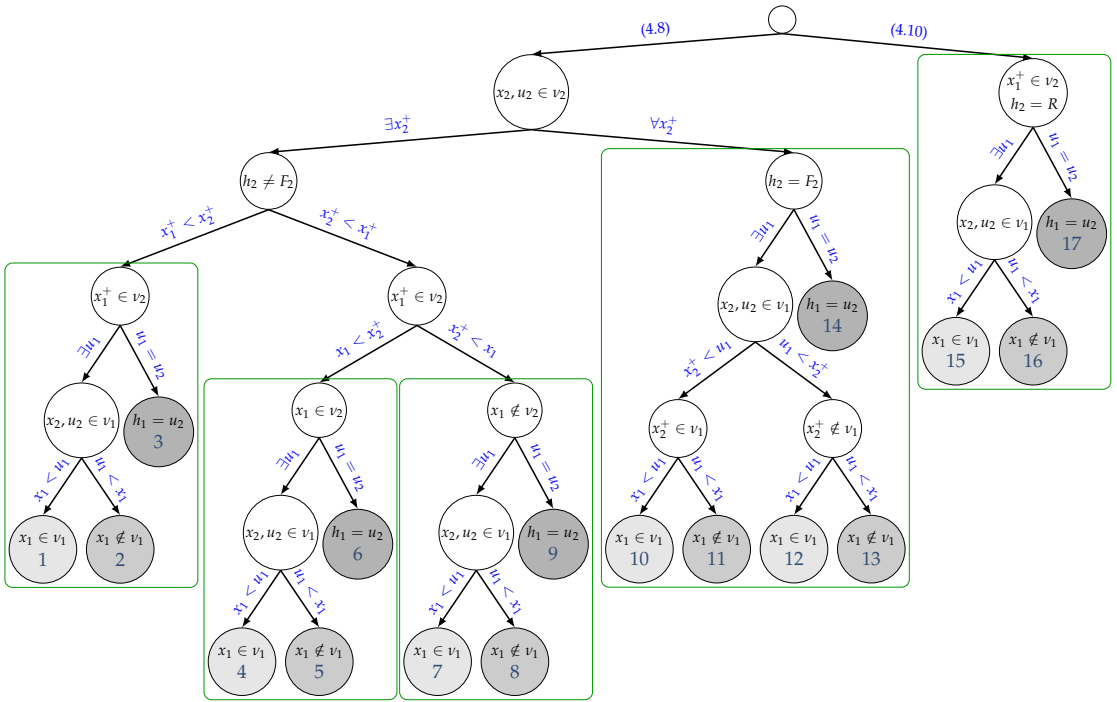


Fig. 4.4 Constructive tree of relation types T that refine ASR (4.6). Each *leaf* corresponds to a relation type listed in Table 4.1. The blue labels on the *edges* of the path from the root to a leaf represent the logical operations applied to (4.7) to construct the *relation* (Section 4.2.2). The *nodes* along the path from the root to a leaf specify the variables ν_1 and ν_2 for the set-valued functions h_1 and h_2, respectively, which define the *interface* of the corresponding relation (Section 4.3).

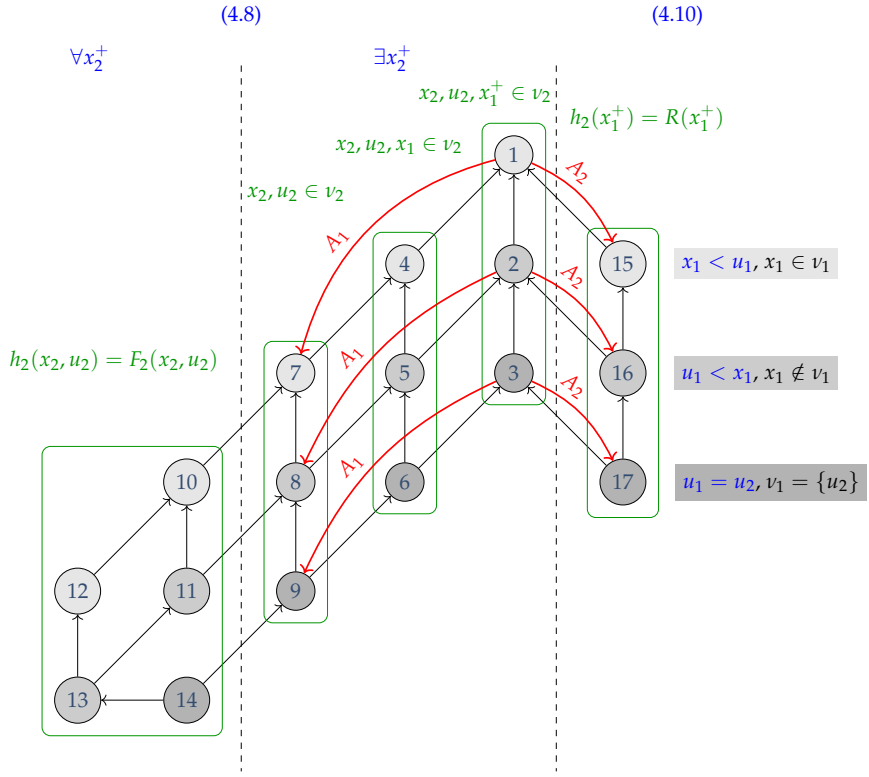


Fig. 4.5 Hierarchy of system relations. For example, an arrow from node 4 to node 1 indicates that $\mathcal{S}_1 \preceq_R^{\text{PSR}} \mathcal{S}_2$ implies $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$. Red arrows represent implications that hold under specific assumptions labeled on the arrows: A_1 denotes that $\mathcal{S}_2$ is deterministic, and A_2 indicates that R is deterministic. Light, medium, and dark grey nodes represent relations where $x_1 < u_1$, $u_1 < x_1$, and $u_1 = u_2$, respectively. Green boxes correspond to the leaves of the subtrees highlighted in green in Figure 4.4.

N°	Definition	Interface	Name
1	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \forall x_1 \in R^{-1}(x_2)\ \exists u_1 \in \mathcal{U}_{S_1}(x_1)$ $\forall x_1^+ \in F_1(x_1,u_1)\ \exists x_2^+ \in F_2(x_2,u_2) :\ x_2^+ \in R(x_1^+)$	$h_1(x_2,u_2,x_1)$ $h_2(x_2,u_2,x_1^+)$	ASR (Definition 3.1) [Tab09, Definition 4.19]
2	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \exists u_1 \in \mathcal{U}_{S_1}(R^{-1}(x_2))\ \forall x_1 \in R^{-1}(x_2)$ $\forall x_1^+ \in F_1(x_1,u_1)\ \exists x_2^+ \in F_2(x_2,u_2) :\ x_2^+ \in R(x_1^+)$	$h_1(x_2,u_2)$ $h_2(x_2,u_2,x_1^+)$	
3	$\bullet\ \mathcal{U}_{S_2}(x_2) \subseteq \mathcal{U}_{S_1}(x_1)$ $\bullet\ \forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \forall x_1 \in R^{-1}(x_2)\ \forall x_1^+ \in F_1(x_1,u_2)$ $\exists x_2^+ \in F_2(x_2,u_2) :\ x_2^+ \in R(x_1^+)$	$h_1(u_2) = \{u_2\}$ $h_2(x_2,u_2,x_1^+)$	S-ASR [BPDB18, Definition 6]
4	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \forall x_1 \in R^{-1}(x_2)\ \exists u_1 \in \mathcal{U}_{S_1}(x_1)$ $\exists x_2^+ \in F_2(x_2,u_2)\ \forall x_1^+ \in F_1(x_1,u_1) :\ x_2^+ \in R(x_1^+)$	**a**: $h_1(x_2,u_2,x_1)$ $h_2(x_2,u_2,x_1,u_1)$ **b**: $h_1(x_2,u_2,x_1,x_2^+)$ $h_2(x_2,u_2,x_1)$ **c**: $h_1(x_2,u_2,x_1)$ $h_2(x_2,u_2,x_1)$	PSR (Definition 4.2) [CGJ24, Definition 9]
5	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \exists u_1 \in \mathcal{U}_{S_1}(R^{-1}(x_2))\ \forall x_1 \in R^{-1}(x_2)$ $\exists x_2^+ \in F_2(x_2,u_2)\ \forall x_1^+ \in F_1(x_1,u_1) :\ x_2^+ \in R(x_1^+)$	**a**: $h_1(x_2,u_2)$ $h_2(x_2,u_2,x_1,u_1)$ **b**: $h_1(x_2,u_2)$ $h_2(x_2,u_2,x_1)$	DSR (Definition 4.3)
6	$\bullet\ \mathcal{U}_{S_2}(x_2) \subseteq \mathcal{U}_{S_1}(x_1)$ $\bullet\ \forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \forall x_1 \in R^{-1}(x_2)\ \exists x_2^+ \in F_2(x_2,u_2)$ $\forall x_1^+ \in F_1(x_1,u_2) :\ x_2^+ \in R(x_1^+)$	$h_1(u_2) = \{u_2\}$ $h_2(x_2,u_2,x_1)$	
7	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \exists x_2^+ \in F_2(x_2,u_2)\ \forall x_1 \in R^{-1}(x_2)$ $\exists u_1 \in \mathcal{U}_{S_1}(x_1)\ \forall x_1^+ \in F_1(x_1,u_1) :\ x_2^+ \in R(x_1^+)$	**a**: $h_1(x_2,u_2,x_1,x_2^+)$ $h_2(x_2,u_2)$ **b**: $h_1(x_2,u_2,x_1)$ $h_2(x_2,u_2)$	FAR (Definition 4.4) [CGJ24, Definition 10]
8	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \exists x_2^+ \in F_2(x_2,u_2)\ \exists u_1 \in \mathcal{U}_{S_1}(R^{-1}(x_2))$ $\forall x_1 \in R^{-1}(x_2)\ \forall x_1^+ \in F_1(x_1,u_1) :\ x_2^+ \in R(x_1^+)$	**a**: $h_1(x_2,u_2,x_2^+)$ $h_2(x_2,u_2)$ **b**: $h_1(x_2,u_2)$ $h_2(x_2,u_2,u_1)$ **c**: $h_1(x_2,u_2)$ $h_2(x_2,u_2)$	
9	$\bullet\ \mathcal{U}_{S_2}(x_2) \subseteq \mathcal{U}_{S_1}(x_1)$ $\bullet\ \forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \exists x_2^+ \in F_2(x_2,u_2)\ \forall x_1 \in R^{-1}(x_2)$ $\forall x_1^+ \in F_1(x_1,u_2) :\ x_2^+ \in R(x_1^+)$	$h_1(u_2) = \{u_2\}$ $h_2(x_2,u_2)$	
10	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \forall x_2^+ \in F_2(x_2,u_2)\ \forall x_1 \in R^{-1}(x_2)$ $\exists u_1 \in \mathcal{U}_{S_1}(x_1)\ \forall x_1^+ \in F_1(x_1,u_1)\ :\ x_2^+ \in R(x_1^+)$	$h_1(x_2,u_2,x_1,x_2^+)$ $h_2(x_2,u_2) = F_2(x_2,u_2)$	SFR [ELJ22, Definition 1]
11	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \forall x_2^+ \in F_2(x_2,u_2)\ \exists u_1 \in \mathcal{U}_{S_1}(R^{-1}(x_2))$ $\forall x_1 \in R^{-1}(x_2)\ \forall x_1^+ \in F_1(x_1,u_1)\ :\ x_2^+ \in R(x_1^+)$	$h_1(x_2,u_2,x_2^+)$ $h_2(x_2,u_2) = F_2(x_2,u_2)$	
12	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \forall x_1 \in R^{-1}(x_2)\ \exists u_1 \in \mathcal{U}_{S_1}(x_1)$ $\forall x_2^+ \in F_2(x_2,u_2)\ \forall x_1^+ \in F_1(x_1,u_1)\ :\ x_2^+ \in R(x_1^+)$	$h_1(x_2,u_2,x_1)$ $h_2(x_2,u_2) = F_2(x_2,u_2)$	
13	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \exists u_1 \in \mathcal{U}_{S_1}(R^{-1}(x_2))\ \forall x_2^+ \in F_2(x_2,u_2)$ $\forall x_1 \in R^{-1}(x_2)\ \forall x_1^+ \in F_1(x_1,u_1)\ :\ x_2^+ \in R(x_1^+)$	$h_1(x_2,u_2)$ $h_2(x_2,u_2) = F_2(x_2,u_2)$	
14	$\bullet\ \mathcal{U}_{S_2}(x_2) \subseteq \mathcal{U}_{S_1}(x_1)$ $\bullet\ \forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \forall x_2^+ \in F_2(x_2,u_2)\ \forall x_1 \in R^{-1}(x_2)$ $\forall x_1^+ \in F_1(x_1,u_2) :\ x_2^+ \in R(x_1^+)$	$h_1(u_2) = \{u_2\}$ $h_2(x_2,u_2) = F_2(x_2,u_2)$	
15	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \forall x_1 \in R^{-1}(x_2)\ \exists u_1 \in \mathcal{U}_{S_1}(x_1)$ $\forall x_1^+ \in F_1(x_1,u_1)\ \forall x_2^+ \in R(x_1^+)\ :\ x_2^+ \in F_2(x_2,u_2)$	$h_1(x_2,u_2,x_1)$ $h_2(x_1^+) = R(x_1^+)$	MCR (Definition 4.5) [CMGJ24, Definition 8]
16	$\forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \exists u_1 \in \mathcal{U}_{S_1}(R^{-1}(x_2))\ \forall x_1 \in R^{-1}(x_2)$ $\forall x_1^+ \in F_1(x_1,u_1)\ \forall x_2^+ \in R(x_1^+)\ :\ x_2^+ \in F_2(x_2,u_2)$	$h_1(x_2,u_2)$ $h_2(x_1^+) = R(x_1^+)$	
17	$\bullet\ \mathcal{U}_{S_2}(x_2) \subseteq \mathcal{U}_{S_1}(x_1)$ $\bullet\ \forall x_2\ \forall u_2 \in \mathcal{U}_{S_2}(x_2)\ \forall x_1 \in R^{-1}(x_2)\ \forall x_1^+ \in F_1(x_1,u_2)$ $\forall x_2^+ \in R(x_1^+) :\ x_2^+ \in F_2(x_2,u_2)$	$h_1(u_2) = \{u_2\}$ $h_2(x_1^+) = R(x_1^+)$	FRR (Definition 3.5) [RWR16, Definition V.2]

Table 4.1 Table of system relations derived from the tree in Figure 4.4. The numbers correspond to the leaf nodes in Figure 4.4, with each relation defined by following the blue labels on the edges of the path from the root to the respective leaf (Section 4.2.2). The "Interface" column lists the corresponding interfaces (Section 4.3), while the "Name" column provides the relation's name and reference if it has been previously introduced in the literature. Architectures labeled in bold represent the deterministic case (refer to the discussion in Section 4.4.2).

4.3 Interface

In this section, we introduce the concept of an *interface* that connects the system $\mathcal{S}_1$ to any controller $\mathcal{C}_2$ f.c. with the system $\mathcal{S}_2$, as shown in Figure 4.1 (right).

Definition 4.9 (Interface)**.** Given two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), an *interface* for $\mathcal{S}_1$ and $\mathcal{S}_2$ is defined by a tuple (h_1, h_2, R), where $h_1 : \mathcal{H}_1 \to 2^{\mathcal{U}_1}$ and $h_2 : \mathcal{H}_2 \to 2^{\mathcal{X}_2}$ are set-valued functions, and $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$. The argument ν_1 (resp. ν_2) of h_1 (resp. h_2) is a subset of the set of variables $\nu = \{x_1, x_1^+, x_2, x_2^+, u_1, u_2\}$, where $x_1, x_1^+ \in \mathcal{X}_1$, $x_2, x_2^+ \in \mathcal{X}_2$, $u_1 \in \mathcal{U}_1$, and $u_2 \in \mathcal{U}_2$[2]. The functions h_1 and h_2 ensure that the following holds. For all $(x_1, x_2) \in R$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, it holds that any u_1, x_1^+ and x_2^+ resulting from the following sequence of equations

$$\begin{aligned} u_1 &\in h_1(\nu_1), \\ x_2^+ &\in h_2(\nu_2), \\ x_1^+ &\in F_1(x_1, u_1), \end{aligned} \tag{4.11}$$

satisfy $x_2^+ \in F_2(x_2, u_2)$ and $(x_1^+, x_2^+) \in R$. △

The order of the equations in (4.11) may vary based on the variables ν_1 and ν_2 of h_1 and h_2, respectively, as exemplified in (4.16), (4.18), and (4.22). The function h_1 determines the concrete input u_1 required to ensure that the resulting concrete state $x_1^+ \in F_1(x_1, u_1)$ is related to an abstract state $x_2^+ \in F_2(x_2, u_2)$, identified by h_2.

To emphasize the invariant preserved by the interface, specifically that $(x_1^+, x_2^+) \in R$ given $(x_1, x_2) \in R$, we rewrite the condition (4.11) in Definition 4.9 as follows

$$\begin{aligned} &\text{Pre: } (x_1, x_2) \in R,\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) \\ &\qquad u_1 \in h_1(\nu_1), \\ &\qquad x_2^+ \in h_2(\nu_2), \\ &\qquad x_1^+ \in F_1(x_1, u_1), \\ &\text{Post: } x_2^+ \in F_2(x_2, u_2),\ (x_1^+, x_2^+) \in R \end{aligned} \tag{4.12}$$

with the *pre-condition* and *post-condition* highlighted in blue.

[2]The domain $\mathcal{H}_1$ (resp. $\mathcal{H}_2$) of h_1 (resp. h_2) is defined as the Cartesian product of the domains of the corresponding variable ν_1 (resp. ν_2).

We now define the controller $\mathcal{C}_1$, referred to as the *concretized controller*, which results from connecting the controller $\mathcal{C}_2$ to the interface.

Definition 4.10 (Concretized controller)**.** Given two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), an interface (h_1, h_2, R) for $\mathcal{S}_1$ and $\mathcal{S}_2$, and a controller $\mathcal{C}_2$ f.c. with $\mathcal{S}_2$ defined as

$$\mathcal{C}_2 = (\mathcal{X}_{\mathcal{C}_2}, \mathcal{X}_2, \mathcal{V}_{\mathcal{C}_2}, \mathcal{U}_2, F_{\mathcal{C}_2}, H_{\mathcal{C}_2}), \tag{4.13}$$

a *concretized controller* is any system $\mathcal{C}_1$ that satisfies the following closed-loop condition: $\boldsymbol{x}_1 \in \mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1)$ if and only if there exist sequences $\boldsymbol{u}_1$, $\boldsymbol{x}_2$, $\boldsymbol{u}_2$, $\boldsymbol{x}_{\mathcal{C}_2}$, and $\boldsymbol{v}_{\mathcal{C}_2}$ such that

$$\begin{aligned}
&(u_2(k), v_{\mathcal{C}_2}(k)) \in H_{\mathcal{C}_2}(x_{\mathcal{C}_2}(k), x_2(k)) \\
&\qquad x_{\mathcal{C}_2}(k+1) \in F_{\mathcal{C}_2}(x_{\mathcal{C}_2}(k), v_{\mathcal{C}_2}(k)) \\
&\qquad\qquad u_1(k) \in h_1(\nu_1(k)) \\
&\qquad\qquad x_2(k+1) \in h_2(\nu_2(k)) \\
&\qquad\qquad\qquad x_1(k+1) \in F_1(x_1(k), u_1(k)).
\end{aligned} \tag{4.14}$$

△

Note that similarly to (4.11), the order of equations in (4.14) may vary depending on the variables ν_1 and ν_2 of h_1 and h_2. In the next section, we will provide a specific implementation of the concretized controller $\mathcal{C}_1$ for each of the considered system relations.

The following theorem demonstrates that the concretized controller $\mathcal{C}_1$, formed by connecting the interface with any controller $\mathcal{C}_2$, satisfies the required concretization condition (3.5).

Theorem 4.11. *Consider two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1) and an interface (h_1, h_2, R) for $\mathcal{S}_1$ and $\mathcal{S}_2$. For any controller $\mathcal{C}_2$ in (4.13) f.c. with $\mathcal{S}_2$, the concretized controller $\mathcal{C}_1$ (Definition 4.10) satisfies $\mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2))$.*

Proof. By Definition 4.10, it holds that $\boldsymbol{x}_1 \in \mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1)$ if and only if there exists sequences $\boldsymbol{u}_1, \boldsymbol{x}_2, \boldsymbol{u}_2, \boldsymbol{x}_{\mathcal{C}_2}, \boldsymbol{v}_{\mathcal{C}_2}$ satisfying (4.14). We prove the result by induction on k. Assume $(x_1(k), x_2(k)) \in R$. At each time step, $(u_2(k), v_{\mathcal{C}_2}(k)) \in H_{\mathcal{C}_2}(x_{\mathcal{C}_2}(k), x_2(k))$, implying $u_2(k) \in \mathcal{U}_{\mathcal{S}_2}(x_2(k))$ since $\mathcal{C}_2$ is f.c. with $\mathcal{S}_2$. Since $u_1(k) \in h_1(\nu_1(k))$ and $x_2(k+1) \in h_2(\nu_2(k))$, by (4.11), we have $x_2(k+1) \in F_2(x_2(k), u_2(k))$ and $(x_1(k+1), x_2(k+1)) \in R$, with

$x_1(k+1) \in F_1(x_1(k), u_1(k))$. Thus, $(\boldsymbol{u}_2, \boldsymbol{x}_2) \in \mathcal{B}(\mathcal{C}_2 \times \mathcal{S}_2)$, establishing that $\mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2))$. □

4.4 Interface for simulation relations

In this section, we aim to identify an interface for each system relation type in Table 4.1. Specifically, we seek to define the set-valued functions h_1^{T} and h_2^{T} for each relation type T, ensuring that if $\mathcal{S}_1 \preceq_R^{\mathrm{T}} \mathcal{S}_2$, then $(h_1^{\mathrm{T}}, h_2^{\mathrm{T}}, R)$ is an interface for $\mathcal{S}_1$ and $\mathcal{S}_2$, i.e., satisfying (4.11).

Similarly to how the tree structure in Figure 4.4 systematically enumerates all valid system relations that refine the ASR, it also allows for the enumeration of different interfaces. The blue labels on the edges from the root to a leaf represent the logical operations applied to (4.15) to construct the system relation T (Section 4.2.2), while the intermediate nodes specify the arguments ν_1 for the function h_1^{T} and ν_2 for h_2^{T} that define the interface of the relation. This tree structure also facilitates the meaningful grouping of system relations according to their control architectures, as shown in Figure 4.5.

We now provide implementations of these interfaces for some system relations in Table 4.1.

Definition 4.12. Let $\mathrm{T} \in \{\mathrm{ASR}, \mathrm{PSR}, \mathrm{FAR}, \mathrm{MCR}, \mathrm{FRR}\}$[3], two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$. We define their set-valued functions h_1^{T} and h_2^{T} (Table 4.1) as follows

$$
\begin{aligned}
h_1^{\mathrm{T}} &= I_R^{\mathrm{T}}, \\
h_2^{\mathrm{ASR}}(x_2, u_2, x_1^+) &= F_2(x_2, u_2) \cap R(x_1^+), \\
h_2^{\mathrm{PSR}}(x_2, u_2, x_1, u_1) &= F_2(x_2, u_2) \cap \{x_2^+ \in \mathcal{X}_2 : F_1(x_1, u_1) \subseteq R^{-1}(x_2^+)\}, \\
h_2^{\mathrm{FAR}}(x_2, u_2) &= F_2(x_2, u_2) \cap \{x_2^+ \in \mathcal{X}_2 : \forall x_1 \in R^{-1}(x_2), \\
&\qquad \exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1) : F_1(x_1, u_1) \subseteq R^{-1}(x_2^+)\}, \\
h_2^{\mathrm{FRR}}(x_1^+) &= h_2^{\mathrm{MCR}}(x_1^+) = R(x_1^+),
\end{aligned}
$$

where I_R^{T} is defined in (4.5). △

[3]Note that DSR is not further discussed here, as it adds little to the current discussion, though results can be easily derived for DSR.

The following proposition establishes that the previously defined interfaces guarantee that the post-condition is met, given the pre-condition in (4.11).

Proposition 4.13. *Let* $\mathrm{T} \in \{\mathrm{ASR}, \mathrm{PSR}, \mathrm{FAR}, \mathrm{MCR}, \mathrm{FRR}\}$, *two simple systems* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *in* (3.1), *and a relation* $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$. *If* $\mathcal{S}_1 \preceq_R^{\mathrm{T}} \mathcal{S}_2$, *then the tuple* $(h_1^{\mathrm{T}}, h_2^{\mathrm{T}}, R)$, *with* h_1^{T} *and* h_2^{T} *in Definition 4.12, is an interface for* $\mathcal{S}_1$ *and* $\mathcal{S}_2$, *i.e., it satisfies* (4.11).

Proof. This follows directly from the definition of I_R^{T} in (4.5). □

The main difference between the functions h_1^{T} and h_2^{T} across the different types T lies in the arguments they take, as listed in Table 4.1. We illustrate the tree construction (Figure 4.4) with the following three relations.

4.4.1 Alternating simulation relation

The condition for ASR (3.8) can be reformulated as

$$\boxed{\forall x_2 \in \mathcal{X}_2}\ \boxed{\forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)}\ \boxed{\forall x_1 \in R^{-1}(x_2)}\ \boxed{\exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)}$$
$$\boxed{\forall x_1^+ \in F_1(x_1, u_1)}\ \boxed{\exists x_2^+ \in F_2(x_2, u_2)} : x_2^+ \in R(x_1^+). \quad (4.15)$$

Following the approach in Section 4.2.2, we decompose the logical formula into blocks corresponding to the variables x_2, u_2, x_1, u_1, x_1^+, and x_2^+. The functions h_1^{ASR} and h_2^{ASR} provide the values for u_1 and x_2^+ based on the existential quantifiers in the u_1 and x_2^+ blocks. Since the ordering is $x_2 < u_2 < x_1 < u_1 < x_1^+ < x_2^+$, the choice of u_1 (respectively, x_2^+) depends on x_2, u_2, x_1 (respectively, x_2, u_2, x_1^+).

Thus, we define $h_1^{\mathrm{ASR}} : \mathcal{X}_2 \times \mathcal{U}_2 \times \mathcal{X}_1 \to 2^{\mathcal{U}_1}$ and $h_2^{\mathrm{ASR}} : \mathcal{X}_2 \times \mathcal{U}_2 \times \mathcal{X}_1 \to 2^{\mathcal{X}_2}$ to ensure

$$\begin{aligned} &\text{Pre: } (x_1, x_2) \in R,\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) \\ &\quad u_1 \in h_1^{\mathrm{ASR}}(x_2, u_2, x_1), \\ &\quad x_1^+ \in F_1(x_1, u_1), \\ &\quad x_2^+ \in h_2^{\mathrm{ASR}}(x_2, u_2, x_1^+), \\ &\text{Post: } x_2^+ \in F_2(x_2, u_2),\ (x_1^+, x_2^+) \in R \end{aligned} \quad (4.16)$$

A valid implementation of the set-valued functions h_1^{ASR} and h_2^{ASR} is provided in Definition 4.12.

As shown in Figure 4.2a, given $(x_1, x_2) \in R$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, the control architecture (4.16) with h_1^{ASR} and h_2^{ASR} (Definition 4.12) ensures that if the concrete system transitions to $x_1' \in F_1(x_1, u_1)$, then $x_2^+ = x_2''$ since $h_2^{\text{ASR}}(x_2, u_2, x_1') = F_2(x_2, u_2) \cap R(x_1') = \{x_2''\}$. This approach avoids quantizing to x_2''', which, although related to x_1' (i.e., $x_2''' \in R(x_1')$), does not belong to $F_2(x_2, u_2)$.

4.4.2 Predictive simulation relation

Certain system relations, like the predictive simulation relation (Definition 4.2), allow for multiple interfaces. The condition for PSR (4.1) can be expressed as

$$\boxed{\forall x_2 \in \mathcal{X}_2}\ \boxed{\forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)}\ \boxed{\forall x_1 \in R^{-1}(x_2)}\ \boxed{\exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)}$$
$$\boxed{\exists x_2^+ \in F_2(x_2, u_2)}\ \boxed{\forall x_1^+ \in F_1(x_1, u_1)}\ :\ x_2^+ \in R(x_1^+). \tag{4.17}$$

This condition leads to an interface characterized by the set-valued functions $h_{1,a}^{\text{PSR}} : \mathcal{X}_2 \times \mathcal{U}_2 \times \mathcal{X}_1 \to 2^{\mathcal{U}_1}$ and $h_{2,a}^{\text{PSR}} : \mathcal{X}_2 \times \mathcal{U}_2 \times \mathcal{X}_1 \times \mathcal{U}_1 \to 2^{\mathcal{X}_2}$, which ensure

$$\begin{aligned}
&\text{Pre: } (x_1, x_2) \in R,\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) \\
&\quad u_1 \in h_{1,a}^{\text{PSR}}(x_2, u_2, x_1), \\
&\quad x_2^+ \in h_{2,a}^{\text{PSR}}(x_2, u_2, x_1, u_1), \\
&\quad x_1^+ \in F_1(x_1, u_1), \\
&\text{Post: } x_2^+ \in F_2(x_2, u_2),\ (x_1^+, x_2^+) \in R
\end{aligned}
\qquad
\begin{aligned}
&\text{Pre: } (x_1, x_2) \in R,\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) \\
&\quad x_2^+ \in h_{2,b}^{\text{PSR}}(x_2, u_2, x_1), \\
&\quad u_1 \in h_{1,b}^{\text{PSR}}(x_2, u_2, x_1, x_2^+), \\
&\quad x_1^+ \in F_1(x_1, u_1), \\
&\text{Post: } x_2^+ \in F_2(x_2, u_2),\ (x_1^+, x_2^+) \in R.
\end{aligned} \tag{4.18}$$

A valid implementation of the set-valued functions $h_{1,a}^{\text{PSR}}$ and $h_{2,a}^{\text{PSR}}$ is provided in Definition 4.12.

The condition for PSR (4.17) can also be reformulated as

$$\boxed{\forall x_2 \in \mathcal{X}_2}\ \boxed{\forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)}\ \boxed{\forall x_1 \in R^{-1}(x_2)}\ \boxed{\exists x_2^+ \in F_2(x_2, u_2)}$$
$$\boxed{\exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)}\ \boxed{\forall x_1^+ \in F_1(x_1, u_1)}\ :\ x_2^+ \in R(x_1^+), \tag{4.19}$$

which indicates that x_2^+ could be determined before u_1, leading to a different interface. While the logical formula remains unchanged by this permutation, (4.17) $\Leftrightarrow$ (4.19), the non-deterministic nature of the functions

requires a different control architecture characterized by $h^{\text{PSR}}_{1,b} : \mathcal{X}_2 \times \mathcal{U}_2 \times \mathcal{X}_1 \times \mathcal{X}_2 \to 2^{\mathcal{U}_1}$ and $h^{\text{PSR}}_{2,b} : \mathcal{X}_2 \times \mathcal{U}_2 \times \mathcal{X}_1 \to 2^{\mathcal{X}_2}$, as shown in (4.18).

If the function $h^{\text{PSR}}_{1,a}$ (resp. $h^{\text{PSR}}_{2,b}$) is deterministic, we can define

$$h^{\text{PSR}}_{2,c}(x_2, u_2, x_1) := h^{\text{PSR}}_{2,a}(x_2, u_2, x_1, h^{\text{PSR}}_{1,a}(x_2, u_2, x_1))$$

(resp. $h^{\text{PSR}}_{1,c}(x_2, u_2, x_1) := h^{\text{PSR}}_{1,b}(x_2, u_2, x_1, h^{\text{PSR}}_{2,b}(x_2, u_2, x_1))$), with $h^{\text{PSR}}_{1,c} := h^{\text{PSR}}_{1,a}$ (resp. $h^{\text{PSR}}_{2,c} := h^{\text{PSR}}_{2,b}$), resulting in the following control architecture

$$\begin{aligned}
&\text{Pre: } (x_1, x_2) \in R,\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) \\
&\quad u_1 \in h^{\text{PSR}}_{1,c}(x_2, u_2, x_1), \\
&\quad x_2^+ \in h^{\text{PSR}}_{2,c}(x_2, u_2, x_1), \\
&\quad x_1^+ \in F_1(x_1, u_1), \\
&\text{Post: } x_2^+ \in F_2(x_2, u_2),\ (x_1^+, x_2^+) \in R.
\end{aligned} \tag{4.20}$$

In contrast to the interface of ASR (4.16), where the next abstract state x_2^+ can only be determined after the concrete control action u_1, the interface for PSR (4.18) allows for predicting x_2^+ beforehand.

As illustrated in Figure 4.2b, given $(x_1, x_2) \in R$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, the control architecture (4.18) ensures that $x_2^+ = x_2'$ can be predicted before applying u_1 to x_1, since $h^{\text{PSR}}_{2,a}(x_2, u_2, x_1, u_1) = \{x_2'\}$, ensuring $x_2' \in F_2(x_2, u_2) \cap R(x_1^+)$ for any $x_1^+ \in F_1(x_1, u_1)$.

4.4.3 Memoryless concretization relation

The condition for MCR (4.4) is expressed as

$$\boxed{\forall x_2 \in \mathcal{X}_2}\ \boxed{\forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)}\ \boxed{\forall x_1 \in R^{-1}(x_2)}\ \boxed{\exists u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)}$$
$$\boxed{\forall x_1^+ \in F_1(x_1, u_1)}\ \boxed{\forall x_2^+ \in R(x_1^+)} :\ x_2^+ \in F_2(x_2, u_2). \tag{4.21}$$

Given the universal quantifier in the x_2^+ block, the control architecture is characterized by the set-valued functions $h^{\text{MCR}}_1 : \mathcal{X}_2 \times \mathcal{U}_2 \times \mathcal{X}_1 \to 2^{\mathcal{U}_1}$

and $h_2^{\mathrm{MCR}} : \mathcal{X}_1 \to 2^{\mathcal{X}_2}$, ensuring

$$\begin{aligned} &\text{Pre: } (x_1, x_2) \in R,\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) \\ &\quad u_1 \in h_1^{\mathrm{MCR}}(x_2, u_2, x_1), \\ &\quad x_1^+ \in F_1(x_1, u_1), \\ &\quad x_2^+ \in h_2^{\mathrm{MCR}}(x_1^+), \\ &\text{Post: } x_2^+ \in F_2(x_2, u_2),\ (x_1^+, x_2^+) \in R. \end{aligned} \tag{4.22}$$

A valid implementation of the set-valued functions h_1^{MCR} and h_2^{MCR} is provided in Definition 4.12.

As illustrated in Figure 4.2c, for $(x_1, x_2) \in R$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, the interface (4.22) ensures that for any $x_1^+ \in F_1(x_1, u_1)$, every $x_2^+ \in R(x_1^+)$ will belong to $F_2(x_2, u_2)$.

4.5 Concretization procedure

With the interfaces for each system relation identified in the previous section, their concretization procedure (3.5) directly follows from constructing their respective concretized controller (Definition 4.10).

Given the system relation $\mathrm{T} \in \{\mathrm{ASR}, \mathrm{PSR}, \mathrm{FAR}, \mathrm{MCR}, \mathrm{FRR}\}$, we now define in our control formalism Chapter 2 the concretized controller (Definition 4.10)

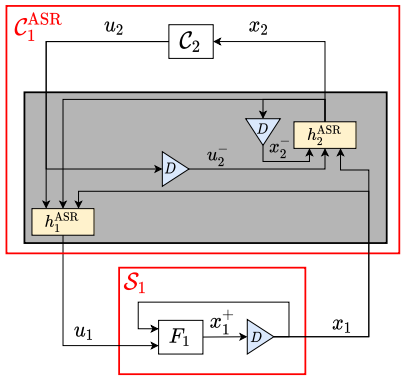
$$\mathcal{C}_1^{\mathrm{T}} = (\mathcal{X}_{\mathcal{C}_1^{\mathrm{T}}}, \mathcal{X}_1, \mathcal{V}_{\mathcal{C}_1^{\mathrm{T}}}, \mathcal{U}_1, F_{\mathcal{C}_1^{\mathrm{T}}}, H_{\mathcal{C}_1^{\mathrm{T}}}), \tag{4.23}$$

resulting from connecting $\mathcal{C}_2$ (4.13) to the interface $(h_1^{\mathrm{T}}, h_2^{\mathrm{T}}, R)$ given in Definition 4.12, as illustrated in Figure 4.6.

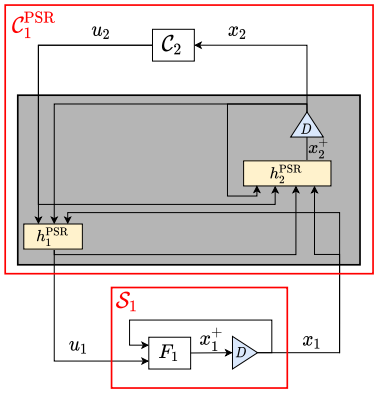
Definition 4.14. Given $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and $\mathcal{C}_2$ in (4.13), the concrete controller $\mathcal{C}_1^{\mathrm{ASR}}$ in (4.23) is defined such that $\mathcal{X}_{\mathcal{C}_1^{\mathrm{ASR}}} = \mathcal{X}_{\mathcal{C}_2} \times \mathcal{X}_2 \times \mathcal{U}_2$, $\mathcal{V}_{\mathcal{C}_1^{\mathrm{ASR}}} = \mathcal{V}_{\mathcal{C}_2} \times \mathcal{X}_2 \times \mathcal{U}_2$, and

$$\begin{aligned} F_{\mathcal{C}_1^{\mathrm{ASR}}}((x_{\mathcal{C}_2}, x_2^-, u_2^-), (v_{\mathcal{C}_2}, x_2, u_2)) &= \{(x_{\mathcal{C}_2}^+, x_2, u_2) \mid x_{\mathcal{C}_2}^+ \in F_{\mathcal{C}_2}(x_{\mathcal{C}_2}, v_{\mathcal{C}_2})\}, \\ H_{\mathcal{C}_1^{\mathrm{ASR}}}((x_{\mathcal{C}_2}, x_2^-, u_2^-), x_1) &= \{(u_1, (v_{\mathcal{C}_2}, x_2, u_2)) \mid x_2 \in h_2^{\mathrm{ASR}}(x_2^-, u_2^-, x_1), \\ &\qquad (u_2, v_{\mathcal{C}_2}) \in H_{\mathcal{C}_2}(x_{\mathcal{C}_2}, x_2), u_1 \in h_1^{\mathrm{ASR}}(x_2, u_2, x_1)\}, \end{aligned}$$

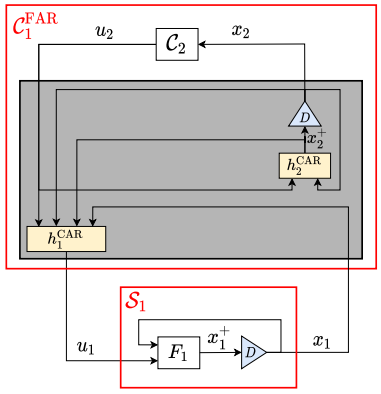
where h_1^{ASR} and h_2^{ASR} are given in Definition 4.12. △



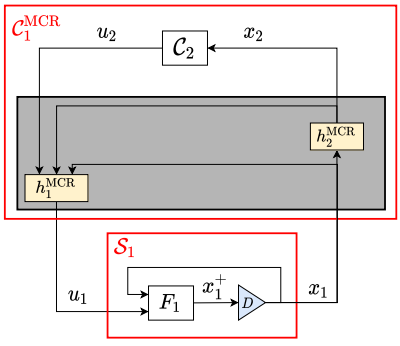
(a) ASR.



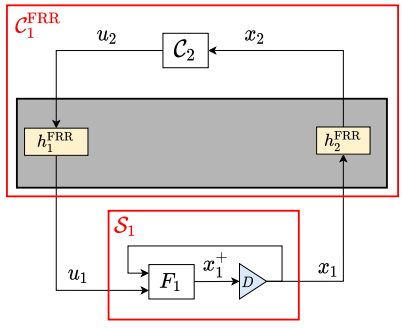
(b) PSR.



(c) FAR.



(d) MCR.



(e) FRR.

Fig. 4.6 Feedback composition of the concrete system $\mathcal{S}_1$ with the concretized controller $\mathcal{C}_1^{\text{T}}$, derived from the abstract controller $\mathcal{C}_2$ and the interface $(h_1^{\text{T}}, h_2^{\text{T}}, R)$, for some system relation T in Table 4.1.

Therefore, the controller $\mathcal{C}_1^{\mathrm{ASR}}$ in Definition 4.14 is a valid implementation of the concrete controller in Theorem 3.4, ensuring the satisfaction of (3.5).

Definition 4.15. Given $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and $\mathcal{C}_2$ in (4.13), the concrete controller $\mathcal{C}_1^{\mathrm{PSR}}$ in (4.23) is defined such that $\mathcal{X}_{\mathcal{C}_1^{\mathrm{PSR}}} = \mathcal{X}_{\mathcal{C}_2} \times \mathcal{X}_2$, $\mathcal{V}_{\mathcal{C}_1^{\mathrm{PSR}}} = \mathcal{V}_{\mathcal{C}_2} \times \mathcal{X}_2$, and

$$
\begin{aligned}
F_{\mathcal{C}_1^{\mathrm{PSR}}}((x_{\mathcal{C}_2}, x_2), (v_{\mathcal{C}_2}, x_2^+)) &= \{(x_{\mathcal{C}_2}^+, x_2^+) \mid x_{\mathcal{C}_2}^+ \in F_{\mathcal{C}_2}(x_{\mathcal{C}_2}, v_{\mathcal{C}_2})\},\\
H_{\mathcal{C}_1^{\mathrm{PSR}}}((x_{\mathcal{C}_2}, x_2), x_1) &= \{(u_1, (v_{\mathcal{C}_2}, x_2^+)) \mid (u_2, v_{\mathcal{C}_2}) \in H_{\mathcal{C}_2}(x_{\mathcal{C}_2}, x_2),\\
&\qquad u_1 \in h_1^{\mathrm{PSR}}(x_2, u_2, x_1), x_2^+ \in h_2^{\mathrm{PSR}}(x_2, u_2, x_1, u_1)\},
\end{aligned}
$$

where h_1^{PSR} and h_2^{PSR} are given in Definition 4.12. △

Note that the interfaces in (4.16) and (4.18) are slight generalizations of those presented in Example 2.2 and Example 2.3, respectively, justifying the use of the control framework introduced in Chapter 2 to define $\mathcal{C}_1^{\mathrm{ASR}}$ and $\mathcal{C}_1^{\mathrm{PSR}}$.

Definition 4.16. Given $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and $\mathcal{C}_2$ in (4.13), the concrete controller $\mathcal{C}_1^{\mathrm{FAR}}$ in (4.23) is defined such that $\mathcal{X}_{\mathcal{C}_1^{\mathrm{FAR}}} = \mathcal{X}_{\mathcal{C}_2} \times \mathcal{X}_2$, $\mathcal{V}_{\mathcal{C}_1^{\mathrm{FAR}}} = \mathcal{V}_{\mathcal{C}_2} \times \mathcal{X}_2$, and

$$
\begin{aligned}
F_{\mathcal{C}_1^{\mathrm{FAR}}}((x_{\mathcal{C}_2}, x_2), (v_{\mathcal{C}_2}, x_2^+)) &= \{(x_{\mathcal{C}_2}^+, x_2^+) \mid x_{\mathcal{C}_2}^+ \in F_{\mathcal{C}_2}(x_{\mathcal{C}_2}, v_{\mathcal{C}_2})\},\\
H_{\mathcal{C}_1^{\mathrm{FAR}}}((x_{\mathcal{C}_2}, x_2), x_1) &= \{(u_1, (v_{\mathcal{C}_2}, x_2^+)) \mid (u_2, v_{\mathcal{C}_2}) \in H_{\mathcal{C}_2}(x_{\mathcal{C}_2}, x_2),\\
&\qquad x_2^+ \in h_2^{\mathrm{FAR}}(x_2, u_2), u_1 \in h_1^{\mathrm{FAR}}(x_2, u_2, x_1, x_2^+)\},
\end{aligned}
$$

where h_1^{FAR} and h_2^{FAR} are given in Definition 4.12. △

Definition 4.17. Given $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and $\mathcal{C}_2$ in (4.13), the concrete controller $\mathcal{C}_1^{\mathrm{MCR}}$ in (4.23) is defined such that $\mathcal{X}_{\mathcal{C}_1^{\mathrm{MCR}}} = \mathcal{X}_{\mathcal{C}_2}$, $\mathcal{V}_{\mathcal{C}_1^{\mathrm{MCR}}} = \mathcal{V}_{\mathcal{C}_2}$, and

$$
\begin{aligned}
F_{\mathcal{C}_1^{\mathrm{MCR}}}(x_{\mathcal{C}_2}, v_{\mathcal{C}_2}) &= F_{\mathcal{C}_2}(x_{\mathcal{C}_2}, v_{\mathcal{C}_2}),\\
H_{\mathcal{C}_1^{\mathrm{MCR}}}(x_{\mathcal{C}_2}, x_1) &= \{(u_1, v_{\mathcal{C}_2}) \mid x_2 \in h_2^{\mathrm{MCR}}(x_1)\\
&\qquad (u_2, v_{\mathcal{C}_2}) \in H_{\mathcal{C}_2}(x_{\mathcal{C}_2}, x_2), u_1 \in h_1^{\mathrm{MCR}}(x_2, u_2, x_1)\},
\end{aligned}
$$

where h_1^{MCR} and h_2^{MCR} are given in Definition 4.12. △

Definition 4.18. Given $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and $\mathcal{C}_2$ in (4.13), the concrete

controller $\mathcal{C}_1^{\text{FRR}}$ in (4.23) is defined such that $\mathcal{X}_{\mathcal{C}_1^{\text{FRR}}} = \mathcal{X}_{\mathcal{C}_2}$, $\mathcal{V}_{\mathcal{C}_1^{\text{FRR}}} = \mathcal{V}_{\mathcal{C}_2}$, and

$$\begin{aligned} F_{\mathcal{C}_1^{\text{FRR}}}(x_{\mathcal{C}_2}, v_{\mathcal{C}_2}) &= F_{\mathcal{C}_2}(x_{\mathcal{C}_2}, v_{\mathcal{C}_2}), \\ H_{\mathcal{C}_1^{\text{FRR}}}(x_{\mathcal{C}_2}, x_1) &= \{(u_1, v_{\mathcal{C}_2}) \mid x_2 \in h_2^{\text{FRR}}(x_1), \\ &\qquad (u_2, v_{\mathcal{C}_2}) \in H_{\mathcal{C}_2}(x_{\mathcal{C}_2}, x_2), u_1 \in h_1^{\text{FRR}}(u_2)\}, \end{aligned}$$

where h_1^{FRR} and h_2^{FRR} are given in Definition 4.12. △

The simple controller in (3.10) for the feedback refinement relation is recovered within our framework (Definition 4.18) as

$$\begin{aligned} H_{\mathcal{C}_1^{\text{FRR}}}(x_{\mathcal{C}_2}, x_1) &= \{(u_2, v_{\mathcal{C}_2}) \mid x_2 \in R(x_1), (u_2, v_{\mathcal{C}_2}) \in H_{\mathcal{C}_2}(x_{\mathcal{C}_2}, x_2)\} \\ &= H_{\mathcal{C}_2}(x_{\mathcal{C}_2}, R(x_1)), \end{aligned}$$

which establishes that $\mathcal{C}_1^{\text{FRR}} = \mathcal{C}_2 \circ R$ as defined in (2.13). Consequently, the architecture in Figure 4.1 (right) equivalent to Figure 4.6e simplifies to the architecture in Figure 4.1 (left).

The validity of the concretization procedures is established by the following corollary of Theorem 4.11.

Corollary 4.19. *Let* $\text{T} \in \{\text{ASR}, \text{PSR}, \text{FAR}, \text{MCR}, \text{FRR}\}$, *two simple systems* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *in* (3.1), *and a relation* $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$. *If* $\mathcal{S}_1 \preceq_R^{\text{T}} \mathcal{S}_2$, *then for any controller* $\mathcal{C}_2$ *f.c. with* $\mathcal{S}_2$, *the concretized controller* $\mathcal{C}_1^{\text{T}}$ (4.23) *satisfies* $\mathcal{B}_\times(\mathcal{C}_1^{\text{T}} \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2))$.

Proof. First, given that $\mathcal{S}_1 \preceq_R^{\text{T}} \mathcal{S}_2$, Proposition 4.13 ensures that $(h_1^{\text{T}}, h_2^{\text{T}}, R)$ is a valid interface for $\mathcal{S}_1$ and $\mathcal{S}_2$. Next, it is straightforward to verify that the concretized controller $\mathcal{C}_1^{\text{T}}$ satisfies (4.14). The result then follows directly from Theorem 4.11. □

To summarize, the *arguments* of the functions h_1^{T} and h_2^{T}, which define the block representation of the interface (see Figure 4.6), depend on the type of relation T, not on the specific systems $\mathcal{S}_1$ and $\mathcal{S}_2$. While the *implementation* of the blocks h_1^{T} and h_2^{T} depends on the specific systems $\mathcal{S}_1$ and $\mathcal{S}_2$ (see Definition 4.12), it is independent of the controller $\mathcal{C}_2$. This allows the interface to be irrespective of the particular specification, enabling a plug-and-play concretization procedure, as discussed in the introduction of the chapter.

4.6 Properties

In this section, we discuss the properties of the previously introduced relations and their concretization procedures. In particular, the ordering relation between the system relations (Theorem 4.8) is also reflected by their respective interfaces in Figure 4.6.

4.6.1 The concretization complexity issue

The specific implementations of the functions h_2^{ASR}, h_2^{PSR}, and h_2^{FAR} in Definition 4.12 require evaluating the transition map F_2 of $\mathcal{S}_2$. Given the typically large state space of $\mathcal{S}_2$, this results in high complexity when storing or evaluating the controllers $\mathcal{C}_1^{\mathrm{ASR}}$ (Definition 4.14), $\mathcal{C}_1^{\mathrm{PSR}}$ (Definition 4.15), and $\mathcal{C}_1^{\mathrm{FAR}}$ (Definition 4.16). This is what we referred to as the *concretization complexity issue* in the introduction of this chapter.

In contrast, the implementation of h_2^{MCR} in Definition 4.12 depends only on the quantizer R and not on the transition map F_2 of the abstract system. Therefore, once the abstract controller is designed, the abstraction itself is no longer needed. However, the function $h_1^{\mathrm{MCR}} = I_R^{\mathrm{MCR}}$ implicitly encodes the extended relation R_e^{MCR} within the concrete controller. When I_R^{MCR} can be explicitly characterized, it eliminates the need to store the extended relation.

Referring to Example 4.1, Theorem 4.8 ensures that $\mathcal{S}_1 \preceq_R^{\mathrm{MCR}} \mathcal{S}_2'$ since the quantizer R is deterministic. Here, the extended relation is efficiently encoded by a linear function; for instance, given $(x_1, q_1) \in R$, we have $I_R^{\mathrm{MCR}}(q_1, \kappa_1', x_1) = \{-x_1\}$.

4.6.2 Static concretization

Even when the abstract controller $\mathcal{C}_2$ (4.13) is static (i.e., $\mathcal{X}_{\mathcal{C}_2} = \{0\}$), the concretized controllers $\mathcal{C}_1^{\mathrm{ASR}}$ (Definition 4.14), $\mathcal{C}_1^{\mathrm{PSR}}$ (Definition 4.15), and $\mathcal{C}_1^{\mathrm{FAR}}$ (Definition 4.16) are non-static because their state spaces, $\mathcal{X}_{\mathcal{C}_1}^{\mathrm{ASR}} = \{0\} \times \mathcal{X}_2 \times \mathcal{U}_2$ and $\mathcal{X}_{\mathcal{C}_1}^{\mathrm{PSR}} = \mathcal{X}_{\mathcal{C}_1}^{\mathrm{FAR}} = \{0\} \times \mathcal{X}_2$, are not singletons.

In contrast, the concretized controller $\mathcal{C}_1^{\mathrm{MCR}}$ (Definition 4.17) for the memoryless concretization relation requires no additional state variables beyond those in $\mathcal{C}_2$, as indicated by the absence of a delay block in its interface in Figure 4.6d. Therefore, if the abstract controller $\mathcal{C}_2$ is static, then the

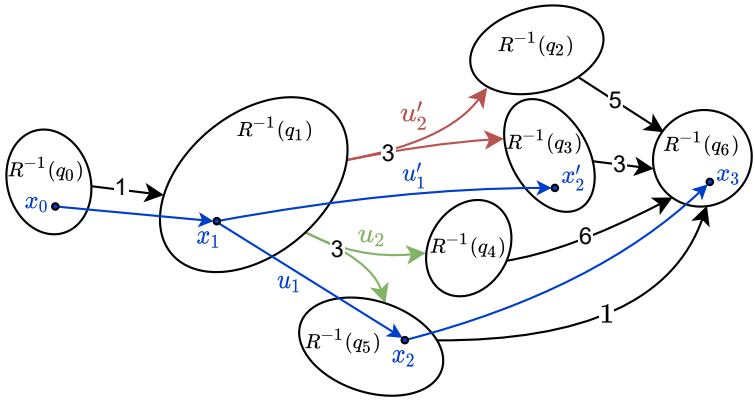


Fig. 4.7 The systems $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$ and $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$, and a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$. Specifically, $\mathcal{X}_1 \subseteq \mathbb{R}^2$, $\mathcal{X}_2 = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$ and $\mathcal{U}_{\mathcal{S}_2}(q_1) = \{u_2', u_2\}$. The transition map F_2 and the relation R are clear from the illustration. A cost is associated to each transition.

concretized controller $\mathcal{C}_1^{\text{MCR}}$ is also static, as established by the following proposition.

Proposition 4.20. *Given the simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$, and a controller $\mathcal{C}_2$ f.c. with $\mathcal{S}_2$, if $\mathcal{C}_2$ is static, then the resulting concretized controller $\mathcal{C}_1^{\text{MCR}}$ (Definition 4.17) is also static.*

Proof. This follows directly from Definition 4.17, where $\mathcal{X}_{\mathcal{C}_1^{\text{MCR}}} = \mathcal{X}_{\mathcal{C}_2}$, $\mathcal{V}_{\mathcal{C}_1^{\text{MCR}}} = \mathcal{V}_{\mathcal{C}_2}$, and $F_{\mathcal{C}_1^{\text{MCR}}} = F_{\mathcal{C}_2}$. Therefore, if $\mathcal{C}_2$ is static (i.e., $\mathcal{X}_{\mathcal{C}_2} = \{0\}$), then $\mathcal{X}_{\mathcal{C}_1^{\text{MCR}}} = \{0\}$, confirming that $\mathcal{C}_1^{\text{MCR}}$ is static. □

4.6.3 Corrective vs predictive control architecture

While $\mathcal{C}_1^{\text{ASR}}$ stores the *current* abstract state x_2 and input u_2 to correct the following related abstract state as $x_2^+ \in F_2(x_2, u_2) \cap R(x_1^+)$ at the next time step, the concretized controller $\mathcal{C}_1^{\text{PSR}}$ for the predictive simulation relation store the *next* abstract state x_2^+, which is determined before applying the current input u_1 to x_1. This distinction is reflected in the delay blocks shown in Figure 4.6b. Consequently, the control architecture of ASR can be interpreted as *corrective*, whereas that of PSR is *predictive*.

This predictive approach, which determines the next abstract state before applying the concrete input to the original system, allows the abstract policy to be updated online. This is particularly beneficial when a cost function is included in the specification as illustrated with the following example.

Example 4.21. Given the systems in Figure 4.7, consider the optimal control problem of guiding the set of concrete states from $R^{-1}(q_0)$ to $R^{-1}(q_6)$ while minimizing the overall cost. The abstract controller solving the associated abstract problem, consisting in guiding $\mathcal{S}_2$ from q_0 to q_6 while minimizing the overall cost, is the static system $\mathcal{C}_2 = (\{0\}, \mathcal{X}_2, \{0\}, \mathcal{U}_2, F_{\mathcal{C}_2}, H_{\mathcal{C}_2})$, where $H_{\mathcal{C}_2}(0, q_1) = \{(u_2', 0)\}$. Indeed, applying u_2' at q_1 results in a worst-case total cost of 9 to reach q_6 from q_0, whereas applying u_2 results in a worst-case total cost of 10. During the simulation of a specific trajectory $x_0 \in R^{-1}(q_0)$, if $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$, the concrete controller will apply $u_1' \in I_R^{\text{ASR}}(q_1, u_2', x_1)$ at x_1, resulting in a total cost of 7 to reach q_6 from q_0. However, if $\mathcal{S}_1 \preceq_R^{\text{PSR}} \mathcal{S}_2$, then at x_1, we can anticipate the next abstract state before applying a concrete control action, which is predicted to be q_5 when applying u_2. Consequently, we can adjust the abstract policy online and apply $u_1 \in I_R^{\text{PSR}}(q_1, u_2, x_1)$, leading to a total cost of 5 to reach q_6 from q_0. △

4.6.4 Feedforward abstract control

Considering the concretized controller $\mathcal{C}_1^{\text{FAR}}$ (Definition 4.16), the subsequent abstract state x_2^+ is determined independently of the concrete variables, including the concrete input u_1, which contrasts with the PSR (see Figure 4.6c). Here, the dynamic of x_2 is entirely determined regardless of the concrete system dynamics. Therefore, an abstract reference trajectory can be designed offline, with the function h_2^{PSR} acting as a feedforward controller at the abstract level, ensuring that the concrete system follows the predefined abstract trajectory.

Referring back to Example 4.1, by Theorem 4.8, we have $\mathcal{S}_1 \preceq_R^{\text{FAR}} \mathcal{S}_2'$ since the abstraction $\mathcal{S}_2'$ is deterministic.

4.6.5 Delayed control

The delayed simulation relation (Definition 4.3), corresponding to relation 5 in Table 4.1, refines PSR (Theorem 4.8). The DSR has a stronger

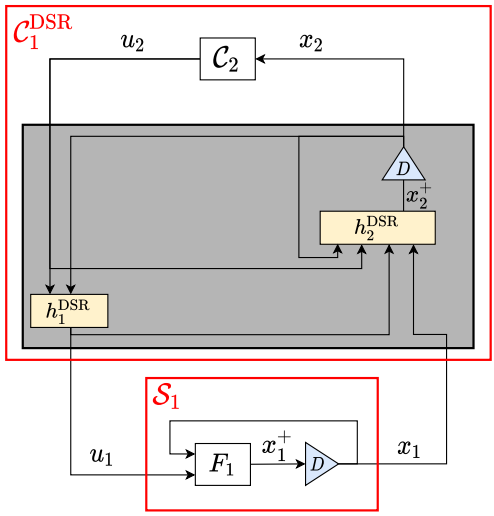


Fig. 4.8 Feedback composition of the concrete system $\mathcal{S}_1$ with the concretized controller $\mathcal{C}_1^{\mathrm{DSR}}$ derived from the abstract controller $\mathcal{C}_2$ for relation 5 in Table 4.1.

predictive property because it benefits from a one-step delay in the measurement of the concrete state x_1^+, since the computation of both x_2^+ and u_1^+ depends only on x_1 and not on x_1^+. We illustrate in Figure 4.8 the closed-loop system with its corresponding concretized controller and interface as given in Table 4.1.

4.7 Constructions

In this section, we present algorithms for constructing the introduced relations and discuss their respective advantages.

4.7.1 System

We study a deterministic simple system

$$\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1), \tag{4.24}$$

where $\mathcal{X}_1 \subseteq \mathbb{R}^n$ is a compact set, $\mathcal{U}_1 \subseteq \mathbb{R}^\nu$, and the transition map $F_1(x,u) = \{f(x,u)\}$ is defined by a continuous nonlinear function $f : \mathcal{X}_1 \times \mathcal{U}_1 \to \mathcal{X}_1$.

The constructions presented in this section rely on some stability notions. As a main tool in classical stability analysis, Lyapunov functions provide a useful method for demonstrating stability. We introduce here a relaxed version of Lyapunov function, referred to as the *growth-bound function*, which can be used to bound the distance between trajectories with different initial conditions under a specific control law. This is the discrete-time analogue of [GPT09, Definition 2.3].

Definition 4.22 (Growth-bound function)**.** A smooth function $V : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_0^+$ is a *growth-bound function* under control $\kappa : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^\nu \to \mathbb{R}^\nu$ for $f : \mathbb{R}^n \times \mathbb{R}^\nu \to \mathbb{R}^n$ if there exist $\mathcal{K}_\infty$ functions $\underline{\alpha}, \overline{\alpha}$ and $\rho > 0$ such that for all $x, y \in \mathbb{R}^n$, $u \in \mathbb{R}^\nu$

$$V(f(y, \kappa(y,x,u)), f(x,u)) \leq \rho V(y,x) \tag{4.25}$$

$$\underline{\alpha}(\|y - x\|_2) \leq V(y,x) \leq \overline{\alpha}(\|y - x\|_2). \tag{4.26}$$

We define the ℓ-sublevel set of $V(\cdot, x)$ as

$$\mathcal{S}(x, \ell) = \{x' \in \mathbb{R}^n \mid V(x', x) \leq \ell\}. \tag{4.27}$$

△

When $\rho < 1$, the system is said to be *incrementally stabilizable*, and the function V is called a δ-GAS Lyapunov function [TRK16, Definition 4]. Intuitively, *incremental stability* means that all the trajectories of the system converge to the same reference trajectory regardless of their initial condition (see Section 2.7).

We impose stabilizability assumptions on $\mathcal{S}_1$ (4.24) by assuming the existence of a growth-bound function V under control κ for f (Definition 4.22) with a contraction factor ρ.

We now introduce a few useful lemmas for the upcoming constructions.

Lemma 4.23. *Let V be a growth-bound function as in Definition 4.22 and parameters $(\eta, \epsilon) > 0$. If $\eta \leq \frac{2}{\sqrt{n}} \overline{\alpha}^{-1}(\epsilon)$, then for all $x \in \mathbb{R}^n$, $x \in \mathcal{S}(x_2, \epsilon)$ with $x_2 = \operatorname{argmin}_{c \in [\eta \mathbb{Z}^n]} \|x - c\|_\infty$.*

Proof. Let $x \in \mathbb{R}^n$ and $x_2 = \operatorname{argmin}_{c \in [\eta \mathbb{Z}^n]} \|x - c\|_\infty$. Then $\|x - x_2\|_\infty \leq \frac{\eta}{2}$, and we have

$$V(x, x_2) \leq \overline{\alpha}(\|x - x_2\|_2) \leq \overline{\alpha}(\sqrt{n}\|x - x_2\|_\infty) \leq \overline{\alpha}\left(\sqrt{n}\tfrac{\eta}{2}\right) \leq \epsilon,$$

where the last inequality holds because $\overline{\alpha}$ is a $\mathcal{K}$ function. □

Lemma 4.24. *Let V be a growth-bound function under control κ for f as in Definition 4.22. Given $x_2 \in \mathbb{R}^n$, $u_2 \in \mathbb{R}^\nu$, and $\epsilon > 0$. Then, the following holds*

$$g_{x_2,u_2}(\mathcal{S}(x_2,\epsilon)) \subseteq \mathcal{S}(f(x_2,u_2),\rho\epsilon), \tag{4.28}$$

where $g_{x_2,u_2}(x) = f(x,\kappa(x,x_2,u_2))$.

Proof. Let $x \in \mathcal{S}(x_2,\epsilon)$. By (4.25), it holds

$$V(f(x,\kappa(x,x_2,u_2)),f(x_2,u_2)) \leq \rho V(x,x_2) \leq \rho\epsilon,$$

which involves $g_{x_2,u_2}(x) \in \mathcal{S}(f(x_2,u_2),\rho\epsilon)$. □

We will make the following supplementary assumption on the growth-bound function: there exists a $\mathcal{K}_\infty$ function γ such that

$$\forall x,y,z \in \mathbb{R}^n : \; V(x,y) - V(x,z) \leq \gamma(\|y-z\|_2). \tag{4.29}$$

While this assumption may seem quite strong, it is not restrictive provided we are interested in the dynamics of the system on a compact subset of the state space $\mathbb{R}^n$, as explained in [GPT09, Section IV.B].

The following lemma is an adaptation of [GPT09, Eq. (9)] for discrete-time systems.

Lemma 4.25. *Let V be a growth-bound function under control κ for f as in Definition 4.22 satisfying (4.29) and parameters $(\eta,\epsilon,\epsilon') > 0$. If $\epsilon' - \rho\epsilon > 0$, then the choice $\eta \leq \frac{2}{\sqrt{n}}\gamma^{-1}(\epsilon' - \rho\epsilon)$ ensures that*

$$\forall x_2 \in \mathbb{R}^n \; \forall u_2 \in \mathbb{R}^\nu : \; g_{x_2,u_2}(\mathcal{S}(x_2,\epsilon)) \subseteq \mathcal{S}(x_2',\epsilon'),$$

with $g_{x_2,u_2}(x) = f(x,\kappa(x,x_2,u_2))$ and $x_2' = \operatorname{argmin}_{x' \in [\eta\mathbb{Z}^n]} \|x' - f(x_2,u_2)\|_\infty$. Special case: when $V(x,y) = \|x-y\|_2$ and $\epsilon = \epsilon'$, the condition on η reduces to $\eta \leq \frac{2}{\sqrt{n}}\epsilon(1-\rho)$.

Proof. Let $x_1 \in \mathcal{S}(x_2,\epsilon)$, $x_1^+ = f(x_1,\kappa(x_1,x_2,u_2))$, $x_2^+ = f(x_2,u_2)$ and $x_2' = \operatorname{argmin}_{x' \in [\eta\mathbb{Z}^n]} \|x' - x_2^+\|_\infty$. Then, $\|x_2' - x_2^+\|_\infty \leq \frac{\eta}{2}$. Using (4.29) and (4.25),

and since γ is a $\mathcal{K}_\infty$ function, we have

$$\begin{aligned} V(x_1^+, x_2') &\leq V(x_1^+, x_2^+) + \gamma(\|x_2' - x_2^+\|_2) \\ &\leq \rho V(x_1, x_2) + \gamma(\sqrt{n}\|x_2' - x_2^+\|_\infty) \\ &\leq \rho\epsilon + \gamma\left(\sqrt{n}\tfrac{\eta}{2}\right) \\ &\leq \epsilon' \end{aligned}$$

which implies that $x_1^+ \in \mathcal{S}(x_2', \epsilon')$. □

We illustrate all the constructions of the different types of relations on the following running planar example.

Example 4.26. We consider a two-dimensional system ($\mathcal{X}_1 \subseteq \mathbb{R}^2$), characterized by the growth-bound function $V(x, y) = \|x - y\|_2$, for which $\underline{\alpha}(x) = \overline{\alpha}(x) = \gamma(x) = x = \underline{\alpha}^{-1}(x) = \overline{\alpha}^{-1}(x) = \gamma^{-1}(x)$, under control κ and contraction factor ρ. Function V satisfies assumption (4.29), which simplifies to the triangular inequality of the 2-norm. △

4.7.2 Abstraction

The abstraction construction relies on discretizing the input space, $\mathcal{U}_2$ is a finite subset of $\mathcal{U}_1$, and discretizing the state-space $\mathcal{X}_1$ into overlapping cells aligned on a uniform grid defined by

$$[\eta\mathbb{Z}^n] = \{c \in \mathbb{R}^n \mid \exists_{k\in\mathbb{Z}^n} \forall_{i\in[1;n]} c_i = k_i\eta\} \tag{4.30}$$

with *grid parameter* $\eta > 0$. The abstract states correspond to the vertices of this lattice, i.e.,

$$\mathcal{X}_2 = [\eta\mathbb{Z}^n] \cap \mathcal{X}_1. \tag{4.31}$$

Given $\epsilon > 0$, an abstract state $x_2 \in \mathcal{X}_2$ is related to the ϵ-sublevel set of the function $V(\cdot, x_2)$:

$$(x_1, x_2) \in R \Leftrightarrow V(x_1, x_2) \leq \epsilon \Leftrightarrow R^{-1}(x_2) = \mathcal{S}(x_2, \epsilon). \tag{4.32}$$

To ensure a strict relation, i.e., $\forall x_1 \in \mathcal{X}_1 : \; R(x_1) \neq \varnothing$, we impose the condition $\eta \leq \frac{2}{\sqrt{n}}\overline{\alpha}^{-1}(\epsilon)$ (see Lemma 4.23). Given $x_2 \in \mathcal{X}_2$ and $u_2 \in \mathcal{U}_2$, the stability assumption (4.25) allows to determine an over-approximation of the attainable set of a cell $R^{-1}(x_2)$ under the control law $g_{x_2,u_2}(x) = f(x, \kappa(x, x_2, u_2))$ (see Lemma 4.24).

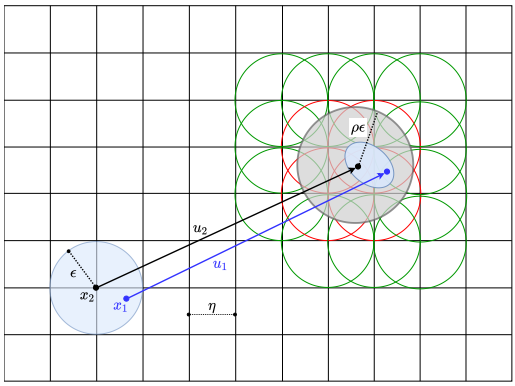


Fig. 4.9 Illustration of $(x_2, u_2, x_1, u_1) \in R_e^{\mathrm{ASR}}$ and R_e^{MCR} for Example 4.26. The cell $R^{-1}(x_2) = \mathcal{S}(x_2, \epsilon)$ and its image under the dynamic F_1 are represented in blue. The grey circle $\mathcal{A} := \mathcal{S}(f(x_2, u_2), \rho\epsilon)$ is the over-approximation of the reachable set of $R^{-1}(x_2)$. The set $F_2^{\mathrm{ASR}}(x_2, u_2)$ just needs to contain enough cells so that their union covers the set $\mathcal{A}$, for example, only the red cells. While the set $F_2^{\mathrm{MCR}}(x_2, u_2)$ must at least contain all the cells intersecting $\mathcal{A}$, which are shown in red and green. Therefore, in this example, we have $|F_2^{\mathrm{ASR}}(x_2, u_2)| = 4$ while $|F_2^{\mathrm{MCR}}(x_2, u_2)| = 15$.

4.7.3 Comparison of control architectures

In this section, we design the transition map F_2^{T} of the abstract system $\mathcal{S}_2^{\mathrm{T}} = (\mathcal{X}_2, \mathcal{U}_2, F_2^{\mathrm{T}})$ to ensure that $\mathcal{S}_1 \preceq_R^{\mathrm{T}} \mathcal{S}_2^{\mathrm{T}}$ with $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ defined in (4.32), and we discuss the differences between the different relation types T.

Proposition 4.27. *The system $\mathcal{S}_2^{\mathrm{ASR}} = (\mathcal{X}_2, \mathcal{U}_2, F_2^{\mathrm{ASR}})$ with F_2^{ASR} satisfying $\forall x_2 \in \mathcal{X}_2,\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$:*

$$R^{-1}(F_2^{\mathrm{ASR}}(x_2, u_2)) \supseteq \mathcal{S}(f(x_2, u_2), \rho\epsilon), \tag{4.33}$$

ensures $\mathcal{S}_1 \preceq_R^{\mathrm{ASR}} \mathcal{S}_2^{\mathrm{ASR}}$. Additionally, one can use the specific implementation $h_1^{\mathrm{ASR}}(x_2, u_2, x_1) = \{\kappa(x_1, x_2, u_2)\}$ for $\mathcal{C}_1^{\mathrm{ASR}}$ in Definition 4.14. Similarly, the system $\mathcal{S}_2^{\mathrm{MCR}} = (\mathcal{X}_2, \mathcal{U}_2, F_2^{\mathrm{MCR}})$ with F_2^{MCR} satisfying $\forall x_2 \in \mathcal{X}_2,\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$:

$$F_2^{\mathrm{MCR}}(x_2, u_2) \supseteq R(\mathcal{S}(f(x_2, u_2), \rho\epsilon)), \tag{4.34}$$

ensures $\mathcal{S}_1 \preceq_R^{\mathrm{MCR}} \mathcal{S}_2^{\mathrm{MCR}}$. Additionally, one can use the specific implementation $h_1^{\mathrm{MCR}}(x_2, u_2, x_1) = \{\kappa(x_1, x_2, u_2)\}$ for $\mathcal{C}_1^{\mathrm{MCR}}$ in Definition 4.17.

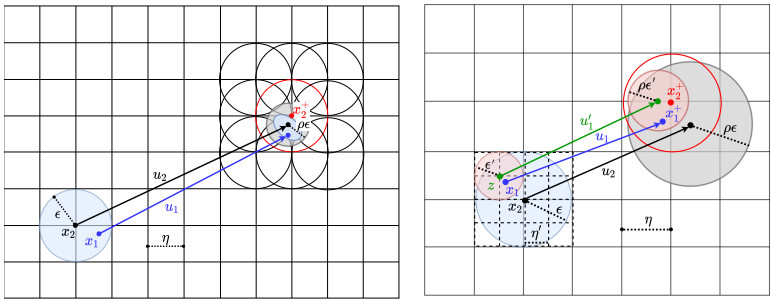


Fig. 4.10 Illustration of $(x_2, u_2, x_1, u_1, x_2^+) \in R_e^{\mathrm{FAR}}$ (left) and $(x_2, u_2, x_1, u_1) \in R_e^{\mathrm{PSR}}$ (right) for Example 4.26.

Proof. Let $(x_1, x_2) \in R$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, then for all $x_1^+ \in F_1(x_1, u_1)$ with $u_1 = \kappa(x_1, x_2, u_2)$, we have $x_1^+ \in \mathcal{S}(f(x_2, u_2), \rho\epsilon)$ by (4.28).

- Condition (4.33) involves $x_1^+ \in \mathcal{S}(f(x_2, u_2), \rho\epsilon) \subseteq R^{-1}(F_2^{\mathrm{ASR}}(x_2, u_2))$, which implies that $R(x_1^+) \cap F_2^{\mathrm{ASR}}(x_2, u_2) \neq \varnothing$. This establishes (3.8) and that $\kappa(x_1, x_2, u_2) \in I_R^{\mathrm{ASR}}(x_2, u_2, x_1)$.
- Condition (4.34) involves $R(x_1^+) \subseteq F_2(x_2, u_2)$, which establishes (4.4) and that $\kappa(x_1, x_2, u_2) \in I_R^{\mathrm{MCR}}(x_2, u_2, x_1)$.

□

While the MCR guarantees a simpler concretization procedure compared to ASR as discussed in Section 4.6.1, the resulting abstraction $\mathcal{S}_2^{\mathrm{MCR}}$ is more non-deterministic than $\mathcal{S}_2^{\mathrm{ASR}}$. In this context, we measure the level of non-determinism by the cardinality of the transition map. Specifically, $|F_2^{\mathrm{ASR}}(x_2, u_2)| \leq |F_2^{\mathrm{MCR}}(x_2, u_2)|$ because $F_2^{\mathrm{ASR}}(x_2, u_2) \subseteq F_2^{\mathrm{MCR}}(x_2, u_2)$, as illustrated in Figure 4.9. Consequently, due to this increased non-determinism, the abstract control problem may be infeasible for $\mathcal{S}_2^{\mathrm{MCR}}$ even if it has a feasible solution for $\mathcal{S}_2^{\mathrm{ASR}}$.

Proposition 4.28. *Given the additional assumption that*

$$\rho < 1 \text{ and } \eta \leq \tfrac{2}{\sqrt{n}} \min\left(\overline{\alpha}^{-1}(\epsilon), \gamma^{-1}((1-\rho)\epsilon)\right),$$

the system $\mathcal{S}_2^{\mathrm{FAR}} = (\mathcal{X}_2, \mathcal{U}_2, F_2^{\mathrm{FAR}})$ *with* F_2^{FAR} *satifying* $\forall x_2 \in \mathcal{X}_2,\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$*:*

$$F_2^{\mathrm{FAR}}(x_2, u_2) \supseteq \operatorname{argmin}_{x_2' \in \mathcal{X}_2} \|f(x_2, u_2) - x_2'\|_\infty, \tag{4.35}$$

ensures $\mathcal{S}_1 \preceq_R^{\mathrm{FAR}} \mathcal{S}_2^{\mathrm{FAR}}$. *Additionally, one can use the concretized controller* $\mathcal{C}_1^{\mathrm{FAR}}$ *(Definition 4.16) with the specific implementations* $h_1^{\mathrm{FAR}}(x_2, u_2, x_1, x_2^+) = \{\kappa(x_1, x_2, u_2)\}$ *and* $h_2^{\mathrm{FAR}}(x_2, u_2) = \operatorname{argmin}_{x_2' \in \mathcal{X}_2} \|f(x_2, u_2) - x_2'\|_\infty$.

Proof. Applying Lemma 4.25 with parameters $(\eta, \epsilon, \epsilon)$ and $\rho < 1$, the condition on η ensures that $\forall x_2 \in \mathcal{X}_2,\ \forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) :\ g_{x_2,u_2}(R^{-1}(x_2)) \subseteq R^{-1}(x_2^+)$ with $x_2^+ = \operatorname{argmin}_{x_2' \in \mathcal{X}_2} \|f(x_2, u_2) - x_2'\|_\infty$. Therefore, by (4.35), $x_2^+ \in F_2^{\mathrm{FAR}}(x_2, u_2)$ and for all $x_1^+ \in F_1(x_1, u_1)$ with $u_1 = \kappa(x_1, x_2, u_2)$, we have $(x_1^+, x_2^+) \in R$. This establishes (4.3), $\kappa(x_1, x_2, u_2) \in I_R^{\mathrm{FAR}}(x_2, u_2, x_1, x_2^+)$ and h_2^{FAR} implements Definition 4.12. □

The choice of $F_2^{\mathrm{FAR}}(x_2, u_2) = \{h_2^{\mathrm{FAR}}(x_2, u_2)\}$ results in a deterministic abstraction as illustrated in Figure 4.10 (left). In this context, constructing $\mathcal{S}_2^{\mathrm{FAR}}$ involves low computational complexity. Designing an abstract transition only requires computing the trajectory of a single point $f(x_2, u_2)$ and identifying the closest grid point. However, unlike other approaches, this method requires a strong assumption on the system, specifically incremental stability ($\rho < 1$). In [CBLJ24a, Section 6], the authors construct a feedforward abstraction relation $\mathcal{S}_2^{\mathrm{FAR}}$ for an incrementally stable dynamical system, specifically a DC-DC converter defined in [RZ16, Section 4.2], and illustrate the speed-up in the construction of the abstraction resulting from this efficient implementation of h_2^{FAR}. Most abstractions based on approximate bisimulation [GPT09] are specific instances of FAR.

We now construct a sub-grid $[\eta'\mathbb{Z}^n]$ of parameter $0 < \eta' < \eta$ to cover the cells $R^{-1}(x_2)$ for $x_2 \in \mathcal{X}_2$ with ϵ'-sublevel set of V aligned on vertices of this new sub-lattice. Given $x_2 \in \mathcal{X}_2$, we denote by $\mathcal{Z}_{\eta,\eta',\epsilon,\epsilon'}(x_2) \subseteq [\eta'\mathbb{Z}^n]$ a set of minimal cardinality such that

$$R^{-1}(x_2) \subseteq \bigcup_{z \in \mathcal{Z}_{\eta,\eta',\epsilon,\epsilon'}(x_2)} \mathcal{S}(z, \epsilon'). \tag{4.36}$$

Proposition 4.29. *Let* $\mathcal{Z}_{\eta,\eta',\epsilon,\epsilon'}(x_2)$ *defined in* (4.36) *with parameters satisfying*

$$\begin{cases} \eta' \leq \frac{2}{\sqrt{n}} \overline{\alpha}^{-1}(\epsilon'), & (4.37) \\ \rho\epsilon' < \epsilon, & (4.38) \\ \eta \leq \frac{2}{\sqrt{n}} \min\left(\overline{\alpha}^{-1}(\epsilon), \gamma^{-1}(\epsilon - \rho\epsilon')\right). & (4.39) \end{cases}$$

The system $\mathcal{S}_2^{\mathrm{PSR}} = (\mathcal{X}_2, \mathcal{U}_2, F_2^{\mathrm{PSR}})$ *with* F_2^{PSR} *satisying* $\forall x_2 \in \mathcal{X}_2,\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$*:*

$$F_2^{\mathrm{PSR}}(x_2, u_2) \supseteq \bigcup_{z \in \mathcal{Z}_{\eta,\eta',\epsilon,\epsilon'}(x_2)} \mathrm{argmin}_{x_2' \in \mathcal{X}_2} \ \|f(z, \kappa(z, x_2, u_2)) - x_2'\|_\infty, \quad (4.40)$$

ensures $\mathcal{S}_1 \preceq_R^{\mathrm{PSR}} \mathcal{S}_2^{\mathrm{PSR}}$*. Additionally, one can use the concretized controller* $\mathcal{C}_1^{\mathrm{PSR}}$ *(Definition 4.15) with*

$$h_1^{\mathrm{PSR}}(x_2, u_2, x_1) = \{\kappa(x_1, z, u_1')\}$$
$$h_2^{\mathrm{PSR}}(x_2, u_2, x_1, u_1) = \mathrm{argmin}_{x_2' \in \mathcal{X}_2} \ \|f(z, u_1') - x_2'\|_\infty,$$

where $z = \mathrm{argmin}_{z' \in \mathcal{Z}_{\eta,\eta',\epsilon,\epsilon'}(x_2)} \ \|x_1 - z'\|_\infty$ *and* $u_1' = \kappa(z, x_2, u_2)$*.*

Proof. Let $(x_1, x_2) \in R$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$. By applying Lemma 4.23 with parameters (η', ϵ') satisfying (4.37), we have $x_1 \in \mathcal{S}(z, \epsilon')$ with

$$z = \mathrm{argmin}_{z' \in \mathcal{Z}_{\eta,\eta',\epsilon,\epsilon'}(x_2)} \|x_1 - z'\|_\infty.$$

Next, applying Lemma 4.25 with parameters $(\eta, \epsilon', \epsilon)$ satisfying (4.38) and (4.39), we ensure that $x_1^+ = f(x_1, u_1) \in \mathcal{S}(x_2^+, \epsilon)$ with $u_1 = \kappa(x_1, z, u_1')$, $x_2^+ = \mathrm{argmin}_{x_2' \in \mathcal{X}_2} \|f(z, u_1') - x_2'\|_\infty$, and $u_1' = \kappa(z, x_2, u_2)$. Thus, $x_1^+ \in R^{-1}(x_2^+)$ and by (4.40), $x_2^+ \in F_2^{\mathrm{PSR}}(x_2, u_2)$, which establishes (4.1). Consequently, $u_1 \in I_R^{\mathrm{PSR}}(x_2, u_2, x_1)$ and h_2^{PSR} satisfies Definition 4.12. □

As discussed in Section 4.6.3, it is possible to predict the next abstract state x_2^+ using $h_2^{\mathrm{PSR}}(x_2, u_2, x_1, u_1)$ before applying the concrete input u_1 given by $h_1^{\mathrm{PSR}}(x_2, u_2, x_1)$. Unlike FAR, the PSR relation can be constructed without requiring the incremental stability ($\rho < 1$).

4.8 Summary and further research directions

Abstraction-based control techniques rely on two main components: the relation, which ensures the correspondence and transition conditions between concrete and abstract states, and the concretization procedure, which derives a valid controller for the original system from an abstract system controller.

In this chapter, we presented a systematic approach to characterize a simulation relation through a concretization procedure with a plug-and-

play control architecture. We showed that one can tailor the abstraction relation depending on the required properties of the concretization step and of the finally obtained concrete controller, independently of the abstract controller synthesis step (step 2 in Figure 1). We illustrated our approach using five key relations (ASR, PSR, FAR, MCR, FRR) that are relevant for designing smart abstractions. We discussed the advantages and drawbacks of their concretization procedures using a common example. Importantly, the results, such as those in Corollary 4.19, can be extended to any relation listed in Table 4.1. We demonstrated (Corollary 4.19) that if the concrete system is related to the abstraction via a system relation, then the abstract controller can be seamlessly connected to the original system through the corresponding interface, regardless of the specific specification to be enforced.

As we will discuss in the next chapter, the existence of a system relation between the original system and the abstraction is not only *sufficient* (Corollary 4.19) to ensure the simple closed-loop structure shown in Figure 4.1 (right), but also *necessary*.

In this chapter, we bridged the gap between the feedback refinement relation and the alternating simulation relation. We illustrated in Figure 4.4 how to construct the control architecture from the given relation and, conversely, how to derive the relation from the desired architecture. As future work, we aim to identify the specific classes of systems for which these system relations can be constructed. For instance, relations such as ASR, MCR, and FRR can be established without additional assumptions on the system (see Proposition 4.27), whereas constructing FAR requires certain stability assumptions (see Proposition 4.28). While this chapter provided a classification of system relations based on their interface or associated control architecture (Figure 4.5), this future work would allow to classify them according to the system classes for which they can be constructed.

5

Characterization of simulation relations

In the previous chapter, we introduced a framework that unifies various types of simulation relations, showing how ASR and FRR fit within this context. We demonstrated (Corollary 4.19) that when a concrete system is related to its abstraction by such system relation, the abstract controller can be seamlessly connected to the original system through the corresponding interface, regardless of the specific specification to be enforced. As a result, the existence of a system relation between the original system and its abstraction is a *sufficient* condition to achieve the corresponding concretization procedure in Figure 4.1 (right).

However, the authors in [RWR16] demonstrated that, in the case of the feedback refinement relation, this relation is not only sufficient but also *necessary* to achieve the simple closed-loop structure shown in Figure 5.1 (left). Inspired by this converse result for FRR, our aim in this chapter is to establish a similar converse result for Corollary 4.19 (Corollary 5.17). Specifically, we will show that if a given concretization procedure works for any abstract controller, then the original system must be related to its abstraction by the system relation associated with that concretization procedure. For example, while the existence of an alternating simulation relation is only *sufficient* to guarantee the concretization step (3.5), we prove (Corollary 5.17) that it is also *necessary* when considering a specific control

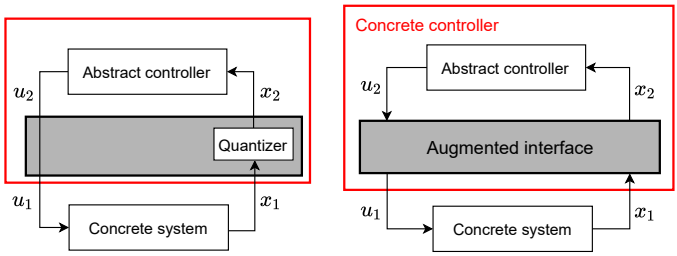


Fig. 5.1 Closed-loop system resulting from the abstraction and concretization approach. Left: Based on the feedback refinement relation and its simple concretization scheme (3.10). Right: Based on the augmented interface proposed in this chapter.

architecture for the original system (Remark 5.19).

To achieve this, we propose a universal system transformation, called the *augmented system*, that encapsulates the original system and its interface, and which is universal in the sense that it is always in feedback refinement relation with the abstract system. More specifically, given a system relation of type T, the overview of the approach is as follows and is illustrated in Figure 5.2. First, we transform the original system $\mathcal{S}_1$ into a new system $\widetilde{\mathcal{S}}_1^{\mathrm{T}}$, referred to as the augmented system, which is related to the abstract system $\mathcal{S}_2$ by a feedback refinement relation $\widetilde{R}$. Then, using the simple concretization procedure of the feedback refinement relation from [RWR16], any abstract controller $\mathcal{C}_2$ can be concretized into a controller $\widetilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \widetilde{R}$ for the augmented system $\widetilde{\mathcal{S}}_1^{\mathrm{T}}$ (Theorem 3.8). The concrete controller $\mathcal{C}_1^{\mathrm{T}}$ for the original system can then be derived from $\widetilde{\mathcal{C}}_1$.

This approach allows for the construction of concrete controllers that typically belong to a more general class than the piecewise constant controllers limited by the feedback refinement relation.

This chapter results from a collaboration with Antoine Girard, and this work has been partially published in [CGJ24].

5.1 Augmented interface

Similarly to Section 4.3, we introduce an *augmented interface* that connects the system $\mathcal{S}_1$ to any abstract controller $\mathcal{C}_2$ f.c. with the system $\mathcal{S}_2$, as illustrated in Figure 5.1 (right). Unlike the standard interface in Definition 4.9,

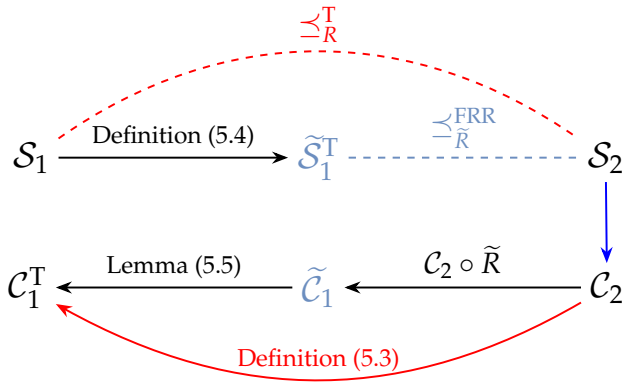


Fig. 5.2 Characterization of a system relation through its concretization procedure. If $\mathcal{S}_1 \preceq_R^{\mathrm{T}} \mathcal{S}_2$, then the augmented system satisfies $\widetilde{\mathcal{S}}_1^{\mathrm{T}} \preceq_{\widetilde{R}}^{\mathrm{FRR}} \mathcal{S}_2$. For any controller $\mathcal{C}_2$ f.c. with $\mathcal{S}_2$, the controllers $\widetilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \widetilde{R}$ and $\mathcal{C}_1^{\mathrm{T}}$ fulfill the condition in (3.5).

this augmented interface is defined by an extended tuple $(\mathcal{Z}_1, h_1, h_2, \widetilde{R})$, as formalized in the following definition.

Definition 5.1 (Augmented interface)**.** Given two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), an *augmented interface* for $\mathcal{S}_1$ and $\mathcal{S}_2$ is defined by the tuple $(\mathcal{Z}_1, h_1, h_2, \widetilde{R})$, where $\mathcal{Z}_1$ is a set, $h_1 : \mathcal{H}_1 \to 2^{\mathcal{U}_1}$ and $h_2 : \mathcal{H}_2 \to 2^{\mathcal{Z}_1}$ are set-valued functions, and $\widetilde{R} \subseteq (\mathcal{X}_1 \times \mathcal{Z}_1) \times \mathcal{X}_2$. The argument ν_1 (resp. ν_2) of h_1 (resp. h_2) is a subset of the set of variables $\nu = \{x_1, x_1^+, z_1, z_1^+, u_1, u_2\}$, where $x_1, x_1^+ \in \mathcal{X}_1$, $z_1, z_1^+ \in \mathcal{Z}_1$, $u_1 \in \mathcal{U}_1$, and $u_2 \in \mathcal{U}_2$[1]. The functions h_1 and h_2 ensure that the following holds. For all $((x_1, z_1), x_2) \in \widetilde{R}$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, it holds that any u_1, z_1^+, x_1^+ and x_2^+ resulting from the following sequence of equations

$$\begin{aligned} u_1 &\in h_1(\nu_1), \\ z_1^+ &\in h_2(\nu_2), \\ x_1^+ &\in F_1(x_1, u_1), \\ x_2^+ &\in \widetilde{R}((x_1^+, z_1^+)), \end{aligned} \tag{5.1}$$

satisfy $x_2^+ \in F_2(x_2, u_2)$ and $((x_1^+, z_1^+), x_2^+) \in \widetilde{R}$. △

As in (4.11), the order of the equations in (5.1) may vary based on the

[1]The domain $\mathcal{H}_1$ (resp. $\mathcal{H}_2$) of h_1 (resp. h_2) is defined as the Cartesian product of the domains of the corresponding variable ν_1 (resp. ν_2).

variables ν_1 and ν_2 of h_1 and h_2, respectively, as exemplified in (5.10).

Similarly to (4.12), to highlight the invariant preserved by the augmented interface, specifically that $((x_1^+, z_1^+), x_2^+) \in \widetilde{R}$ given $((x_1, z_1), x_2) \in \widetilde{R}$, we rewrite the condition (5.1) in Definition 5.1 as follows

$$\begin{aligned} \text{Pre: } & ((x_1, z_1), x_2) \in \widetilde{R},\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) \\ & u_1 \in h_1(\nu_1), \\ & z_1^+ \in h_2(\nu_2), \\ & x_1^+ \in F_1(x_1, u_1), \\ & x_2^+ \in \widetilde{R}((x_1^+, z_1^+)), \\ \text{Post: } & x_2^+ \in F_2(x_2, u_2),\ ((x_1^+, z_1^+), x_2^+) \in \widetilde{R} \end{aligned} \tag{5.2}$$

with the *pre-condition* and *post-condition* highlighted in blue.

The following lemma establishes how the augmented interface (Definition 5.1) generalizes the standard interface (Definition 4.9) by providing a specific implementation where condition (5.1) simplifies to (4.11).

Lemma 5.2. *Let two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1). If $\mathcal{Z}_1 = \mathcal{X}_2$ and*

$$\widetilde{R} = \{((x_1, x_2), x_2) \mid (x_1, x_2) \in R\}, \tag{5.3}$$

then the tuple (h_1, h_2, R) is an interface (Definition 4.9) for $\mathcal{S}_1$ and $\mathcal{S}_2$ if and only if $(\mathcal{Z}_1, h_1, h_2, \widetilde{R})$ is an augmented interface (Definition 5.1) for $\mathcal{S}_1$ and $\mathcal{S}_2$.

Proof. By (5.3), the pre-condition of (5.1) is equivalent to that of (4.11) since $((x_1, z_1), x_2) \in \widetilde{R} \Leftrightarrow (z_1 = x_2, (x_1, x_2) \in R)$. Then, by (4.11), any $u_1 \in h_1(\nu_1)$, $z_1^+ \in h_2(\nu_2)$, and $x_1^+ \in F_1(x_1, u_1)$ satisfy $z_1^+ \in F_2(x_2, u_2)$ and $(x_1^+, z_1^+) \in R$. Therefore, by (5.3), we have $\widetilde{R}((x_1^+, z_1^+)) = \{z_1^+\}$, implying $x_2^+ = z_1^+$, which establishes the post-condition of (5.1). □

We now define the controller $\mathcal{C}_1$, referred to as the *augmented concretized controller*, which results from connecting the controller $\mathcal{C}_2$ to the augmented interface.

Definition 5.3 (Augmented concretized controller)**.** Given two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), an augmented interface $(\mathcal{Z}_1, h_1, h_2, \widetilde{R})$ for $\mathcal{S}_1$ and $\mathcal{S}_2$, and a controller $\mathcal{C}_2$ f.c. with $\mathcal{S}_2$ defined as

$$\mathcal{C}_2 = (\mathcal{X}_{\mathcal{C}_2}, \mathcal{X}_2, \mathcal{V}_{\mathcal{C}_2}, \mathcal{U}_2, F_{\mathcal{C}_2}, H_{\mathcal{C}_2}), \tag{5.4}$$

an *augmented concretized controller* is any system $\mathcal{C}_1$ that satisfies the following closed-loop condition: $\boldsymbol{x}_1 \in \mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1)$ if and only if there exists $\boldsymbol{u}_1, \boldsymbol{z}_1, \boldsymbol{x}_2, \boldsymbol{u}_2, \boldsymbol{x}_{\mathcal{C}_2}, \boldsymbol{v}_{\mathcal{C}_2}$ such that

$$\begin{aligned}
&(u_2(k), v_{\mathcal{C}_2}(k)) \in H_{\mathcal{C}_2}(x_{\mathcal{C}_2}(k), x_2(k)) \\
&\quad x_{\mathcal{C}_2}(k+1) \in F_{\mathcal{C}_2}(x_{\mathcal{C}_2}(k), v_{\mathcal{C}_2}(k)) \\
&\qquad u_1(k) \in h_1(\nu_1(k)) \\
&\quad z_1(k+1) \in h_2(\nu_2(k)) \\
&\quad x_2(k+1) \in \widetilde{R}((x_1(k+1), z_1(k+1))) \\
&\quad x_1(k+1) \in F_1(x_1(k), u_1(k)).
\end{aligned} \tag{5.5}$$

△

Note that similarly to (5.1), the order of equations in (5.5) may vary depending on the variables ν_1 and ν_2 of h_1 and h_2.

We now introduce the notion of *augmented system* that encapsulate the concrete system $\mathcal{S}_1$ and the augmented interface $(\mathcal{Z}_1, h_1, h_2, \widetilde{R})$.

Definition 5.4 (Augmented system)**.** Given the simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and an augmented interface $(\mathcal{Z}_1, h_1, h_2, \widetilde{R})$ for $\mathcal{S}_1$ and $\mathcal{S}_2$, we define the associated *augmented system* $\widetilde{\mathcal{S}}_1$ as the simple system $\widetilde{\mathcal{S}}_1 = (\widetilde{\mathcal{X}}_1, \widetilde{\mathcal{U}}_1, \widetilde{F}_1)$, where $\widetilde{\mathcal{X}}_1 = \mathcal{X}_1 \times \mathcal{Z}_1$, $\widetilde{\mathcal{U}}_1 = \mathcal{U}_2$, and

$$\widetilde{F}_1((x_1, z_1), u_2) = \{(x_1^+, z_1^+) \mid u_1 \in h_1(\nu_1),\ z_1^+ \in h_2(\nu_2),\ x_1^+ \in F_1(x_1, u_1)\}. \tag{5.6}$$

△

We now present two lemmas that will be useful for the subsequent developments.

Lemma 5.5. *Let two simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), an augmented interface $(\mathcal{Z}_1, h_1, h_2, \widetilde{R})$ for $\mathcal{S}_1$ and $\mathcal{S}_2$, and the augmented system $\widetilde{\mathcal{S}}_1$ (Definition 5.4). For any controller $\mathcal{C}_2$ f.c. with $\mathcal{S}_2$, the concretized controller $\mathcal{C}_1$ (Definition 5.3), and $\widetilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \widetilde{R}$ satisfy*

$$\boldsymbol{x}_1 \in \mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1) \Leftrightarrow \widetilde{\boldsymbol{x}}_1 \in \mathcal{B}_\times(\widetilde{\mathcal{C}}_1 \times \widetilde{\mathcal{S}}_1),$$

where $\boldsymbol{x}_1 = \pi_{\mathcal{X}_1}(\widetilde{\boldsymbol{x}}_1)$ (1.1).

Proof. According to Definition 2.7, the controller $\widetilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \widetilde{R}$ is defined as

$$\widetilde{\mathcal{C}}_1 = (\mathcal{X}_{\mathcal{C}_2}, \mathcal{X}_1 \times \mathcal{Z}_1, \mathcal{V}_{\mathcal{C}_2}, \mathcal{U}_2, F_{\mathcal{C}_2}, H_{\widetilde{\mathcal{C}}_1}), \tag{5.7}$$

where $H_{\widetilde{\mathcal{C}}_1}(x_{\mathcal{C}_2}, (x_1, z_1)) = H_{\mathcal{C}_2}(x_{\mathcal{C}_2}, \widetilde{R}(x_1, z_1))$, since $\widetilde{R}$ is a static input quantizer (see (2.13)). Consequently, $\widetilde{\boldsymbol{x}}_1 \in \mathcal{B}_\times(\widetilde{\mathcal{C}}_1 \times \widetilde{\mathcal{S}}_1)$ if and only if there exist sequences $\boldsymbol{u}_1, \boldsymbol{x}_2, \boldsymbol{u}_2, \boldsymbol{x}_{\mathcal{C}_2}, \boldsymbol{v}_{\mathcal{C}_2}$ satisfying (5.5) with $\boldsymbol{x}_1 = \pi_{\mathcal{X}_1}(\widetilde{\boldsymbol{x}}_1)$ and $\boldsymbol{z}_1 = \pi_{\mathcal{Z}_1}(\widetilde{\boldsymbol{x}}_1)$. This establishes that $\boldsymbol{x}_1 \in \mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1)$. □

Lemma 5.6. *Let* $\widetilde{R} \subseteq (\mathcal{X}_1 \times \mathcal{Z}_1) \times \mathcal{X}_2$, *and*

$$R = \{(x_1, x_2) \in \mathcal{X}_1 \times \mathcal{X}_2 \mid \exists z_1 \in \mathcal{Z}_1 \text{ such that } ((x_1, z_1), x_2) \in \widetilde{R}\}. \tag{5.8}$$

If $\widetilde{R}((x_1, z_1)) \neq \varnothing$, *then* $\widetilde{R}((x_1, z_1)) \subseteq R(x_1)$.

Proof. Trivial, by definition $R(x_1) = \bigcup_{(x_1,z_1)|\widetilde{R}((x_1,z_1))\neq\varnothing} \widetilde{R}((x_1, z_1))$. □

Theorem 5.7. *Let two simple systems* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *in* (3.1), *and the augmented system* $\widetilde{\mathcal{S}}_1$ *(Definition 5.4) associated with the tuple* $(\mathcal{Z}_1, h_1, h_2, \widetilde{R})$. *The following propositions are equivalent.*

(1) $(\mathcal{Z}_1, h_1, h_2, \widetilde{R})$ *is an augmented interface for* $\mathcal{S}_1$ *and* $\mathcal{S}_2$;

(2) $\widetilde{\mathcal{S}}_1 \preceq^{\mathrm{FRR}}_{\widetilde{R}} \mathcal{S}_2$;

(3) for any controller $\mathcal{C}_2$ *f.c. with* $\mathcal{S}_2$, *it holds that* $\mathcal{C}_2$ *is f.c. with* $\widetilde{R} \circ \widetilde{\mathcal{S}}_1$ *and* $\widetilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \widetilde{R}$ *is f.c. with* $\widetilde{\mathcal{S}}_1$, *and* $\mathcal{B}_\times(\mathcal{C}_2 \times (\widetilde{R} \circ \widetilde{\mathcal{S}}_1)) \subseteq \mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2)$.

Additionally, for any controllers $\mathcal{C}_2$ *f.c. with* $\mathcal{S}_2$, *the augmented concretized controller* $\mathcal{C}_1$ *(Definition 5.3) satisfies* $\mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2))$ *with* R *in* (5.8).

Proof.
We first establish the equivalences.

(1) ⇔ (2) The tuple $(\mathcal{Z}_1, h_1, h_2, \widetilde{R})$ is an augmented interface if and only if it satisfies (5.1). This is equivalent to requiring that for any $((x_1, z_1), x_2) \in \widetilde{R}$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, (i) $\widetilde{F}_1((x_1, z_1), u_2) \neq \varnothing$, and (ii) for all $(x_1^+, z_1^+) \in \widetilde{F}_1((x_1, z_1), u_2)$, $\widetilde{R}((x_1^+, z_1^+)) \neq \varnothing$ and $\widetilde{R}((x_1^+, z_1^+)) \subseteq F_2(x_2, u_2)$. This, in turn, is equivalent to conditions (i) and (ii) of Definition 3.5, meaning that $\widetilde{\mathcal{S}}_1 \preceq^{\mathrm{FRR}}_{\widetilde{R}} \mathcal{S}_2$.

(2) ⇔ (3) Directly follows from Theorem 3.8.

In addition, Theorem 3.8 implies

$$\mathcal{B}_\times((\mathcal{C}_2 \circ \widetilde{R}) \times \widetilde{\mathcal{S}}_1) \subseteq \widetilde{R}^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2)). \tag{5.9}$$

Let $\boldsymbol{x}_1 \in \mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1)$ be defined on $[0; N[$, where $N \in \mathbb{N} \cup \{\infty\}$. Then, by Lemma 5.5, there exists a map $\boldsymbol{z}_1$ such that $\widetilde{\boldsymbol{x}}_1 \in \mathcal{B}_\times((\mathcal{C}_2 \circ \widetilde{R}) \times \widetilde{\mathcal{S}}_1)$ with $\widetilde{x}_1(k) = (x_1(k), z_1(k))$ for all $k \in [0; N[$. Then, by (5.9), there exists a map $\boldsymbol{x}_2 \in \mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2)$ such that $x_2(k) \in \widetilde{R}((x_1(k), z_1(k)))$ for all $k \in [0; N[$. By Lemma 5.6, since R satisfies (5.8), we can conclude that $x_2(k) \in R(x_1(k))$ for all $k \in [0; N[$, i.e., $\boldsymbol{x}_2 \in R(\boldsymbol{x}_1)$. As a result, $\mathcal{B}_\times(\mathcal{C}_1 \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2))$. □

5.2 Augmented interface for system relations

We now define the arguments ν_1 and ν_2 for the set-valued functions h_1^{T} and h_2^{T} in the augmented interface for some system relations, similar to how they were specified for the standard interface in Table 4.1.

Definition 5.8. Let $\mathrm{T} \in \{\mathrm{ASR}, \mathrm{PSR}, \mathrm{FAR}, \mathrm{MCR}, \mathrm{FRR}\}$. We define the arguments of the set-valued functions h_1^{T} and h_2^{T} as follows

$$h_1^{\mathrm{ASR}}(z_1, u_2, x_1), h_1^{\mathrm{PSR}}(z_1, u_2, x_1), h_1^{\mathrm{FAR}}(z_1, u_2, x_1, z_1^+), h_1^{\mathrm{MCR}}(z_1, u_2, x_1), h_1^{\mathrm{FRR}}(u_2),$$
$$h_2^{\mathrm{ASR}}(z_1, u_2, x_1^+), h_2^{\mathrm{PSR}}(z_1, u_2, x_1, u_1), h_2^{\mathrm{FAR}}(z_1, u_2), h_2^{\mathrm{FRR}}(x_1^+), h_2^{\mathrm{MCR}}(x_1^+).$$

△

Note that these functions are derived by substituting the variables x_2 and x_2^+ in the standard interface from Table 4.1 with z_1 and z_1^+, respectively. As an example, the conditions (5.1) for ASR and PSR are given by

$$\begin{aligned}
&\text{Pre: } ((x_1, z_1), x_2) \in \widetilde{R},\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2) && \text{Pre: } ((x_1, z_1), x_2) \in \widetilde{R},\ u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)\\
&\quad u_1 \in h_1^{\mathrm{ASR}}(z_1, u_2, x_1), && \quad u_1 \in h_1^{\mathrm{PSR}}(z_1, u_2, x_1),\\
&\quad x_1^+ \in F_1(x_1, u_1), && \quad z_1^+ \in h_2^{\mathrm{PSR}}(z_1, u_2, x_1, u_1),\\
&\quad z_1^+ \in h_2^{\mathrm{ASR}}(z_1, u_2, x_1^+), && \quad x_1^+ \in F_1(x_1, u_1),\\
&\quad x_2^+ \in \widetilde{R}((x_1^+, z_1^+)), && \quad x_2^+ \in \widetilde{R}((x_1^+, z_1^+)),\\
&\text{Post: } x_2^+ \in F_2(x_2, u_2),\ ((x_1^+, z_1^+), x_2^+) \in \widetilde{R} && \text{Post: } x_2^+ \in F_2(x_2, u_2),\ ((x_1^+, z_1^+), x_2^+) \in \widetilde{R}.
\end{aligned} \tag{5.10}$$

Definition 5.9. A relation $\widetilde{R} \subseteq (\mathcal{X}_1 \times \mathcal{Z}_1) \times \mathcal{X}_2$ satisfies the *common quantization assumption* if $\forall x_1, x_1' \in \mathcal{X}_1,\ z_1 \in \mathcal{Z}_1$ such that $\widetilde{R}((x_1, z_1)) \neq \varnothing$ and

$\widetilde{R}((x_1', z_1)) \neq \varnothing$ implies $\widetilde{R}(x_1, z_1) \cap \widetilde{R}((x_1', z_1)) \neq \varnothing$. △

The following theorem establishes that the existence of a relation of type T between $\mathcal{S}_1$ and $\mathcal{S}_2$ is equivalent to the existence of an augmented interface for $\mathcal{S}_1$ and $\mathcal{S}_2$ satisfying Definition 5.8.

Theorem 5.10. *Given* $\mathrm{T} \in \{\mathrm{ASR}, \mathrm{PSR}, \mathrm{FAR}, \mathrm{MCR}, \mathrm{FRR}\}$ *and the simple systems* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *in* (3.1)*, the following propositions hold.*

(1) If there exists $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ *such that* $\mathcal{S}_1 \preceq_R^{\mathrm{T}} \mathcal{S}_2$*, then* $(\mathcal{Z}_1, h_1^{\mathrm{T}}, h_2^{\mathrm{T}}, \widetilde{R})$ *where* $\mathcal{Z}_1 = \mathcal{X}_2$*,* h_1^{T} *and* h_2^{T} *are given in Definition 4.12, and* $\widetilde{R}$ *in* (5.3)*, is an augmented interface for* $\mathcal{S}_1$ *and* $\mathcal{S}_2$*, i.e.,* $\widetilde{\mathcal{S}}_1^{\mathrm{T}} \preceq_{\widetilde{R}}^{\mathrm{FRR}} \mathcal{S}_2$*.*

(2) If there exists an augmented interface $(\mathcal{Z}_1, h_1^{\mathrm{T}}, h_2^{\mathrm{T}}, \widetilde{R})$*, i.e.,* $\widetilde{\mathcal{S}}_1^{\mathrm{T}} \preceq_{\widetilde{R}}^{\mathrm{FRR}} \mathcal{S}_2$*, with* h_1^{T} *and* h_2^{T} *satisfying Definition 5.8 and* $\widetilde{R}$ *satisfying Definition 5.9, then* $\mathcal{S}_1 \preceq_R^{\mathrm{T}} \mathcal{S}_2$ *with* R *in* (5.8)*.*

Proof.

(1) By Proposition 4.13, the functions h_1^{T} and h_2^{T} in Definition 4.12 ensure that if $\mathcal{S}_1 \preceq_R^{\mathrm{T}} \mathcal{S}_2$, then $(h_1^{\mathrm{T}}, h_2^{\mathrm{T}}, R)$ forms an interface (Definition 4.9) for $\mathcal{S}_1$ and $\mathcal{S}_2$. Therefore, by Lemma 5.2, $(\mathcal{Z}_1, h_1^{\mathrm{T}}, h_2^{\mathrm{T}}, \widetilde{R})$, with $\mathcal{Z}_1 = \mathcal{X}_2$ and $\widetilde{R}$ defined in (5.3), is an augmented interface (Definition 5.1) for $\mathcal{S}_1$ and $\mathcal{S}_2$. Finally, Theorem 5.7 establishes that $\widetilde{\mathcal{S}}_1^{\mathrm{T}} \preceq_{\widetilde{R}}^{\mathrm{FRR}} \mathcal{S}_2$.

(2) Given $\widetilde{\mathcal{S}}_1^{\mathrm{T}} \preceq_{\widetilde{R}}^{\mathrm{FRR}} \mathcal{S}_2$ (Definition 3.5), for any $((x_1, z_1), x_2) \in \widetilde{R}$:

(*i*) $\mathcal{U}_{\mathcal{S}_2}(x_2) \subseteq \mathcal{U}_{\widetilde{\mathcal{S}}_1^{\mathrm{T}}}((x_1, z_1))$;

(*ii*) $\forall u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)\ \forall (x_1^+, z_1^+) \in \widetilde{F}_1((x_1, z_1), u_2) :\ \widetilde{R}((x_1^+, z_1^+)) \neq \varnothing$ and $\widetilde{R}((x_1^+, z_1^+)) \subseteq F_2(x_2, u_2)$.

Let $(x_1, x_2) \in R$ and $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$. This implies the existence of z_1 such that $((x_1, z_1), x_2) \in \widetilde{R}$. By condition (*i*), we have $u_2 \in \mathcal{U}_{\widetilde{\mathcal{S}}_1}((x_1, z_1))$, meaning $\widetilde{F}_1((x_1, z_1), u_2) \neq \varnothing$. Thus, by Definition 5.4, for any $u_1 \in h_1^{\mathrm{T}}(v_1)$, $u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)$, and for all $x_1^+ \in F_1(x_1, u_1)$, $z_1^+ \in h_2^{\mathrm{T}}(v_2)$, it holds that $(x_1^+, z_1^+) \in \widetilde{F}_1((x_1, z_1), u_2)$. In addition, by (*ii*), $\widetilde{R}((x_1^+, z_1^+)) \neq \varnothing$ and $\widetilde{R}((x_1^+, z_1^+)) \subseteq F_2(x_2, u_2)$.

The rest of the proof is specific to each relation type:

- ASR: According to Lemma 5.6, $\widetilde{R}((x_1^+, z_1^+)) \subseteq R(x_1^+)$. Therefore,

$$F_2(x_2, u_2) \cap R(x_1^+) \supseteq \widetilde{R}((x_1^+, z_1^+)) \neq \varnothing,$$

which is equivalent to (3.8).

- PSR: Let $u_1 \in h_1^{\mathrm{PSR}}(z_1, u_2, x_1)$ and $z_1^+ \in h_2^{\mathrm{PSR}}(z_1, u_2, u_1, x_1)$. Since $\widetilde{R}$ satisfies Definition 5.9, it holds that

$$\mathcal{A} \coloneqq \left(\bigcap_{x_1^+ \in F_1(x_1, u_1)} \widetilde{R}((x_1^+, z_1^+)) \right) \neq \varnothing,$$

as $\widetilde{R}((x_1^+, z_1^+)) \neq \varnothing$ by (ii). Let $x_2^+ \in \mathcal{A}$. Firstly, $x_2^+ \in F_2(x_2, u_2)$ since $\widetilde{R}((x_1^+, z_1^+)) \subseteq F_2(x_2, u_2)$ by (ii). Secondly, according to Lemma 5.6, $\widetilde{R}((x_1^+, z_1^+)) \subseteq R(x_1^+)$, therefore $x_2^+ \in R(x_1^+)$, which is equivalent to (4.1).

- FAR: Let $z_1^+ \in h_2^{\mathrm{FAR}}(z_1, u_2)$. Since $\widetilde{R}$ satisfies Definition 5.9, it holds that

$$\mathcal{A} \coloneqq \left(\bigcap_{u_1 \in h_1^{\mathrm{FAR}}(z_1, u_2, x_1, z_1^+)} \bigcap_{x_1^+ \in F_1(x_1, u_1)} \widetilde{R}((x_1^+, z_1^+)) \right) \neq \varnothing,$$

as $\widetilde{R}((x_1^+, z_1^+)) \neq \varnothing$ by (ii). Let $x_2^+ \in \mathcal{A}$. Firstly, $x_2^+ \in F_2(x_2, u_2)$ since $\widetilde{R}((x_1^+, z_1^+)) \subseteq F_2(x_2, u_2)$ by (ii). Secondly, according to Lemma 5.6, $\widetilde{R}((x_1^+, z_1^+)) \subseteq R(x_1^+)$, therefore $x_2^+ \in R(x_1^+)$, which is equivalent to (4.3).

- MCR: By definition of R in (5.8), we have

$$R(x_1^+) = \bigcup_{z_1^+ \in h_2^{\mathrm{MCR}}(x_1^+)} \widetilde{R}((x_1^+, z_1^+)),$$

which establishes that $R(x_1^+) \neq \varnothing$ and $R(x_1^+) \subseteq F_2(x_2, u_2)$, which is equivalent to (4.4).

□

Remark 5.11. Note that the assumption that $\widetilde{R}$ satisfies Definition 5.9 in Theorem 5.10 (2) is only necessary for $\mathrm{T} \in \{\mathrm{PSR}, \mathrm{FAR}\}$. Additionally, it can be observed that $\widetilde{R}$ in (5.3) satisfies Definition 5.9. Indeed, given x_1, x_1' such that $\widetilde{R}((x_1, x_2)) \neq \varnothing$ and $\widetilde{R}((x_1', x_2)) \neq \varnothing$, we have $\widetilde{R}((x_1, x_2)) \cap \widetilde{R}((x_1', x_2)) = \{x_2\} \neq \varnothing$. △

Theorem 5.10 demonstrates that the existence of a system relation between a system and its abstraction can be equivalently expressed as the existence of a feedback refinement relation between its associated augmented system and the abstraction, as illustrated in Figure 5.3.

By Lemma 5.2, Theorem 5.10 (1) is equivalent to Proposition 4.13.

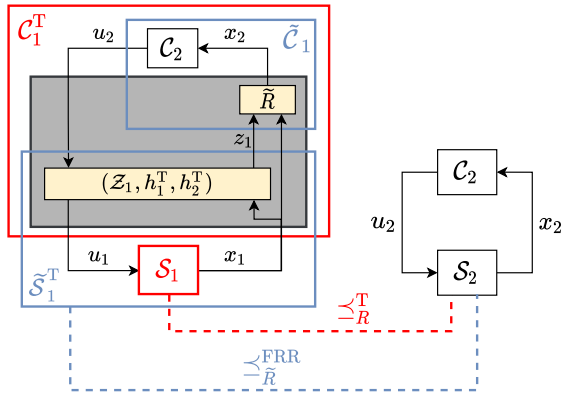


Fig. 5.3 Concretization of an abstract controller $\mathcal{C}_2$ into a concrete controller $\mathcal{C}_1^{\mathrm{T}}$ for a relation of type T ($\mathcal{S}_1 \preceq_R^{\mathrm{T}} \mathcal{S}_2$) using the augmented interface $(\mathcal{Z}_1, h_1^{\mathrm{T}}, h_2^{\mathrm{T}}, \widetilde{R})$. The figure illustrates the augmented system $\widetilde{\mathcal{S}}_1^{\mathrm{T}}$, where $\widetilde{\mathcal{S}}_1 \preceq_{\widetilde{R}}^{\mathrm{FRR}} \mathcal{S}_2$, along with the controller $\tilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \widetilde{R}$ for $\widetilde{\mathcal{S}}_1^{\mathrm{T}}$.

5.3 Concretization procedure

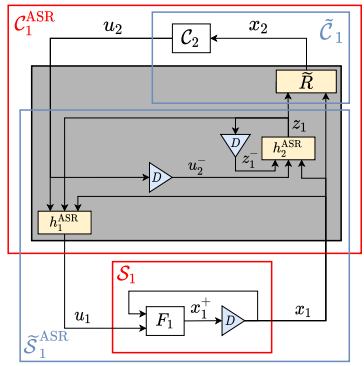
Similarly to Section 4.5, once the interfaces for each system relation are identified their concretization procedure (3.5) directly follows from constructing their respective augmented concretized controller (Definition 5.3).

Given the system relation $\mathrm{T} \in \{\mathrm{ASR}, \mathrm{PSR}, \mathrm{FAR}, \mathrm{MCR}, \mathrm{FRR}\}$, we now define in our control formalism (Chapter 2) the augmented concretized controller (Definition 5.3)

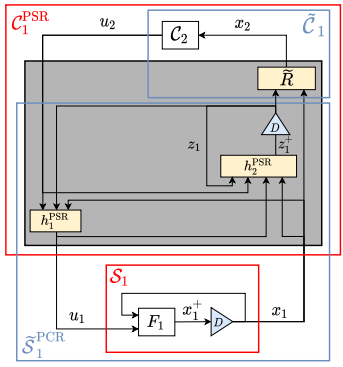
$$\mathcal{C}_1^{\mathrm{T}} = (\mathcal{X}_{\mathcal{C}_1^{\mathrm{T}}}, \mathcal{X}_1, \mathcal{V}_{\mathcal{C}_1^{\mathrm{T}}}, \mathcal{U}_1, F_{\mathcal{C}_1^{\mathrm{T}}}, H_{\mathcal{C}_1^{\mathrm{T}}}), \tag{5.11}$$

resulting from connecting $\mathcal{C}_2$ (4.13) to the augmented interface $(\mathcal{Z}_1, h_1^{\mathrm{T}}, h_2^{\mathrm{T}}, \widetilde{R})$ given in Definition 5.8, as illustrated in Figure 5.4.

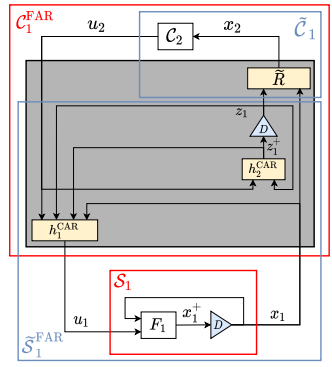
Definition 5.12. Given $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and $\mathcal{C}_2$ in (5.4), the concrete controller $\mathcal{C}_1^{\mathrm{ASR}}$ in (5.11) is defined such that $\mathcal{X}_{\mathcal{C}_1^{\mathrm{ASR}}} = \mathcal{X}_{\tilde{\mathcal{C}}_1} \times \mathcal{Z}_1 \times \mathcal{U}_2$, $\mathcal{V}_{\mathcal{C}_1^{\mathrm{ASR}}} =$



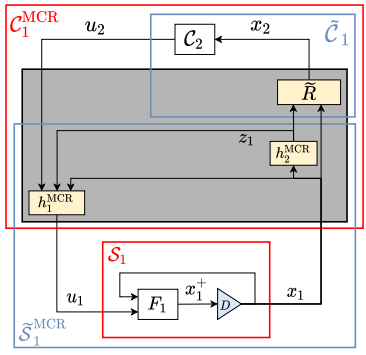
(a) ASR.



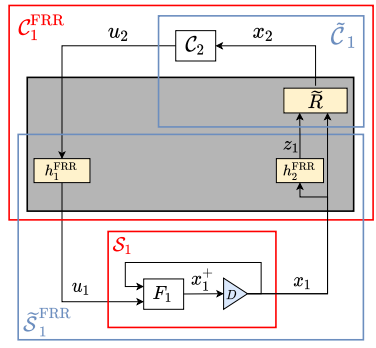
(b) PSR.



(c) FAR.



(d) MCR.



(e) FRR.

Fig. 5.4 Feedback composition of the concrete system $\mathcal{S}_1$ with the augmented concretized controller $\mathcal{C}_1^{\mathrm{T}}$, derived from the abstract controller $\mathcal{C}_2$ and the augmented interface $(\mathcal{Z}_1, h_1^{\mathrm{T}}, h_2^{\mathrm{T}}, \tilde{R})$, for some system relation T in Table 4.1. The associated augmented system $\tilde{\mathcal{S}}_1^{\mathrm{T}}$ and its controller $\tilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \tilde{R}$ are also depicted.

$\mathcal{V}_{\widetilde{\mathcal{C}}_1} \times \mathcal{Z}_1 \times \mathcal{U}_2$, and

$$F_{\mathcal{C}_1^{\mathrm{ASR}}}((x_{\widetilde{\mathcal{C}}_1}, z_1^-, u_2^-), (v_{\widetilde{\mathcal{C}}_1}, z_1, u_2)) = \{(x_{\widetilde{\mathcal{C}}_1}^+, z_1, u_2) \mid x_{\widetilde{\mathcal{C}}_1}^+ \in F_{\widetilde{\mathcal{C}}_1}(x_{\widetilde{\mathcal{C}}_1}, v_{\widetilde{\mathcal{C}}_1})\},$$
$$H_{\mathcal{C}_1^{\mathrm{ASR}}}((x_{\widetilde{\mathcal{C}}_1}, z_1^-, u_2^-), x_1) = \{(u_1, (v_{\widetilde{\mathcal{C}}_1}, z_1, u_2)) \mid z_1 \in h_2^{\mathrm{ASR}}(z_1^-, u_2^-, x_1),$$
$$(u_2, v_{\widetilde{\mathcal{C}}_1}) \in H_{\widetilde{\mathcal{C}}_1}(x_{\widetilde{\mathcal{C}}_1}, (x_1, z_1)), u_1 \in h_1^{\mathrm{ASR}}(z_1, u_2, x_1)\},$$

with $\widetilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \widetilde{R}$. △

Definition 5.13. Given $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and $\mathcal{C}_2$ in (5.4), the concrete controller $\mathcal{C}_1^{\mathrm{PSR}}$ in (5.11) is defined such that $\mathcal{X}_{\mathcal{C}_1^{\mathrm{PSR}}} = \mathcal{X}_{\widetilde{\mathcal{C}}_1} \times \mathcal{Z}_1$, $\mathcal{V}_{\mathcal{C}_1^{\mathrm{PSR}}} = \mathcal{V}_{\widetilde{\mathcal{C}}_1} \times \mathcal{Z}_1$, and

$$F_{\mathcal{C}_1^{\mathrm{PSR}}}((x_{\widetilde{\mathcal{C}}_1}, z_1), (v_{\widetilde{\mathcal{C}}_1}, z_1^+)) = \{(x_{\widetilde{\mathcal{C}}_1}^+, z_1^+) \mid x_{\widetilde{\mathcal{C}}_1}^+ \in F_{\widetilde{\mathcal{C}}_1}(x_{\widetilde{\mathcal{C}}_1}, v_{\widetilde{\mathcal{C}}_1})\},$$
$$H_{\mathcal{C}_1^{\mathrm{PSR}}}((x_{\widetilde{\mathcal{C}}_1}, z_1), x_1) = \{(u_1, (v_{\widetilde{\mathcal{C}}_1}, z_1^+)) \mid (u_2, v_{\widetilde{\mathcal{C}}_1}) \in H_{\widetilde{\mathcal{C}}_1}(x_{\widetilde{\mathcal{C}}_1}, (x_1, z_1)),$$
$$u_1 \in h_1^{\mathrm{PSR}}(z_1, u_2, x_1), z_1^+ \in h_2^{\mathrm{PSR}}(z_1, u_2, x_1, u_1)\},$$

with $\widetilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \widetilde{R}$. △

Definition 5.14. Given $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and $\mathcal{C}_2$ in (5.4), the concrete controller $\mathcal{C}_1^{\mathrm{FAR}}$ in (5.11) is defined such that $\mathcal{X}_{\mathcal{C}_1^{\mathrm{FAR}}} = \mathcal{X}_{\widetilde{\mathcal{C}}_1} \times \mathcal{Z}_1$, $\mathcal{V}_{\mathcal{C}_1^{\mathrm{FAR}}} = \mathcal{V}_{\widetilde{\mathcal{C}}_1} \times \mathcal{Z}_1$, and

$$F_{\mathcal{C}_1^{\mathrm{FAR}}}((x_{\widetilde{\mathcal{C}}_1}, z_1), (v_{\widetilde{\mathcal{C}}_1}, z_1^+)) = \{(x_{\widetilde{\mathcal{C}}_1}^+, z_1^+) \mid x_{\widetilde{\mathcal{C}}_1}^+ \in F_{\widetilde{\mathcal{C}}_1}(x_{\widetilde{\mathcal{C}}_1}, v_{\widetilde{\mathcal{C}}_1})\},$$
$$H_{\mathcal{C}_1^{\mathrm{FAR}}}((x_{\widetilde{\mathcal{C}}_1}, z_1), x_1) = \{(u_1, (v_{\widetilde{\mathcal{C}}_1}, z_1^+)) \mid (u_2, v_{\widetilde{\mathcal{C}}_1}) \in H_{\widetilde{\mathcal{C}}_1}(x_{\widetilde{\mathcal{C}}_1}, (x_1, z_1)),$$
$$z_1^+ \in h_2^{\mathrm{FAR}}(z_1, u_2), u_1 \in h_1^{\mathrm{FAR}}(z_1, u_2, x_1, z_1^+)\},$$

with $\widetilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \widetilde{R}$. △

Definition 5.15. Given $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and $\mathcal{C}_2$ in (5.4), the concrete controller $\mathcal{C}_1^{\mathrm{MCR}}$ in (5.11) is defined such that $\mathcal{X}_{\mathcal{C}_1^{\mathrm{MCR}}} = \mathcal{X}_{\widetilde{\mathcal{C}}_1}$, $\mathcal{V}_{\mathcal{C}_1^{\mathrm{MCR}}} = \mathcal{V}_{\widetilde{\mathcal{C}}_1}$, and

$$F_{\mathcal{C}_1^{\mathrm{MCR}}}(x_{\widetilde{\mathcal{C}}_1}, v_{\widetilde{\mathcal{C}}_1}) = F_{\widetilde{\mathcal{C}}_1}(x_{\widetilde{\mathcal{C}}_1}, v_{\widetilde{\mathcal{C}}_1}),$$
$$H_{\mathcal{C}_1^{\mathrm{MCR}}}(x_{\widetilde{\mathcal{C}}_1}, x_1) = \{(u_1, v_{\widetilde{\mathcal{C}}_1}) \mid z_1 \in h_2^{\mathrm{MCR}}(x_1),$$
$$(u_2, v_{\widetilde{\mathcal{C}}_1}) \in H_{\widetilde{\mathcal{C}}_1}(x_{\widetilde{\mathcal{C}}_1}, (x_1, z_1)), u_1 \in h_1^{\mathrm{MCR}}(z_1, u_2, x_1)\},$$

with $\widetilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \widetilde{R}$. △

Definition 5.16. Given $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and $\mathcal{C}_2$ in (5.4), the concrete con-

troller $\mathcal{C}_1^{\mathrm{FRR}}$ in (5.11) is defined such that $\mathcal{X}_{\mathcal{C}_1^{\mathrm{FRR}}} = \mathcal{X}_{\widetilde{\mathcal{C}}_1}$, $\mathcal{V}_{\mathcal{C}_1^{\mathrm{FRR}}} = \mathcal{V}_{\widetilde{\mathcal{C}}_1}$, and

$$\begin{aligned} F_{\mathcal{C}_1^{\mathrm{FRR}}}(x_{\widetilde{\mathcal{C}}_1}, v_{\widetilde{\mathcal{C}}_1}) &= F_{\widetilde{\mathcal{C}}_1}(x_{\widetilde{\mathcal{C}}_1}, v_{\widetilde{\mathcal{C}}_1}), \\ H_{\mathcal{C}_1^{\mathrm{FRR}}}(x_{\widetilde{\mathcal{C}}_1}, x_1) &= \{(u_1, v_{\widetilde{\mathcal{C}}_1}) \mid z_1 \in h_2^{\mathrm{FRR}}(x_1), \\ &\qquad (u_2, v_{\widetilde{\mathcal{C}}_1}) \in H_{\widetilde{\mathcal{C}}_1}(x_{\widetilde{\mathcal{C}}_1}, (x_1, z_1)), u_1 \in h_1^{\mathrm{FRR}}(u_2)\}, \end{aligned}$$

with $\widetilde{\mathcal{C}}_1 = \mathcal{C}_2 \circ \widetilde{R}$. △

By Lemma 5.2, for $\mathcal{Z}_1 = \mathcal{X}_2$, h_1^{T} and h_2^{T} in Definition 4.12, and $\widetilde{R}$ in (5.3), the concrete controller $\mathcal{C}_1^{\mathrm{T}}$ in (5.11) simplifies to (4.23), and Figure 5.4 reduces to Figure 4.6. Consequently, the properties discussed in Section 4.6 also apply to these controllers.

The following corollary of Theorem 5.7 demonstrates that the existence of a relation of type T between the original system and the abstraction is not only sufficient to ensure the simple closed-loop structure shown in Figure 5.1 (right), but also necessary.

Corollary 5.17. *Let* $\mathrm{T} \in \{\mathrm{ASR}, \mathrm{PSR}, \mathrm{FAR}, \mathrm{MCR}, \mathrm{FRR}\}$ *and two simple systems* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *in* (3.1)*. The following propositions are equivalent.*

(1) There exists $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ *such that* $\mathcal{S}_1 \preceq_R^{\mathrm{T}} \mathcal{S}_2$*;*

(2) There exists an augmented interface $(\mathcal{Z}_1, h_1^{\mathrm{T}}, h_2^{\mathrm{T}}, \widetilde{R})$ *for* $\mathcal{S}_1$ *and* $\mathcal{S}_2$*, where* h_1^{T} *and* h_2^{T} *satisfy Definition 5.8 and* $\widetilde{R}$ *satisfies Definition 5.9.*

Additionally, for any controllers $\mathcal{C}_2$ *f.c. with* $\mathcal{S}_2$*, the augmented concretized controller* $\mathcal{C}_1^{\mathrm{T}}$ (5.11) *satisfies* $\mathcal{B}_\times(\mathcal{C}_1^{\mathrm{T}} \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_\times(\mathcal{C}_2 \times \mathcal{S}_2))$ *with* R *in* (5.8)*.*

Proof. It directly follows from Theorem 5.10 and Theorem 5.7. □

While the forward implication (1) ⇒ (2) in Corollary 5.17 was already established by Corollary 4.19 with the implementation $\mathcal{Z}_1 = \mathcal{X}_2$, h_1^{T} and h_2^{T} in Definition 4.12, and $\widetilde{R}$ in (5.3) (see Lemma 5.2), Corollary 5.17 provides the converse result.

Remark 5.18. For T = FRR, Corollary 5.17 is equivalent to Theorem 3.8. Thus, Corollary 5.17 generalizes Theorem 3.8 to other system relations than the feedback refinement relation.

Remark 5.19. While the existence of an ASR is only a *sufficient* condition (Theorem 3.4) to ensure the concretization procedure (3.5), it also *necessary* (Corollary 5.17) when considering the specific control architecture in Figure 5.4a.

5.4 Summary and further research directions

In this chapter, we proposed a systematic approach to precisely characterize simulation relations through a concretization procedure that features a plug-and-play control architecture. To achieve this, we introduced the *augmented system*, a system transformation that is always in feedback refinement relation with the abstract system, enabling us to leverage feedback refinement relation results to other types of system relations. This enabled us to generalize prior results and establish a unified framework for systematically classifying and characterizing system relations. As in the previous chapter, we applied this approach to five key relations (ASR, PSR, FAR, MCR, FRR).

In relation to this chapter, we identify two main research directions.

First, our approach, as shown in Figure 5.2, constructs an augmented system related to the abstraction through a feedback refinement relation, enabling a plug-and-play concretization procedure that supports more general controller classes beyond piecewise constant controllers. In future work, we aim to expand this method by exploring new system transformations that connect augmented systems to abstractions via different types of relations. While this may sacrifice the plug-and-play feature, it could allow the design of even broader classes of controllers.

Second, we aim to extend our current framework, which focuses on state-feedback control of simple systems, to *output-feedback* control. Various types of relations for systems with output map are well-established in the literature [MOS20, APGCZ20, SR14, CPMJ22]. We plan to classify and characterize these relations in terms of their associated plug-and-play control architecture.

PART III
Algorithms

6

Dynamic programming on abstractions

In this chapter, we establish the connection between alternating simulations and the Bellman operator introduced in Section 2.5. Specifically, we demonstrate how to translate suboptimal (Definition 2.14) and superoptimal (Definition 2.15) value functions from the abstract system to the concrete system. This includes showing how these functions can be efficiently computed for the original system based on their counterparts in the abstract system. We will demonstrate in Chapter 7 and Chapter 8 how these dynamic programming surrogate functions, though suboptimal, can be effectively used to construct smart abstractions.

The results presented in this chapter are based on preliminary work published in [CLEJ21].

6.1 Suboptimal value function: from concrete to abstract

The following theorem demonstrates how to derive a suboptimal value function (Definition 2.14) for the abstract system from a suboptimal value function of the concrete system.

Theorem 6.1. *Consider simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ such that $\mathcal{S}_1 \preceq_R^{\mathrm{ASR}} \mathcal{S}_2$. Given a transition cost c_2 for $\mathcal{S}_2$, we consider a transition cost c_1 for $\mathcal{S}_1$, such that*

$$c_1(x_1, u_1) \leq \min_{(x_2,u_2,x_1,u_1)\in R_e^{\mathrm{ASR}}} c_2(x_2, u_2). \tag{6.1}$$

If v_1 is a suboptimal value function for $\mathcal{S}_1$ with transition cost c_1, then

$$v_2(x_2) = \max_{x_1\in R^{-1}(x_2)} v_1(x_1) \tag{6.2}$$

is a suboptimal value function with transition cost c_2 for $\mathcal{S}_2$.

Proof. For any $(x_2, u_2, x_1, u_1) \in R_e^{\mathrm{ASR}}$, it follows from (3.8) that for all $x_1^+ \in F_1(x_1, u_1)$, we have $R(x_1^+) \cap F_2(x_2, u_2) \neq \varnothing$. Consequently, we obtain

$$\max_{x_2^+\in F_2(x_2,u_2)} \max_{x_1^+\in R^{-1}(x_2^+)} v_1(x_1^+) \geq \max_{x_1^+\in F_1(x_1,u_1)} v_1(x_1^+). \tag{6.3}$$

By the definition of R_e^{ASR}, for any $(x_1, x_2) \in R$ and any $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, there exists $u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)$ such that $(x_2, u_2, x_1, u_1) \in R_e^{\mathrm{ASR}}$. Therefore, for any $(x_1, x_2) \in R$ and any $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, it follows from (6.3) that

$$\max_{x_2^+\in F_2(x_2,u_2)} \max_{x_1^+\in R^{-1}(x_2^+)} v_1(x_1^+) \geq \min_{u_1\in\mathcal{U}_{\mathcal{S}_1}(x_1)} \max_{x_1^+\in F_1(x_1,u_1)} v_1(x_1^+), \tag{6.4}$$

which, when combined with (6.1), yields

$$c_2(x_2, u_2) + \max_{x_2^+\in F_2(x_2,u_2)} \max_{x_1^+\in R^{-1}(x_2^+)} v_1(x_1^+) \geq \min_{u_1\in\mathcal{U}_{\mathcal{S}_1}(x_1)} \left(c_1(x_1, u_1) + \max_{x_1^+\in F_1(x_1,u_1)} v_1(x_1^+) \right). \tag{6.5}$$

Thus, for any $(x_1, x_2) \in R$, we have

$$\min_{u_2\in\mathcal{U}_{\mathcal{S}_2}(x_2)} \left(c_2(x_2, u_2) + \max_{x_2^+\in F_2(x_2,u_2)} \max_{x_1^+\in R^{-1}(x_2^+)} v_1(x_1^+) \right) \geq \min_{u_1\in\mathcal{U}_{\mathcal{S}_1}(x_1)} \left(c_1(x_1, u_1) + \max_{x_1^+\in F_1(x_1,u_1)} v_1(x_1^+) \right), \tag{6.6}$$

since (6.5) holds for any $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$. Then, for any $x_2 \in \mathcal{X}_2$, we have

$$\min_{u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)} \left(c_2(x_2, u_2) + \max_{x_2^+ \in F_2(x_2,u_2)} \max_{x_1^+ \in R^{-1}(x_2^+)} v_1(x_1^+) \right) \geq \max_{x_1 \in R^{-1}(x_2)} \min_{u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)} \left(c_1(x_1, u_1) + \max_{x_1^+ \in F_1(x_1,u_1)} v_1(x_1^+) \right), \quad (6.7)$$

since (6.6) holds for any $x_1 \in R^{-1}(x_2)$. Note that (6.7) holds without any assumption on v_1.

Therefore, using (6.7) and the assumption that v_1 is a suboptimal value function, we have

$$\begin{aligned}
[\mathcal{T}_{c_2}(v_2)](x_2) &= \min_{u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)} \left(c_2(x_2, u_2) + \max_{x_2^+ \in F_2(x_2,u_2)} v_2(x_2^+) \right) \\
&= \min_{u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)} \left(c_2(x_2, u_2) + \max_{x_2^+ \in F_2(x_2,u_2)} \max_{x_1^+ \in R^{-1}(x_2^+)} v_1(x_1^+) \right) \\
&\geq \max_{x_1 \in R^{-1}(x_2)} \min_{u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)} \left(c_1(x_1, u_1) + \max_{x_1^+ \in F_1(x_1,u_1)} v_1(x_1^+) \right) \\
&= \max_{x_1 \in R^{-1}(x_2)} [\mathcal{T}(v_1)](x_1) \\
&\geq \max_{x_1 \in R^{-1}(x_2)} v_1(x_1) \\
&= v_2(x_2).
\end{aligned}$$

□

This theorem generalizes [LBJ21, Theorem 6] which was restricted to the case where R^{-1} is deterministic. If R^{-1} is deterministic, the suboptimal value function in (6.2) is $v_2 = v_1 \circ R^{-1}$ which corresponds to the result from [LBJ21, Theorem 6].

6.2 Superoptimal value function: from abstract to concrete

The following theorem demonstrates how to derive a superoptimal value function (Definition 2.15) for the concrete system from a superoptimal value function of the abstract system.

Theorem 6.2 ([ELJ22, Theorem 1]). *Consider simple systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in (3.1), and a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ such that $\mathcal{S}_1 \preceq_R^{\mathrm{ASR}} \mathcal{S}_2$. Given a transition cost c_1 for $\mathcal{S}_1$, we consider a transition cost c_2 for $\mathcal{S}_2$, such that*

$$c_2(x_2, u_2) \geq \max_{(x_2,u_2,x_1,u_1)\in R_e^{\mathrm{ASR}}} c_1(x_1, u_1). \tag{6.8}$$

If v_2 is a superoptimal value function for $\mathcal{S}_2$ with transition cost c_2, then

$$v_1(x_1) = \min_{x_2\in R(x_1)} v_2(x_2) \tag{6.9}$$

is a superoptimal value function with transition cost c_1 for $\mathcal{S}_1$.

Proof. For any $(x_2, u_2, x_1, u_1) \in R_e^{\mathrm{ASR}}$, it follows from (3.8) that for all $x_1^+ \in F_1(x_1, u_1)$, we have $R(x_1^+) \cap F_2(x_2, u_2) \neq \varnothing$. Consequently, we obtain

$$\max_{x_1^+\in F_1(x_1,u_1)} \min_{x_2^+\in R(x_1^+)} v_2(x_2^+) \leq \max_{x_2^+\in F_2(x_2,u_2)} v_2(x_2^+). \tag{6.10}$$

By the definition of R_e^{ASR}, for any $(x_1, x_2) \in R$ and any $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, there exists $u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)$ such that $(x_2, u_2, x_1, u_1) \in R_e^{\mathrm{ASR}}$. Therefore, for any $(x_1, x_2) \in R$ and any $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$, it follows from (6.10) that

$$\min_{u_1\in\mathcal{U}_1(x_1)} \max_{x_1^+\in F_1(x_1,u_1)} \min_{x_2^+\in R(x_1^+)} v_2(x_2^+) \leq \max_{x_2^+\in F_2(x_2,u_2)} v_2(x_2^+), \tag{6.11}$$

which, when combined with (6.8), yields

$$\min_{u_1\in\mathcal{U}_1(x_1)} \left(c_1(x_1, u_1) + \max_{x_1^+\in F_1(x_1,u_1)} \min_{x_2^+\in R(x_1^+)} v_2(x_2^+) \right) \leq c_2(x_2, u_2) + \max_{x_2^+\in F_2(x_2,u_2)} v_2(x_2^+). \tag{6.12}$$

Thus, for any $(x_1, x_2) \in R$, we have

$$\min_{u_1\in\mathcal{U}_1(x_1)} \left(c_1(x_1, u_1) + \max_{x_1^+\in F_1(x_1,u_1)} \min_{x_2^+\in R(x_1^+)} v_2(x_2^+) \right) \leq \min_{u_2\in\mathcal{U}_2(x_2)} \left(c_2(x_2, u_2) + \max_{x_2^+\in F_2(x_2,u_2)} v_2(x_2^+) \right), \tag{6.13}$$

since (6.12) holds for any $u_2 \in \mathcal{U}_{\mathcal{S}_2}(x_2)$. Then, for any $x_1 \in \mathcal{X}_1$, we have

$$\min_{u_1 \in \mathcal{U}_1(x_1)} \left(c_1(x_1, u_1) + \max_{x_1^+ \in F_1(x_1,u_1)} \min_{x_2^+ \in R(x_1^+)} v_2(x_2^+) \right) \leq \min_{x_2 \in R(x_1)} \min_{u_2 \in \mathcal{U}_2(x_2)} \left(c_2(x_2, u_2) + \max_{x_2^+ \in F_2(x_2,u_2)} v_2(x_2^+) \right), \quad (6.14)$$

since (6.13) holds for any $x_2 \in R(x_1)$. Note that (6.14) holds without any assumption on v_2.

Therefore, using (6.14) and the assumption that v_2 is a superoptimal value function, we have

$$\begin{aligned}
[\mathcal{T}_{c_1}(v_1)](x_1) &= \min_{u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)} \left(c_1(x_1, u_1) + \max_{x_1^+ \in F_1(x_1,u_1)} v_1(x_1^+) \right) \\
&= \min_{u_1 \in \mathcal{U}_{\mathcal{S}_1}(x_1)} \left(c_1(x_1, u_1) + \max_{x_1^+ \in F_1(x_1,u_1)} \min_{x_2^+ \in R(x_1^+)} v_2(x_2^+) \right) \\
&\leq \min_{x_2 \in R(x_1)} \min_{u_2 \in \mathcal{U}_2(x_2)} \left(c_2(x_2, u_2) + \max_{x_2^+ \in F_2(x_2,u_2)} v_2(x_2^+) \right) \\
&= \min_{x_2 \in R(x_1)} [\mathcal{T}(v_2)](x_2) \\
&\leq \min_{x_2 \in R(x_1)} v_2(x_2) \\
&= v_1(x_1).
\end{aligned}$$

□

This theorem generalizes [LBJ21, Theorem 8] which was restricted to the case where R is deterministic. If R is deterministic, the superoptimal value function in (6.9) is $v_1 = v_2 \circ R$ which corresponds to the result from [LBJ21, Theorem 8].

Note that Proposition 2.16 (resp. Proposition 2.17) allows us to combine different suboptimal value functions (resp. superoptimal value functions), such as one provided by the control engineer and another determined through abstraction by Theorem 6.1 (resp. Theorem 6.2).

6.3 Suboptimal value function: from abstract to concrete

Theorem 6.1 provides a method to construct a suboptimal value function for the abstraction $\mathcal{S}_2$ from a suboptimal value function of the concrete system $\mathcal{S}_1$. In this section, we demonstrate how to efficiently perform the reverse process: computing a suboptimal value function for the concrete system $\mathcal{S}_1$ from its abstraction $\mathcal{S}_2$.

We now introduce the deterministic version $\overrightarrow{\mathcal{S}}$ of a simple system $\mathcal{S}$, where the input is a tuple composed of the original input of $\mathcal{S}$ and the next state, allowing to deal with the non-determinism in a cooperative rather than adversarial manner.

Definition 6.3 (Associated deterministic system)**.** Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, its *associated deterministic system* is the simple system defined as $\overrightarrow{\mathcal{S}} = (\mathcal{X}, \mathcal{U} \times \mathcal{X}, \overrightarrow{F})$, where

$$\overrightarrow{F}(x, (u, x^+)) = \begin{cases} \{x^+\} & \text{if } x^+ \in F(x, u) \\ \varnothing & \text{otherwise.} \end{cases} \tag{6.15}$$

Given a cost function c for the system $\mathcal{S}$, the corresponding cost function $\overrightarrow{c}$ is defined as $\overrightarrow{c}(x, (u, x^+)) = c(x, u)$. △

We now introduce the reversed version $\overleftarrow{\mathcal{S}}$ of a simple system $\mathcal{S}$, constructed by reversing the direction of its transitions.

Definition 6.4 (Controlled reverse system)**.** Given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, its *controlled reverse system* is the simple system defined as $\overleftarrow{\mathcal{S}} = (\mathcal{X}, \mathcal{U} \times \mathcal{X}, \overleftarrow{F})$, where

$$\overleftarrow{F}(x, (u, x^-)) = \begin{cases} \{x^-\} & \text{if } x \in F(x^-, u) \\ \varnothing & \text{otherwise.} \end{cases} \tag{6.16}$$

Given a cost function c for the system $\mathcal{S}$, the corresponding controlled reverse cost function $\overleftarrow{c}$ is defined as $\overleftarrow{c}(x, (u, x^-)) = c(x^-, u)$. △

Note that both systems $\overrightarrow{\mathcal{S}}$ and $\overleftarrow{\mathcal{S}}$ are deterministic.

Given a partition of the state-space, the following theorem defines an abstraction $\mathcal{S}_2$ of a system $\mathcal{S}_1$ such that $\overrightarrow{\mathcal{S}_2}$ and $\overleftarrow{\mathcal{S}_2}$, illustrated in Figure 6.1,

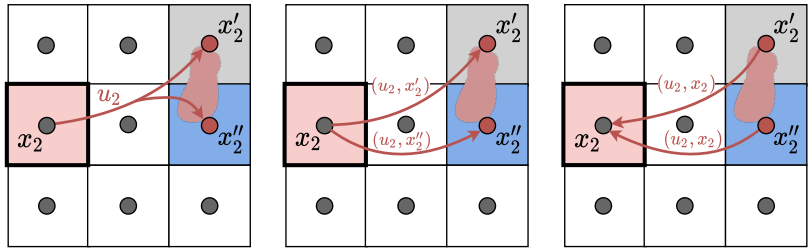


Fig. 6.1 Illustrations of transitions for the systems $\mathcal{S}_2$ (left), $\overrightarrow{\mathcal{S}_2}$ (middle), and $\overleftarrow{\mathcal{S}_2}$ (right) as defined in Theorem 6.5 for a system $\mathcal{S}_1$. The relation R is deterministic, defining a partition of the state-space. In the middle figure, one can choose the destination (optimistic), whereas in the left figure, one cannot (pessimistic).

are abstracted by $\mathcal{S}_1$ and $\overleftarrow{\mathcal{S}_1}$, respectively. This facilitates the efficient computation of a suboptimal value function for both $\mathcal{S}_1$ and $\overleftarrow{\mathcal{S}_1}$.

Theorem 6.5. *Let $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}, F_1)$ and $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}, F_2)$ be simple systems with a strict and deterministic relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ such that*

$$x_2^+ \in F_2(x_2, u) \Leftrightarrow F_1(R^{-1}(x_2), u) \cap R^{-1}(x_2^+) \neq \emptyset. \tag{6.17}$$

Then, the following hold

$$\textit{(i)}\ \mathcal{S}_1 \preceq_R^{\mathrm{FRR}} \mathcal{S}_2,\ \textit{(ii)}\ \overrightarrow{\mathcal{S}_2} \preceq_{R^{-1}}^{\mathrm{ASR}} \mathcal{S}_1,\ \textit{(iii)}\ \overleftarrow{\mathcal{S}_2} \preceq_{R^{-1}}^{\mathrm{ASR}} \overleftarrow{\mathcal{S}_1},\ \textit{(iv)}\ \overrightarrow{\mathcal{S}_1} \preceq_I^{\mathrm{FRR}} \mathcal{S}_1,$$

where $\overrightarrow{\mathcal{S}_2}$ and $\overleftarrow{\mathcal{S}_2}$ are the associated deterministic system (Definition 6.3) and the controlled reverse system (Definition 6.4) of $\mathcal{S}_2$, respectively. Here, $I = \{(x_1, x_1) \mid x_1 \in \mathcal{X}_1\}$.

Proof. To prove (i), note that $\mathcal{U}_{\mathcal{S}_2}(x_2) = \bigcap_{x_1 \in R^{-1}(x_2)} \mathcal{U}_{\mathcal{S}_1}(x_1)$. Therefore, for any $(x_1, x_2) \in R$, it holds that $\mathcal{U}_{\mathcal{S}_2}(x_2) \subseteq \mathcal{U}_{\mathcal{S}_1}(x_1)$. Additionally, condition (6.17) implies (3.9), establishing that $\mathcal{S}_1 \preceq_R^{\mathrm{FRR}} \mathcal{S}_2$.

To prove (ii), define the set

$$V = \{(x_1, u, R(x_1), (u, R(x_1^+))) : \forall x_1 \in \mathcal{X}_1, u \in \mathcal{U}_{\mathcal{S}_1}(x_1), x_1^+ \in F_1(x_1, u)\}.$$

By the definitions of $\mathcal{S}_2$ and $\overrightarrow{\mathcal{S}_2}$, we have $\{x_2^+\} = \overrightarrow{F_2}(x_2, (u, x_2^+))$ if and only if there exist $x_1 \in R^{-1}(x_2)$ and $x_1^+ \in R^{-1}(x_2^+)$ such that $x_1^+ \in F_1(x_1, u)$. Given $x_2 = R(x_1)$ and $x_2^+ = R(x_1^+)$, this shows that V is the extended

relation $(R^{-1})_e^{\mathrm{ASR}}$ of R^{-1}. For any $x_1 \in \mathcal{X}_1$, if $u \in \mathcal{U}_{\mathcal{S}_1}(x_1)$, then there exists an $x_1^+ \in F_1(x_1, u)$, and the input $(u, R(x_1^+)) \in \mathcal{U} \times \mathcal{X}_2$ ensures that $(x_1, u, R(x_1), (u, R(x_1^+))) \in (R^{-1})_e^{\mathrm{ASR}}$, thus $\overrightarrow{\mathcal{S}_2} \preceq^{\mathrm{ASR}}_{R^{-1}} \mathcal{S}_1$.

To prove (iii), similarly to (ii), define the set

$$W = \{(x_1, (u, x_1^-), R(x_1), (u, R(x_1^-))) : \forall x_1^- \in \mathcal{X}_1, u \in \mathcal{U}_{\mathcal{S}_1}(x_1^-), x_1 \in F_1(x_1^-, u)\}.$$

By the definitions of $\overleftarrow{\mathcal{S}_1}$ and $\overleftarrow{\mathcal{S}_2}$, we have $\{x_2^-\} = \overleftarrow{F_2}(x_2, (u, x_2^-))$ if and only if there exist $x_1 \in R^{-1}(x_2)$ and $x_1^- \in R^{-1}(x_2^-)$ such that $x_1 \in F_1(x_1^-, u)$. Given $x_2 = R(x_1)$ and $x_2^- = R(x_1^-)$, this shows that W is the extended relation $(R^{-1})_e^{\mathrm{ASR}}$ of R^{-1}. For any $x_1 \in \mathcal{X}_1$, if $(u, x_1^-) \in \mathcal{U}_{\overleftarrow{\mathcal{S}_1}}(x_1)$, then $x_1^- \in \overleftarrow{F_1}(x_1, (u, x_1^-))$, and the input $(u, R(x_1^-)) \in \mathcal{U} \times \mathcal{X}_2$ ensures that $(x_1, (u, x_1^-), R(x_1), (u, R(x_1^-))) \in (R^{-1})_e^{\mathrm{ASR}}$, thus $\overleftarrow{\mathcal{S}_2} \preceq^{\mathrm{ASR}}_{R^{-1}} \overleftarrow{\mathcal{S}_1}$.

The proof for (iv) follows directly from $\mathcal{S}_1 \preceq^{\mathrm{FRR}}_{I} \mathcal{S}_1$ and (ii). □

To summarize the implications of Theorem 6.5, given a transition cost function c_1 for $\mathcal{S}_1$ (resp. $\overleftarrow{c_1}$ for $\overleftarrow{\mathcal{S}_1}$), if $\mathcal{X}_2$ is a finite set, then the abstraction $\overrightarrow{\mathcal{S}_2}$ (resp. $\overleftarrow{\mathcal{S}_2}$) defined in Theorem 6.5 with the cost $\overrightarrow{c_2}$ (resp. $\overleftarrow{c_2}$) given by Theorem 6.1 is a weighted directed graph. Thus, a suboptimal value function $\overrightarrow{v_2}$ (resp. $\overleftarrow{v_2}$) for $\overrightarrow{\mathcal{S}_2}$ (resp. $\overleftarrow{\mathcal{S}_2}$) can be computed using a shortest path algorithm such as Dijkstra [Dij22] or Bellman-Ford [Bel58]. Since $\overrightarrow{\mathcal{S}_2} \preceq^{\mathrm{ASR}}_{R^{-1}} \mathcal{S}_1$ (resp. $\overleftarrow{\mathcal{S}_2} \preceq^{\mathrm{ASR}}_{R^{-1}} \overleftarrow{\mathcal{S}_1}$), by Theorem 6.1, we can construct a suboptimal value function v_1 (resp. $\overleftarrow{v_1}$) for the system $\mathcal{S}_1$ (resp. $\overleftarrow{\mathcal{S}_1}$) as follows

$$v_1(x_1) = \max_{x_2 \in R(x_1)} \overrightarrow{v_2}(x_2) = (\overrightarrow{v_2} \circ R)(x_1), \tag{6.18}$$

$$\overleftarrow{v_1}(x_1) = \max_{x_2 \in R(x_1)} \overleftarrow{v_2}(x_2) = (\overleftarrow{v_2} \circ R)(x_1), \tag{6.19}$$

which are piecewise constant functions since R is deterministic. In addition, given any simple system $\mathcal{S}_1$, since $\overrightarrow{\mathcal{S}_1} \preceq^{\mathrm{FRR}}_{I} \mathcal{S}_1$, we can construct a suboptimal value function v_1 for $\mathcal{S}_1$ from $\overrightarrow{v_1}$ for $\overrightarrow{\mathcal{S}_1}$, which can be efficiently computed using Dijkstra's algorithm for finite systems, as follows

$$v_1(x_1) = \overrightarrow{v_1}(x_1), \tag{6.20}$$

which corresponds to (6.18) with $\mathcal{S}_2 = \mathcal{S}_1$ and $R = I$.

7

Lazy and hierarchical abstractions

ABSTRACTION-based techniques provide formal guarantees for optimal control problems on nonlinear and hybrid systems. Computing an abstraction solving the problem over the whole state-space is computationally demanding in high-dimensional spaces. In this chapter and more generally in Part III, we propose to co-design the abstraction and the controller guided by the optimal control problem in order to reduce the computed part of the abstraction, and ask the following question

How can we tailor the abstraction construction to address the specific control problem considered, thereby reducing computational complexity?

To this end, we provide algorithms based on the following three observations.

First, a common approach is to seek for an abstraction that is in alternating bisimulation relation with the original system like in [AHKV98, Tab09, Gir14]. That is, the abstraction is both an alternating simulation of and alternatingly simulated by the original system. This requires both that the system satisfies some assumptions such as incremental stability (Definition 2.22) and that the abstraction is sufficiently fine-grained, which heavily intensifies the curse of dimensionality. However, bisimulation can be replaced by *alternating simulation* relations (Definition 3.1), which still

preserve key properties, such as suboptimal (Definition 2.14) and superoptimal value functions (Definition 2.15) [LBJ21]. While similar concepts had been used in particular frameworks, e.g. in [GLB14] where the concept of superoptimal value function was used for a specific case of alternating simulation, the conjugate use of alternating simulation and suboptimal and superoptimal value functions allows for leveraging optimal control tools in the framework of symbolic control. As we show in this chapter, for any partition of the state-space, one can build an alternating simulation and an alternatingly simulated system, together with a suboptimal and superoptimal value functions, which in turn provide lower and upper bounds to the optimal cost of an optimal control problem. While finer abstractions reduce the gap between the two bounds, the techniques developed in this section do not have any requirements on this gap thus allowing the use of coarser abstractions to improve scalability.

Second, while computing a fine enough abstraction over the whole state-space is often impossible due to high computational cost, computing the abstraction in a restricted area, close to the optimal trajectory is much less affected by an increase in the number of dimensions. In practice, this optimal trajectory is unknown, and we propose a methodology based on the so-called A* algorithm (revisited in Algorithm 7.4) in order to determine it in an incremental way. The A* algorithm is commonly used in artificial intelligence and usually dramatically reduces the computational time when a good consistent lower bound on the optimal cost is provided. For discrete-time systems, this lower bound is represented by a suboptimal value function. Algorithm 7.4 demonstrates how to compute an abstraction and its associated superoptimal value function in a *lazy* manner, meaning that transitions for each state of the abstraction are calculated only when necessary [HJMS02, HMMS18a, CGG11a, GGM15, TI09, HMMS18b].

Third, similarly to the crucial importance of adaptive step-size for simulations, using a uniform grid as partition of the state-space leads to poor abstraction, conservatively simulating the behavior of the original system. We show in Algorithm 7.6 how to decouple the abstraction into distinct parts of the state-space in which the suboptimal value function and the superoptimal value function as well as the abstraction itself can be computed independently and then combined. This allows the computation to be carried out in a distributed manner.

Our main goal is to combine these ideas in a hierarchical approach with three nested levels of partitions in order to solve an optimal control problem for a nonlinear system while mitigating classical problems induced by

the curse of dimensionality. We provide a *minimax with α-β pruning* algorithm [Ber05, Section 6.3.2] (Algorithm 7.6), that is able to exploit the decoupling to avoid computing some part of the abstraction if it is pruned by bounds provided by the suboptimal and superoptimal value functions.

This chapter is structured into three sections, each introducing an algorithm that builds upon the previous one. The first section deals with finite systems, while the following sections focus on constructing smart abstractions.

The results presented in this chapter are based on preliminary work published in [CLEJ21]. The corresponding numerical experiments are available in `Dionysos.jl` at `https://github.com/dionysos-dev/Dionysos.jl/releases/tag/2024_LCSS`, in the `docs/src/examples/solvers` subfolder.

7.1 Generalization of the A* algorithm for hypergraphs

In this section, we introduce Algorithm 7.4, a generalization of the A* algorithm (Algorithm B.1; see Appendix B for a description in our framework), for weighted hypergraphs, designed to find the optimal policy to enforce reach-avoid specification. While shortest hyperpath problems on hypergraphs are NP-hard in general [IN89], this particular case can be solved in polynomial time by a generalization of the Dijkstra or Bellman-Ford algorithm [GLPN93]. Therefore, Algorithm 7.4, illustrated in Figure 7.1, generalizes Algorithm B.1 from finite simple deterministic systems to finite non-deterministic simple systems. Similar to how Algorithm B.1 is a lazy version of Dijkstra's algorithm, Algorithm 7.4 is a lazy version of Algorithm 3.2.

7.1.1 Algorithm

We consider an optimal control problem (Definition 2.11) with a reach-avoid specification $\Sigma^{\text{Reach}} = [\mathcal{I}, \mathcal{T}, \mathcal{O}]$, and a transition cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{>0}$, for a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$.

Unlike Algorithm B.1, which operates *forward* (Figure B.1), Algorithm 7.4 operates *backwards* (see Figure 7.1), expanding from the target set to the initial set due to the non-deterministic nature of $\mathcal{S}$, similar to Algorithm 3.2.

Like Algorithm B.1, this algorithm maintains two sets: $\mathcal{N}$ and $\mathcal{M}$. The set $\mathcal{N}$ contains states q for which a control policy to solve the reach-avoid

Algorithm 7.4: Algorithm for computing a Bellman value function g with transition cost c for a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, based on a generalization of the A* algorithm for hypergraphs. This algorithm addresses the reach-avoid specification $\Sigma^{\text{Reach}} = [\mathcal{I}, \mathcal{T}, \mathcal{O}]$ and uses a consistent heuristic function h (7.2).

Function `LazyReachAvoidSolver`$(\mathcal{S}, \Sigma^{Reach}, c, h)$**:**
  $g(q) \leftarrow \infty$, $\forall q \in \mathcal{X}$;
  $f(q) \leftarrow \infty$, $\forall q \in \mathcal{X} \setminus \mathcal{T}$;
  $f(q) \leftarrow h(q)$, $\forall q \in \mathcal{T}$;
  $\mathcal{N} \leftarrow \mathcal{T}$;
  $\mathcal{M} \leftarrow \varnothing$;
  $\mathcal{X}^\ell \leftarrow \mathcal{I} \cup \mathcal{T} \cup \mathcal{O}$;
  while $\mathcal{N} \neq \varnothing$ *and* $\exists q \in \mathcal{I}$ *such that* $g(q) = \infty$ **do**
    $q \leftarrow \operatorname{argmin}\{f(q) \mid q \in \mathcal{N}\}$;
    $\mathcal{N} \leftarrow \mathcal{N} \setminus \{q\}$;
    $g(q) \leftarrow f(q) - h(q)$;
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{q\}$;
    for $u \in \mathcal{U}$ **do**
      for q^- *such that* $q \in F(q^-, u)$ *and* $q^- \notin \mathcal{O}$ **do**
        if $F(q^-, u)$ *not computed yet* **then**
          Compute $F(q^-, u)$;
          $F^\ell(q^-, u) \leftarrow F(q^-, u)$;
          $\mathcal{X}^\ell \leftarrow \mathcal{X}^\ell \cup \{q^-\} \cup F^\ell(q^-, u)$;
        end
        if $F^\ell(q^-, u) \subseteq \mathcal{M}$ **then**
          $f(q^-) \leftarrow \min(f(q^-), h(q^-) + c(q^-, u) + \max_{q' \in F^\ell(q^-, u)} g(q'))$;
          $\mathcal{N} \leftarrow \mathcal{N} \cup \{q^-\}$;
        end
      end
    end
  end
  $\mathcal{S}^\ell \leftarrow (\mathcal{X}^\ell, \mathcal{U}, F^\ell)$;
  return $\mathcal{S}^\ell, g, \mathcal{M}, \mathcal{N}$;
end

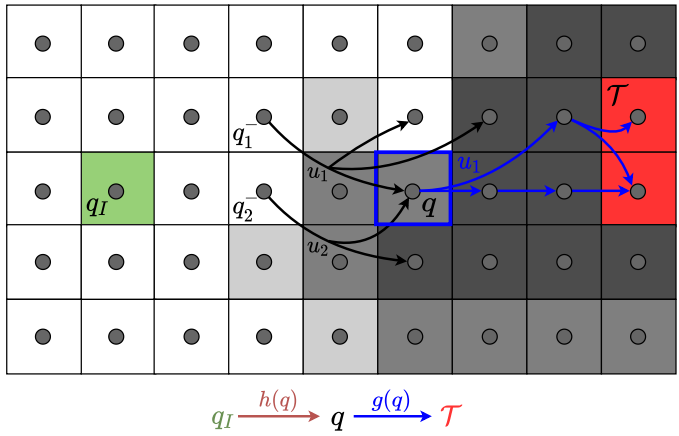


Fig. 7.1 Illustration of Algorithm 7.4. The sets $\mathcal{M}$, $\mathcal{N}$, $\mathcal{X}^\ell \setminus (\mathcal{M} \cup \mathcal{N})$, and $\mathcal{X} \setminus \mathcal{X}^\ell$ are represented in dark grey, medium grey, light grey, and white, respectively. When q is selected from $\mathcal{N}$ in Line 9, it is added to $\mathcal{M}$ and expanded, meaning that we compute its predecessors, i.e., states q' such that $q \in F(q^-, u)$ for some u. Since q_1^- is not guaranteed to reach $\mathcal{M}$ under control action u_1, q_1^- is not added to $\mathcal{N}$, while since $F(q_2^-, u_2) \subseteq \mathcal{M}$, q_2^- is added to $\mathcal{N}$ and its evaluation cost $f(q_2^-)$ is updated if $h(q_2^-) + \max_{q' \in F(q_2^-, u_2)} c(q_2^-, u_2) + g(q') < f(q_2^-)$. As illustrated, once $q \in \mathcal{M}$, $g(q)$ provides the optimal worst-case cost of traveling from q to $\mathcal{T}$, while $h(q)$ provides a lower bound on the cost to reach q from q_I.

problem $[\{q\}, \mathcal{T}, \mathcal{O}]$ is available but have not yet been *expanded*, meaning their predecessors have not yet been computed. The algorithm incrementally extends this set one step at a time until the initial set is reached. At each iteration, Algorithm 7.4 determines which state q to expand based on the cost incurred so far to reach the target set $\mathcal{T}$ from q and an estimate of the remaining cost to reach q from the initial state $\mathcal{I}$. Specifically, Algorithm 7.4 selects the state that minimizes the evaluation function

$$f(q) = g(q) + h(q), \tag{7.1}$$

where $g(q)$ is the worst-case cost from q to $\mathcal{T}$, and $h(q)$ is a *heuristic function* estimating the cost from the initial set $\mathcal{I}$ to q. Once a state q is expanded, it is removed from $\mathcal{N}$ and added to $\mathcal{M}$.

In this context, the heuristic function $h(q)$ is *admissible* if it never overestimates the actual cost to reach q from the initial set $\mathcal{I}$, and *consistent* if, for all $q^- \in \mathcal{X}$,

$$\min_{u \in \mathcal{U}_{\mathcal{S}}(q^-)} \left(c(q^-, u) + h(q^-) - \max_{q' \in F(q^-, u)} h(q') \right) \geq 0. \tag{7.2}$$

The condition (7.2) is a kind of triangular inequality, because it requires that $h(q')$, i.e. a lower bound on the cost of going from $\mathcal{I}$ to q', is no greater than the cost of going from $\mathcal{I}$ to q^- and then from q^- to q'.

The tighter the heuristic h, the smaller the portion of the state space that needs to be explored. As with Algorithm B.1, at each iteration, the sets $\mathcal{N}$ and $\mathcal{M}$ satisfy (B.3).

The following theorem demonstrates how Algorithm 7.4 lazily constructs a superoptimal value function, from which a static controller enforcing a reach-avoid specification for a finite simple system $\mathcal{S}$ can be derived.

Theorem 7.1. *Consider a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, an optimal control problem with a reach-avoid specification $\Sigma^{Reach} = [\mathcal{I}, \mathcal{T}, \mathcal{O}]$, and a transition cost function $c : \mathcal{X} \to \mathbb{R}_{>0}$. Let $h : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be a consistent heuristic function satisfying (7.2), with $h(q) = 0$ for all $q \in \mathcal{I}$. If $\mathcal{S}$ is finite, meaning $\mathcal{X}$ and $\mathcal{U}$ are finite, then Algorithm 7.4 terminates in finite time and returns a Bellman value function g with cost c, where $g(q) = 0$ for all $q \in \mathcal{T}$, and a simple system $\mathcal{S}^\ell = (\mathcal{X}^\ell, \mathcal{U}, F^\ell)$ such that $\mathcal{X}^\ell \subseteq \mathcal{X}$, and for all $q \in \mathcal{X}^\ell$ and $u \in \mathcal{U}_{\mathcal{S}^\ell}(q)$, $F^\ell(q, u) = F(q, u)$.*

Let $\mathcal{C}$ be the static system associated with g (Definition 2.12). If $\mathcal{I} \subseteq \mathcal{U}_{\mathcal{C}}(0) =$

$\{q \in \mathcal{X} \mid g(q) < \infty\}$, *then* $\mathcal{C}$ *solves the control problems* $(\mathcal{S}^\ell, \Sigma^{Reach})$ *and* $(\mathcal{S}, \Sigma^{Reach})$*; otherwise, these two control problems are infeasible.*

Proof. We observe that $\forall q \in \mathcal{N} \cup \mathcal{M}, \forall u \in \mathcal{U} : \; F^\ell(q,u) = F(q,u)$. We first show that $\forall q^- \in \mathcal{N} \cup \mathcal{M}, \forall u \in \mathcal{U}_\mathcal{S}(q^-)$: $f(q^-) \geq f(q')$ for all $q' \in F(q^-,u)$. According to Line 20, and (7.2), it holds

$$\begin{aligned}
f(q^-) &= \min_{u \in \mathcal{U}_\mathcal{S}(q^-)} \left(h(q^-) + c(q^-,u) + \max_{q' \in F(q^-,u)} g(q') \right) \\
&= \min_{u \in \mathcal{U}_\mathcal{S}(q^-)} \left(h(q^-) + c(q^-,u) + \max_{q' \in F(q^-,u)} (f(q') - h(q')) \right) \\
&\geq \min_{u \in \mathcal{U}_\mathcal{S}(q^-)} \left(h(q^-) + c(q^-,u) - \max_{q' \in F(q^-,u)} h(q') + \max_{q' \in F(q^-,u)} f(q') \right) \\
&\geq \min_{u \in \mathcal{U}_\mathcal{S}(q^-)} \left(h(q^-) + c(q^-,u) - \max_{q' \in F(q^-,u)} h(q') \right) + \min_{u \in \mathcal{U}_\mathcal{S}(q^-)} \max_{q' \in F(q^-,u)} f(q') \\
&\geq \min_{u \in \mathcal{U}_\mathcal{S}(q^-)} \max_{q' \in F(q^-,u)} f(q'). \qquad (7.3)
\end{aligned}$$

We now prove by contradiction that $g(q)$ is a Bellman value function for $\mathcal{S}$ and $\mathcal{S}^\ell$ with cost c. If this is not the case, there is a q that is selected in $\mathcal{N}$ in Line 9 while $f(q)$ would be updated to a lower value later in the algorithm. However, as q was selected in $\mathcal{N}$ in Line 9, f is less than for other $q' \in \mathcal{N}$. Hence this also holds for all future updates of f as we showed in (7.3).

Since $f[q] = h(q)$ if and only if $q \in \mathcal{T}$ by Line 4, it follows from Line 11 that $g[q] = 0$ if and only if $q \in \mathcal{T}$. Additionally, $g[q] = \infty$ if $q \in \mathcal{O}$ because, according to Line 14, $f(q)$ and therefore $g(q)$ are never updated after being initialized to ∞. Therefore, by Proposition 2.13, $\mathcal{C}$ solves the control problems $(\mathcal{S}^\ell, [\mathcal{U}_\mathcal{C}(0), \mathcal{T}, \mathcal{O}])$ and $(\mathcal{S}, [\mathcal{U}_\mathcal{C}(0), \mathcal{T}, \mathcal{O}])$, where $\mathcal{U}_\mathcal{C}(0) = \{q \in \mathcal{X} \mid g(q) < \infty\}$. If $\mathcal{I} \not\subseteq \mathcal{U}_\mathcal{C}(0)$, then $(\mathcal{S}^\ell, \Sigma^{\text{Reach}})$ and $(\mathcal{S}, \Sigma^{\text{Reach}})$ are infeasible since, according to Line 8, the algorithm has solved the fixed point equation (3.18). □

The sets $\mathcal{M}$, $\mathcal{N}$, $\mathcal{X}^\ell \setminus (\mathcal{M} \cup \mathcal{N})$, and $\mathcal{X} \setminus \mathcal{X}^\ell$ contain states for which we have the optimal worst-case policy, a policy, no policy but at least one transition has been computed from the state, and no transitions have been computed from the state, respectively.

The following proposition provides a consistent heuristic function for Algorithm 7.4 that can be efficiently computed based on the controlled reverse system.

Proposition 7.2. *Any suboptimal value function h for the controlled reverse system $\overleftarrow{\mathcal{S}}$ with costs $\overleftarrow{c}$ of $\mathcal{S}$ satisfies (7.2).*

Proof. By Definition 6.4, and since $\overleftarrow{\mathcal{S}}$ is deterministic and h is a suboptimal value function for $\overleftarrow{\mathcal{S}}$ with costs $\overleftarrow{c}$, it holds that

$$\begin{aligned}[\mathcal{T}_{\overleftarrow{c}}(h)](q) &= \min_{(u,q^-)\in\mathcal{U}_{\overleftarrow{\mathcal{S}}}(q)} \overleftarrow{c}(q,(u,q^-)) + \max_{q^-\in\overleftarrow{F}(q,(u,q^-))} h(q^-) \\ &= \min_{u\in\mathcal{U}_{\mathcal{S}}(q^-),q\in F(q^-,u)} c(q^-,u) + h(q^-) \\ &\geq h(q),\end{aligned}$$

which implies (7.2). □

7.1.2 Complexity

Given a finite simple system $\mathcal{S} = (\mathcal{X},\mathcal{U},F)$, the *number of evaluations* of F required to explicitly construct $\mathcal{S}$ is defined as

$$n_e(\mathcal{S}) = \sum_{q\in\mathcal{X}} |\mathcal{U}_{\mathcal{S}}(q)|. \tag{7.4}$$

Note, if $\mathcal{U}_{\mathcal{S}}(q) = \mathcal{U}$ for all $q \in \mathcal{X}$, then $n_e(\mathcal{S}) = |\mathcal{X}| \cdot |\mathcal{U}|$.

The system $\mathcal{S}^\ell$ returned by Algorithm 7.4 is a sub-graph of $\mathcal{S}$ that includes all transitions computed during its execution. In practice, depending on the quality of the heuristic, the number of evaluations of F to construct $\mathcal{S}^\ell$, given by $n_e(\mathcal{S}^\ell)$, is much smaller than $n_e(\mathcal{S})$, the number of evaluations required to explicitly compute the entire system $\mathcal{S}$.

7.2 Lazy abstraction

As previously mentioned, the classical abstraction-based approach (see Figure 1), involves first constructing the abstract system $\mathcal{S}_2$ over the entire state-space, regardless of the specification to be enforced, and then solving a specific control problem. In this section, we demonstrate how to combine the previously introduced concepts of suboptimal and superoptimal value functions, controlled reverse systems, and the generalization of the A* algorithm for hypergraphs, into a lazy abstraction approach (Algorithm 7.5)

whose goal is to co-design the abstraction and the controller simultaneously to solve a specific control problem, constructing the abstraction over a reduced part of the state-space.

7.2.1 Algorithm

Consider a simple system $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$ and an optimal control problem with a reach-avoid specification $\Sigma_1^{\text{Reach}} = [\mathcal{I}_1, \mathcal{T}_1, \mathcal{O}_1]$ and a transition cost function $c_1 : \mathcal{X}_1 \times \mathcal{U}_1 \to \mathbb{R}_{>0}$.

We consider two finite abstract systems $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$ and $\mathcal{S}_3 = (\mathcal{X}_3, \mathcal{U}_2, F_3)$ such that

$$\mathcal{S}_1 \preceq^{\text{ASR}}_{R_{1,2}} \mathcal{S}_2 \preceq^{\text{FRR}}_{R_{2,3}} \mathcal{S}_3, \tag{7.5}$$

where $R_{1,2} \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ is a strict relation, and $R_{2,3} \subseteq \mathcal{X}_2 \times \mathcal{X}_3$ is a strict and deterministic relation. We define the strict relation $R_{1,3} \coloneqq R_{2,3} \circ R_{1,2}$ ensuring that $\mathcal{S}_1 \preceq^{\text{ASR}}_{R_{1,3}} \mathcal{S}_3$ (Proposition 3.3).

For the remainder of this chapter, we adopt the slight abuse of notation $v(\cdot; s)$ for a value function $v : \mathcal{X} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$, meaning $v(x) = 0$ for all $x \in s$ where $s \subseteq \mathcal{X}$, to enhance readability.

The lazy abstraction algorithm operates as follows.

1. **Subroutine Algorithm:** We assume an algorithm is available to compute the abstract system $\mathcal{S}_2$. The goal of the lazy approach is to avoid constructing this abstract system $\mathcal{S}_2$ explicitly over the entire state-space by minimizing the number of evaluations of F_2.

2. **Abstract Specification:** We abstract the specification Σ_1^{Reach} according to Proposition 3.12 into $\Sigma_2^{\text{Reach}} = [\mathcal{I}_2, \mathcal{T}_2, \mathcal{O}_2]$ and consider a transition cost function c_2 as in (6.8) for $\mathcal{S}_2$.

3. **Heuristic:** To solve the control problem $(\mathcal{S}_2, \Sigma_2^{\text{Reach}})$ with transition cost c_2 using Algorithm 7.4, we need a consistent heuristic function h_2 for $\mathcal{S}_2$ (Theorem 7.1). According to Proposition 7.2, a suboptimal value function $\overleftarrow{b_2}$ with transition cost function $\overleftarrow{c_2}$ for $\overleftarrow{\mathcal{S}_2}$, such that $\overleftarrow{b_2}(q_2) = 0 \Leftrightarrow q_2 \in \mathcal{I}_2$, also denoted $\overleftarrow{b_2}(\cdot; \mathcal{I}_2)$, is a consistent heuristic for Algorithm 7.4.

   To compute $\overleftarrow{b_2}$, we rely on Section 6.3. We apply Theorem 6.5 to system $\mathcal{S}_2$ to construct the system $\mathcal{S}_3$ (Line 2), where $R_{2,3}$ is a strict and deterministic relation.

By Theorem 6.5, we have $\overleftarrow{\mathcal{S}_3} \preceq^{\text{ASR}}_{R_{2,3}^{-1}} \overleftarrow{\mathcal{S}_2}$. Therefore, as $\overleftarrow{\mathcal{S}_3}$ is a finite and deterministic system, we can efficiently compute a suboptimal value function $\overleftarrow{b_3}$ for $\overleftarrow{\mathcal{S}_3}$ with transition cost function $\overleftarrow{c_3}$ satisfying (6.1) using a shortest path algorithm such as Dijkstra (Line 4). Then, by Theorem 6.1, we can derive $\overleftarrow{b_2} = \overleftarrow{b_3} \circ R_{2,3}$ (Line 5) since $R_{2,3}$ is deterministic.

4. **Solve the Abstract Problem Lazily:** We compute the lazy abstraction $\mathcal{S}_2^\ell$ and the Bellman value function l_2 for $\mathcal{S}_2^\ell$ using Algorithm 7.4

$$(\mathcal{S}_2^\ell, l_2, \mathcal{M}_2, \mathcal{N}_2) = \texttt{LazyReachAvoidSolver}(\mathcal{S}_2, \Sigma_2^{\text{Reach}}, c_2, \overleftarrow{b_2}).$$

The Bellman value function l_2 is finite on $\mathcal{M}_2 \cup \mathcal{N}_2$. We construct the abstract controller $\mathcal{C}_2$ from l_2 according to Definition 2.12. If $\mathcal{I}_2 \subseteq \mathcal{U}_{\mathcal{C}_2}(0)$, then $\mathcal{C}_2$ solves the abstract control problem $(\mathcal{S}_2, \Sigma_2^{\text{Reach}})$; otherwise, it is infeasible.

5. **Concretization:** By Theorem 6.2, the function

$$l_1(x_1) = \min_{x_2 \in R_{1,2}(x_1)} l_2(x_2) \tag{7.6}$$

is a superoptimal value function with cost c_1 for $\mathcal{S}_1$, which is finite on $R_{1,2}^{-1}(\mathcal{M}_2 \cup \mathcal{N}_2)$. We construct $\mathcal{C}_1^{\text{ASR}}$ from $\mathcal{C}_2$ (Definition 4.14). If $\mathcal{C}_2$ solves $(\mathcal{S}_2, \Sigma_2^{\text{Reach}})$, then $\mathcal{C}_1^{\text{ASR}}$ solves $(\mathcal{S}_1, \Sigma_1^{\text{Reach}})$ (Corollary 4.19). If $\mathcal{C}_2$ does not solve the abstract problem, we cannot conclude anything about the feasibility of $(\mathcal{S}_1, \Sigma_1^{\text{Reach}})$.

Algorithm 7.5: Lazy abstraction algorithm.

1 **Function** `LazyAbstraction`$(\mathcal{S}_2, \Sigma_2^{Reach}, c_2, \mathcal{S}_3, R_{2,3})$**:**
2 `compute` $\mathcal{S}_3$;
3 $\mathcal{I}_3 \leftarrow R_{2,3}(\mathcal{I}_2)$;
4 `compute` $\overleftarrow{b_3}(\cdot; \mathcal{I}_3)$;
5 $\overleftarrow{b_2}(\cdot; \mathcal{I}_2) \leftarrow \overleftarrow{b_3}(\cdot; \mathcal{I}_3) \circ R_{2,3}$;
6 $(\mathcal{S}_2^\ell, l_2(\cdot; \mathcal{T}_2), \sim, \sim) \leftarrow$
`LazyReachAvoidSolver`$(\mathcal{S}_2, \Sigma_2^{\text{Reach}}, c_2, \overleftarrow{b_2}(\cdot; \mathcal{I}_2))$;
7 **return** $\mathcal{S}_2^\ell, l_2(\cdot; \mathcal{T}_2)$;
end

Theorem 7.3. *Algorithm 7.5 returns the (optimal) Bellman value function l_2 with costs c_2 for $\mathcal{S}_2$, from which a superoptimal value function l_1 for $\mathcal{S}_1$ can be derived as described in (7.6). Let $\mathcal{C}_2$ be the controller associated with l_2 (Definition 2.12). If $\mathcal{C}_2$ solves $(\mathcal{S}_2, \Sigma_2^{Reach})$, then $\mathcal{C}_1^{\text{ASR}}$, constructed from $\mathcal{C}_2$ (Definition 4.14), solves $(\mathcal{S}_1, \Sigma_1^{Reach})$.*

Note: All functions and parameters referenced in this theorem and its associated algorithm are defined within this section.

Proof. Refer to the discussion in Section 7.2 for the detailed proof. □

In summary, the abstraction $\mathcal{S}_2^\ell$ is computed lazily by leveraging the A* algorithm's ability to explore a reduced part of the state-space while still guaranteeing optimality. The tighter the suboptimal value function $\overleftarrow{b_2}$, the smaller the portion of the abstraction that needs to be built.

7.2.2 Complexity

To compare the computational complexity of the lazy abstraction-based approach (Algorithm 7.5), denoted $\mathrm{O}_l(\mathcal{S}_1)$, with the complexity of the classical abstraction-based approach (Algorithm 3.1), denoted $\mathrm{O}_c(\mathcal{S}_1)$, we use the number of evaluations of the abstract transition map F_2 (7.4).

The classical approach (Algorithm 3.1) requires $n_e(\mathcal{S}_2)$ evaluations of F_2 since it computes the abstraction over the entire state space

$$\mathrm{O}_c(\mathcal{S}_1) = n_e(\mathcal{S}_2). \tag{7.7}$$

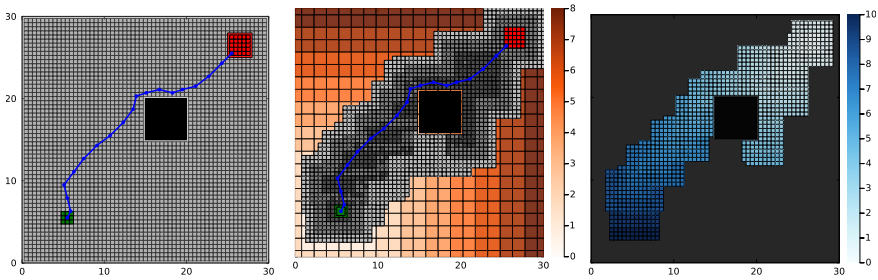
In contrast, the lazy approach constructs two abstractions

- The abstraction $\mathcal{S}_3$ used to compute the heuristic $\overleftarrow{b_2}$ must be evaluated over its entire state space, requiring $n_e(\mathcal{S}_3)$ evaluations of F_2.
- The abstraction $\mathcal{S}_2^\ell$, which is lazy version of $\mathcal{S}_2$ computed over a reduced portion of the state space, requiring $n_e(\mathcal{S}_2^\ell)$ evaluations of F_2.

The total number of evaluations of F_2 for the lazy approach is given by

$$\mathrm{O}_l(\mathcal{S}_1) = n_e(\mathcal{S}_3) + n_e(\mathcal{S}_2^\ell). \tag{7.8}$$

Similar to the classical approach, $\mathcal{S}_3$ must be computed over the entire state space. However, $\mathcal{S}_3$ is based on a coarser discretization of the state space



(a) Algorithm 3.1. **(b)** Algorithm 7.5: $\overleftarrow{b_2}$. **(c)** Algorithm 7.5: l_1.

Fig. 7.2 Outputs of Algorithm 3.1 and Algorithm 7.5 for Example 7.4. Left: Algorithm 3.1 computes $\mathcal{S}_2$ over the entire state-space. Middle: Algorithm 7.5 constructs $\mathcal{S}_2$ lazily as $\mathcal{S}_2^\ell$, with the suboptimal value function $\overleftarrow{b_2}$ shown in orange. Right: Algorithm 7.5 constructs the superoptimal value function l_1 for $\mathcal{S}_1$, finite on $R_{1,2}^{-1}(\mathcal{M}_2 \cup \mathcal{N}_2)$ in blue, with x_1 states where $l_1(x_1) = \infty$ in black.

than $\mathcal{S}_2$. Indeed,

$$|\mathcal{X}_3| = |R_{1,3}(\mathcal{X}_1)| = |R_{2,3}(R_{1,2}(\mathcal{X}_1))| = |R_{2,3}(\mathcal{X}_2)| \leq |\mathcal{X}_2|,$$

since $R_{2,3}$ is deterministic. Consequently, $n_e(\mathcal{S}_3) \leq n_e(\mathcal{S}_2)$. Additionally, since $\mathcal{S}_2^\ell$ is a lazy version of $\mathcal{S}_2$, $n_e(\mathcal{S}_2^\ell) \leq n_e(\mathcal{S}_2)$. The quality of the heuristic provided by the suboptimal value function $\overleftarrow{b_2}$, built from $\overleftarrow{b_3}$, can be improved by refining the partition of the state space embedded by $R_{2,3}$, which underlies the abstraction $\mathcal{S}_3$. This increases $n_e(\mathcal{S}_3)$ while reducing $n_e(\mathcal{S}_2^\ell)$. Therefore, there is a trade-off between computing a good but costly heuristic and obtaining an abstraction $\mathcal{S}_2^\ell$ with fewer cells to ensure that

$$\mathrm{O}_l(\mathcal{S}_1) = n_e(\mathcal{S}_3) + n_e(\mathcal{S}_2^\ell) \ll n_e(\mathcal{S}_2) = \mathrm{O}_c(\mathcal{S}_1).$$

7.2.3 Numerical experiments

We illustrate Algorithm 7.5 on the following optimal control problem for a continuous-time system.

Example 7.4. We consider the system (3.14), where $f : \mathbb{R}^2 \times \mathcal{U} \to \mathbb{R}^2$ is

given by

$$f(x,(u_1,u_2)) = \begin{pmatrix} x_1+u_1 \\ x_2+u_2 \end{pmatrix},$$

with $\mathcal{X} = [0,30]^2$ and $\mathcal{U} = [-2,2]^2$. There is no perturbation ($\mathcal{W} = \{(0,0)^\top\}$). We consider the optimal control problem $\Sigma_1^{\text{Reach}} = [\mathcal{I}_1, \mathcal{T}_1, \mathcal{O}_1]$ (Definition 2.10), with $\mathcal{I}_1 = [5,6]^2$, $\mathcal{T}_1 = [25,28]^2$, and $\mathcal{O}_1 = [15,20]^2$, illustrated in green, red, and black respectively in Figure 7.2a, and a transition cost function $c(x,u) = 0.5$. △

The function $\beta(r,u) = r$ is a growth-bound function associated with the τ-sampled system $\mathcal{S}_1$ from Example 7.4. For this numerical experiment, we use $\tau = 0.8$. We use the specific implementation described in Section 3.2.2 for the computation of F_2 as the subroutine for Algorithm 7.5.

Let $\mathcal{S}_2$ be defined as in Theorem 3.11 with $\mathcal{X}_2 = [\eta\mathbb{Z}^2] \cap \mathcal{X}_1$ (4.31) with $\eta = (0.5,0.5)^\top$ and $\mathcal{U}_2 \subseteq \mathcal{U}_1$ such that $|\mathcal{X}_2| = 3600$ and $|\mathcal{U}_2| = 80$. We define $\mathcal{S}_3$ with $\mathcal{X}_3 = [\eta'\mathbb{Z}^2] \cap \mathcal{X}_1$ with $\eta' = (1.5,1.5)^\top$ such that $|\mathcal{X}_3| = 400$. The state-space discretization of systems $\mathcal{S}_2$ and $\mathcal{S}_3$ are illustrated in Figure 7.2.

The suboptimal value function $\overleftarrow{b_2}$ constructed by Algorithm 7.5 from $\mathcal{S}_3$ is illustrated with the orange color map in Figure 7.2b. The lazy abstraction $\mathcal{S}_2^\ell$ returned by Algorithm 7.5 is shown in Figure 7.2b, where $\mathcal{M}_2$, $\mathcal{N}_2$, and $\mathcal{X}_2^\ell \setminus (\mathcal{M}_2 \cup \mathcal{N}_2)$ are illustrated in dark, medium, and light grey, respectively. The piecewise constant superoptimal value function l_1 is illustrated in Figure 7.2c. The trajectory of the closed-loop system with the initial state $x_0 = (5.5,5.5)^\top \in \mathcal{I}_1$ is shown in blue. The superoptimal value function $l_1(x_1) = 9.5$ provides an upper bound on the actual cost, which is 9.

We compare the performance based on the number of evaluations of the transition map F_2 for Algorithm 3.1 and Algorithm 7.5. Algorithm 3.1 computes $\mathcal{S}_2$ entirely, with $n_e(\mathcal{S}_2) = |\mathcal{X}_2| \cdot |\mathcal{U}_2| = 288000$ evaluations of F_2. Algorithm 7.5 computes $\mathcal{S}_3$ entirely and $\mathcal{S}_2$ lazily (see Figure 7.2c), with $n_e(\mathcal{S}_3) = |\mathcal{X}_3| \cdot |\mathcal{U}_2| = 32000$ plus $n_e(\mathcal{S}_2^\ell) = 47560$ evaluations.

Thus, the lazy abstraction approach (Algorithm 7.5) is significantly more efficient than the classical abstraction-based approach (Algorithm 3.1), requiring far fewer evaluations of F_2:

$$\mathrm{O}_l(\mathcal{S}_1) = n_e(\mathcal{S}_3) + n_e(\mathcal{S}_2^\ell) = 79560 \ll 288000 = n_e(\mathcal{S}_2) = \mathrm{O}_c(\mathcal{S}_1).$$

This efficiency gain becomes even more pronounced as the state space $\mathcal{X}_1$

grows larger or the number of inputs $|\mathcal{U}_2|$ increases, highlighting the benefits of the lazy approach.

7.3 Hierarchical Abstraction

In this section, we demonstrate how suboptimal and superoptimal value functions can be incorporated into a branch-and-bound algorithm (Algorithm 7.6), serving as the α and β functions in an α–β pruning algorithm [Ber05, Section 6.3.2]. This is achieved through a hierarchical abstraction approach with three nested levels of partitions.

7.3.1 Algorithm

Consider a simple system $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$ and an optimal control problem with a reach-avoid specification $\Sigma_1^{\text{Reach}} = [\mathcal{I}_1, \mathcal{T}_1, \mathcal{O}_1]$, where $\mathcal{I}_1 = \{x_I\}$ with $x_I \in \mathcal{X}_1$, and a transition cost function $c_1 : \mathcal{X}_1 \times \mathcal{U}_1 \to \mathbb{R}_{>0}$.

We consider three finite abstract systems

$$\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2), \quad \mathcal{S}_3 = (\mathcal{X}_3, \mathcal{U}_2, F_3), \quad \mathcal{S}_4 = (\mathcal{X}_4, \mathcal{U}_2, F_4), \tag{7.9}$$

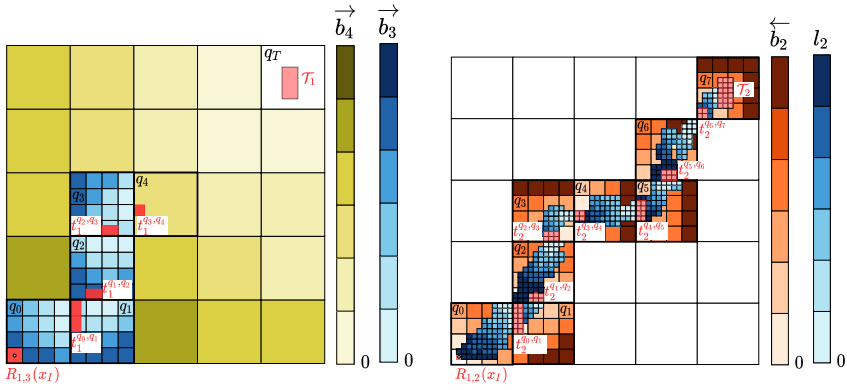
such that

$$\mathcal{S}_1 \preceq^{\text{FRR}}_{R_{1,2}} \mathcal{S}_2 \preceq^{\text{FRR}}_{R_{2,3}} \mathcal{S}_3 \preceq^{\text{FRR}}_{R_{3,4}} \mathcal{S}_4, \tag{7.10}$$

where $R_{1,2}$, $R_{2,3}$, and $R_{3,4}$ are strict deterministic relations defining three nested levels of partitions of the original state space. We define the strict deterministic relations $R_{2,4} := R_{3,4} \circ R_{2,3}$ and $R_{1,4} := R_{2,4} \circ R_{1,2}$ ensuring that $\mathcal{S}_2 \preceq^{\text{FRR}}_{R_{2,4}} \mathcal{S}_4$ and $\mathcal{S}_1 \preceq^{\text{FRR}}_{R_{1,4}} \mathcal{S}_4$ (Proposition 3.6). We assume that $\mathcal{T}_1 \subseteq R_{1,4}^{-1}(q_T)$ for some $q_T \in \mathcal{X}_4$.

The hierarchical abstraction algorithm, as illustrated in Figure 7.3, operates as follows.

The coarser abstraction $\mathcal{S}_4$ decouples the state-space into distinct parts where suboptimal and superoptimal value functions, as well as the finer abstractions, can be computed independently and then combined. It divides the overall control problem into local sub-problems, leveraging lower and upper bounds derived from suboptimal and superoptimal functions in a branch-and-bound algorithm to avoid unnecessary state-space exploration. The abstraction $\mathcal{S}_4$ acts as a planner for the trajectories of the closed-



(a) $\alpha((q_i)_{i=0}^4)$ (Algorithm 7.7). **(b)** $\beta((q_i)_{i=0}^7)$ (Algorithm 7.8).

Fig. 7.3 Illustration of Algorithm 7.6 with the three nested partitions. Left: Computation of $\alpha((q_i)_{i=0}^4)$. Local objectives $t_1^{q_j,q_{j+1}}$ are shown in red. Suboptimal value functions $\overrightarrow{b_3}(\cdot; t_3^{q_j,q_{j+1}})$ are depicted with a blue color bar, while $\overrightarrow{b_4}(\cdot; \{q_T\})$ is shown in yellow. Right: Computation of $\beta((q_i)_{i=0}^7)$ with $q_7 = q_T$. Local objectives $t_2^{q_j,q_{j+1}}$ are shown in red. Suboptimal value functions $\overleftarrow{b_2}(\cdot; t_2^{q_j,q_{j+1}})$ are depicted in orange, while superoptimal value functions $l_2(\cdot; t_2^{q_j,q_{j+1}})$ are shown in blue. Both abstractions $\mathcal{S}_3$ and $\mathcal{S}_2$ are lazily constructed via $\mathcal{S}_3^\ell$ and $\mathcal{S}_2^\ell$.

loop system.

A *node* in the branch-and-bound algorithm (Algorithm 7.6) is a finite sequence $(q_i)_{i=0}^{l}$ of states of $\mathcal{X}_4$ with $q_0 = R_{1,4}(x_I)$. It defines the sequence of abstract states of $\mathcal{X}_4$ through which we attempt to design a controller to enforce the system $\mathcal{S}_1$ to satisfy Σ_1^{Reach}. The function $\alpha((q_i)_{i=0}^{l})$ provides a lower bound on the cost to achieve Σ_2^{Reach} by initially following the sequence $(q_i)_{i=0}^{l}$, while $\beta((q_i)_{i=0}^{l})$ provides an upper bound on this cost.

This branch-and-bound algorithm builds on the lazy abstraction algorithm (Algorithm 7.5) to compute these lower and upper bounds. Systems $\mathcal{S}_2$ and $\mathcal{S}_3$ are used to solve local sub-problems, playing the same roles as in Algorithm 7.5. Specifically, $\mathcal{S}_2$ is the abstraction used to construct a superoptimal value function l_1 and controller $\mathcal{C}_1$ for $\mathcal{S}_1$ to solve $(\mathcal{S}_1, \Sigma_1^{\text{Reach}})$. System $\mathcal{S}_3$ is used to construct a suboptimal value function $\overleftarrow{b_2}$ for $\overleftarrow{\mathcal{S}_2}$, serving as a heuristic for Algorithm 7.5.

We now provide definitions for Algorithm 7.6, Algorithm 7.7, and Algorithm 7.8. The coarser abstraction $\mathcal{S}_4$ defines local objectives using the following sets. For any $q, q^+ \in \mathcal{X}_4$, we define

$$t_1^{q,q^+} = \{x_1^+ \in R_{1,4}^{-1}(q^+) \mid \exists x_1 \in R_{1,4}^{-1}(q), u \in \mathcal{U}_{\mathcal{S}_2}(x_1) : x_1^+ \in F_1(x_1, u)\}, \tag{7.11}$$

$$t_2^{q,q^+} = R_{1,2}(t_1^{q,q^+}),\ t_3^{q,q^+} = R_{2,3}(t_2^{q,q^+}). \tag{7.12}$$

For a subset $s_3 \subseteq \mathcal{X}_3$, define $\overrightarrow{b_3}(\cdot; s_3)$ (resp. $\overleftarrow{b_3}(\cdot; s_3)$) as the suboptimal value function of $\overrightarrow{\mathcal{S}_3}$ (resp. $\overleftarrow{\mathcal{S}_3}$) with cost $\overrightarrow{c_3}$ (resp. $\overleftarrow{c_3}$) satisfying (6.1) with respect to c_2 (resp. $\overleftarrow{c_2}$) according to $\overrightarrow{\mathcal{S}_3} \preceq^{\text{ASR}}_{R_{2,3}^{-1}} \mathcal{S}_2$ (resp. $\overleftarrow{\mathcal{S}_3} \preceq^{\text{ASR}}_{R_{2,3}^{-1}} \overleftarrow{\mathcal{S}_2}$). For a subset $s_4 \subseteq \mathcal{X}_4$ (resp. $t_2 \subseteq \mathcal{X}_2$), define $\overrightarrow{b_4}(\cdot; s_4)$ (resp. $l_2(\cdot; t_2)$) as the suboptimal (resp. superoptimal) value function of $\overrightarrow{\mathcal{S}_4}$ (resp. $\mathcal{S}_2$) with cost $\overrightarrow{c_4}$ (resp. c_2) satisfying (6.1) (resp. (6.8)) with respect to c_2 (resp. c_1) according to $\overrightarrow{\mathcal{S}_4} \preceq^{\text{ASR}}_{R_{2,4}^{-1}} \mathcal{S}_2$ (resp. $\mathcal{S}_1 \preceq^{\text{ASR}}_{R_{1,2}} \mathcal{S}_2$). For a subset $s_2 \subseteq \mathcal{X}_2$ (resp. $t_1 \subseteq \mathcal{X}_1$), define $\overleftarrow{b_2}(\cdot; s_2)$ (resp. $l_1(\cdot; t_1)$) as the suboptimal (resp. superoptimal) value function of $\overleftarrow{\mathcal{S}_2}$ (resp. $\mathcal{S}_1$) with cost $\overleftarrow{c_2}$ (resp. c_1).

Theorem 7.5. *Algorithm 7.6 with functions α (Algorithm 7.7) and β (Algorithm 7.8) returns a sequence of superoptimal value functions $\boldsymbol{l}_2^* = (l_2(\cdot; t_2^j))_{j=0}^{k-1}$ for $\mathcal{S}_2$ with cost c_2. Consider the optimal control problem $(\mathcal{S}_2, \Sigma_2^{Reach})$ with cost c_2, where $\Sigma_2^{Reach} = [\mathcal{I}_2, \mathcal{T}_2, \mathcal{O}_2]$ and $\mathcal{I}_2 = \{R_{1,2}(x_I)\}$, with $\mathcal{T}_2$ and $\mathcal{O}_2$ satisfying Proposition 3.12. Let $\mathcal{C}_2$ be the controller associated with $\boldsymbol{l}_2^*$ (Definition 2.19). If $\mathcal{C}_2$ solves $(\mathcal{S}_2, \Sigma_2^{Reach})$, i.e., if $\beta^* < \infty$, then $\mathcal{C}_1 = \mathcal{C}_2 \circ R_{1,2}$ solves $(\mathcal{S}_1, \Sigma_1^{Reach})$.*

Algorithm 7.6: Branch-and-bound algorithm. The set $\mathcal{B}$ represents the set of nodes of the search tree for which subtrees still need to be explored. The heuristic function h determines which node is considered next. The initial state x_I for $\mathcal{S}_1$, a *lower bound function* α, an *upper bound function* β and a maximum number of steps in $\mathcal{S}_4$ of $\tilde{l} \in \mathbb{N}$. Return a sequence of Bellman value function $\boldsymbol{l}_2^*$ for $\mathcal{S}_2$ and a upper bound on the optimal cost worst case β^*.

```
compute S_4;
l_2^* ← ();
β^* ← ∞;
B ← {R_{1,4}(x_I)};
while B ≠ ∅ do
    {(q_i)_{i=0}^l} ← argmin{h((q_i)_{i=0}^l) | (q_i)_{i=0}^l ∈ B};
    B ← B \ {(q_i)_{i=0}^l};
    if α((q_i)_{i=0}^l) ≤ β^* and l < l̃ then
        for q' ∈ X_4 do
            l̂_2, β̂ ← β(((q_i)_{i=0}^l; q'));
            if β̂ < β^* then
                l_2^*, β^* ← l̂_2, β̂;
            end
            B ← B ∪ {((q_i)_{i=0}^l; q')};
        end
    end
end
return l_2^*, β^*;
```

Algorithm 7.7: Lower bound algorithm that can be used as α function for Algorithm 7.6.

```
Function α((q_i)_{i=0}^k):
    s_3^0 ← {R_{1,3}(x_I)};
    for j ← 1 to k do
        s_3^j ← {t_3^{q_{j-1},q_j}};
        Q_j ← min_{x_3 ∈ s_3^{j-1}} b_3(x_3; s_3^j);
    end
    return (Σ_{j=1}^k Q_j) + b_4(q_k; {q_T});
end
```

Algorithm 7.8: Upper bound algorithm that can be used as β function for Algorithm 7.6.

Function $\beta((q_i)_{i=0}^k)$**:**
  $l, \text{cost} \leftarrow 0, 0$;
  $s_2^0 \leftarrow R_{1,2}(x_I)$;
  for $j \leftarrow 1$ **to** $k-1$ **do**
    $s_2^j \leftarrow \{t_2^{q_{j-1}q_j}\}$;
    $t_2^{j-1} \leftarrow \{t_2^{q_{j-1}q_j}\}$;
  end
  $t_2^{k-1} \leftarrow \mathcal{T}_2$;
  $x_2 \leftarrow R_{1,2}(x_I)$;
  for $j \leftarrow 0$ **to** $k-1$ **do**
    $(\mathcal{S}_2^\ell, l_2(\cdot; t_2^j)) \leftarrow$
     `LazyAbstraction`$(\mathcal{S}_2, [s_2^j, t_2^j, \mathcal{O}_2], c_2, \mathcal{S}_3, R_{2,3})$;
    while $x_2 \notin t_2^j$ **do**
      if $l_2(x_2; t_2^j) = \infty$ **then**
        return $(), \infty$;
      end
      $u_l \leftarrow$
       $\text{argmin}_{u \in \mathcal{U}_{\mathcal{S}_2^\ell}(x_2)} c_2(x_2, u) + \max_{x_2^+ \in F_2^\ell(x_2, u)} l_2(x_2^+; t_2^j)$;
      $\text{cost} \leftarrow \text{cost} + c_2(x_2, u_l)$;
      $x_2 \leftarrow \text{argmax}_{x_2^+ \in F_2^\ell(x_2, u_l)} l_2(x_2^+; t_2^j)$;
      $l \leftarrow l + 1$;
    end
  end
  return $(l_2(\cdot; t_2^j))_{j=0}^{k-1}, \text{cost}$;
end

Additionally, β^* provides an upper bound on the optimal worst-case cost.

Note: All functions and parameters referenced in this theorem and its associated algorithms are defined within this section.

Proof. We prove the theorem in three steps.

1. **Branch-and-bound (Algorithm 7.6):** The set $\mathcal{B}$ in Algorithm 7.6 represents the nodes of the search tree with unexplored sub-trees. Each node $(q_i)_{i=0}^{l}$ is a sequence of abstract states in $\mathcal{X}_4$. Since α and β provide lower and upper bounds, respectively, on the solution of $(\mathcal{S}_1, \Sigma_1^{\text{Reach}})$ for trajectories following the initial abstract path $(q_i)_{i=0}^{l}$, the branch-and-bound algorithm explores all possible paths in finite time as systems $\mathcal{S}_2$, $\mathcal{S}_3$ and $\mathcal{S}_4$ are finite, and $\tilde{l}$ defines an upper bound on the tree depth.

2. **Lower bound (Algorithm 7.7):**

   Since $\mathcal{S}_2 \preceq^{\text{ASR}}_{R_{2,3}} \mathcal{S}_3$ (resp. $\mathcal{S}_2 \preceq^{\text{ASR}}_{R_{2,4}} \mathcal{S}_4$), by Theorem 6.5, we have $\overrightarrow{\mathcal{S}_3} \preceq^{\text{ASR}}_{R_{2,3}^{-1}} \mathcal{S}_2$ (resp. $\overrightarrow{\mathcal{S}_4} \preceq^{\text{ASR}}_{R_{2,4}^{-1}} \mathcal{S}_2$). Therefore, by Theorem 6.1, we can construct a suboptimal value function $b_2^3 = \overrightarrow{b_3} \circ R_{2,3}$ (resp. $b_2^4 = \overrightarrow{b_4} \circ R_{2,4}$) for $\mathcal{S}_2$ from a suboptimal value function $\overrightarrow{b_3}$ for $\mathcal{S}_3$ (resp. $\overrightarrow{b_4}$ for $\mathcal{S}_4$).

   Therefore, since Algorithm 7.7 sums the lower bounds for the sequence of local objectives determined by $(q_i)_{i=0}^{k}$ along with the lower bound from q_k to q_T, $\alpha((q_i)_{i=0}^{k})$ provides a lower bound on the cost to reach $\mathcal{T}_2$ from $\mathcal{I}_2$ by following the initial abstract path $(q_i)_{i=0}^{k}$.

3. **Upper bound (Algorithm 7.8):**

   Given $\mathcal{S}_2 \preceq^{\text{ASR}}_{R_{2,3}} \mathcal{S}_3$, by Theorem 6.5, we have $\overleftarrow{\mathcal{S}_3} \preceq^{\text{ASR}}_{R_{2,3}^{-1}} \overleftarrow{\mathcal{S}_2}$. Therefore, by Theorem 6.1, we can construct a suboptimal value function $\overleftarrow{b_2} = \overleftarrow{b_3} \circ R_{2,3}$ for $\mathcal{S}_2$ from $\overleftarrow{b_3}$ for $\mathcal{S}_3$.

   Since $\mathcal{S}_1 \preceq^{\text{ASR}}_{R_{1,2}} \mathcal{S}_2$, by Theorem 6.1, we can construct a superoptimal value function $l_1 = l_2 \circ R_{1,2}$ for $\mathcal{S}_1$ from l_2 for $\mathcal{S}_2$ with costs c_2 satisfying (6.8). Given $t_2 = R_{1,2}(t_1)$ (7.12), we have $l_1(\cdot; t_1) = l_2(\cdot; t_2) \circ R_{1,2}$. By Theorem 7.3, the function $l_2(\cdot, t_2^j)$ in Line 10 is a superoptimal value function for $\mathcal{S}_2$ that can be used to reach t_2^j.

   Therefore, since Algorithm 7.8 sums the optimal costs in $\mathcal{S}_2$ by solving the sequence of local control problems determined by $(q_i)_{i=0}^{k}$,

$\beta((q_i)_{i=0}^k)$ provides the optimal cost (resp. upper bound cost) to reach $\mathcal{T}_2$ (resp. $\mathcal{T}_1$) from $\mathcal{I}_2$ (resp. $\mathcal{I}_1$) in $\mathcal{S}_2$ (resp. $\mathcal{S}_1$) by following the abstract path $(q_i)_{i=0}^k$.

Finally, by Theorem 2.20 and Theorem 3.8, if $\mathcal{C}_2$ solves $(\mathcal{S}_2, \Sigma_2^{\text{Reach}})$, then $\mathcal{C}_1$ solves $(\mathcal{S}_1, \Sigma_1^{\text{Reach}})$. □

Unlike Algorithm 7.5, the resulting concrete and abstract controllers are dynamic. As illustrated in Figure 7.3b, they enforce the system to visit the intermediate targets in sequence, with the controller's state determining the local targets considered.

The functions $\overrightarrow{b_3}(\cdot; s_3^j)$, $\overleftarrow{b_3}(\cdot; s_3^j)$, and $\overrightarrow{b_4}(\cdot; \{q_T\})$ can be efficiently computed using Dijkstra's algorithm from s_3^j, s_3^j, and q_T in $\overrightarrow{\mathcal{S}_3}$, $\overleftarrow{\mathcal{S}_3}$, and $\mathcal{S}_4$, respectively. The function $l_2(\cdot; t_2^j)$ can be lazily computed using Algorithm 7.4. The entire algorithm operates lazily, meaning that the abstraction and the suboptimal and superoptimal value functions are computed as needed, as established by the following remark.

Remark 7.6. The computation time and space for the suboptimal and superoptimal value functions in Algorithm 7.7 and Algorithm 7.8 can be significantly reduced using *memoization* techniques [CLRS22, Chapter 15]:

1. The abstraction only depends on the state $q \in \mathcal{X}_4$ and thus needs to be computed at most once.
2. For the suboptimal value function $\overrightarrow{b_3}(\cdot; t_3^j)$ (resp. $\overleftarrow{b_3}(\cdot; s_3^j)$), the value depends only on the target (resp. source), not the entire sequence $(q_i)_{i=0}^k$, allowing different sequences to reuse the same computation.
3. For the superoptimal value function $l_2(\cdot; t_2^j)$, the abstraction $\mathcal{S}_2^\ell$ is computed lazily and stops once the source is fully covered (Algorithm 7.4). When computing $l_2(\cdot; t_2^j)$ for the same target but a different source, the computation can continue from where it was left off until the new source is covered, reusing the previously computed parts of $l_2(\cdot; t_2^j)$ and $\mathcal{S}_2^\ell$.

In addition, decoupling the state-space to independently compute and combine the suboptimal and superoptimal value functions, as well as the abstraction itself, enables distributed computation, leveraging parallel computing. △

7.3.2 Complexity

As in Section 7.2.2, to compare the computational complexity of the hierarchical abstraction-based approach (Algorithm 7.6), denoted $\mathrm{O}_h(\mathcal{S}_1)$, with the complexity of the lazy (Algorithm 7.5) and the classical (Algorithm 3.1) approaches, we use the number of evaluations of the transition map F_2 (7.4).

In Algorithm 7.6, only the abstraction $\mathcal{S}_4$ is computed over the entire state-space, while $\mathcal{S}_3$ and $\mathcal{S}_2$ are lazily computed via $\mathcal{S}_3^\ell$ and $\mathcal{S}_2^\ell$, respectively, as illustrated in Figure 7.3. Therefore, the number of evaluations of F_2 for Algorithm 7.6 is given by

$$\mathrm{O}_h(\mathcal{S}_1) = n_e(\mathcal{S}_4) + n_e(\mathcal{S}_3^\ell) + n_e(\mathcal{S}_2^\ell). \tag{7.13}$$

The meta-parameters of the algorithm, such as the grid parameters that describe the abstract state-spaces, should be chosen to ensure that

$$\mathrm{O}_h(\mathcal{S}_1) = n_e(\mathcal{S}_4) + n_e(\mathcal{S}_3^\ell) + n_e(\mathcal{S}_2^\ell) \ll n_e(\mathcal{S}_2) = \mathrm{O}_c(\mathcal{S}_1). \tag{7.14}$$

7.3.3 Numerical experiments

We apply Algorithm 7.6 to the following example.

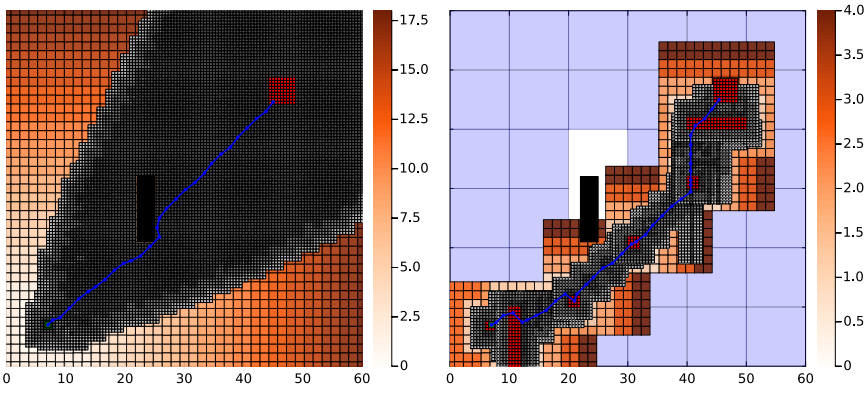
Example 7.7. We consider the same system as Example 7.4 with $\mathcal{X} = [0, 60]^2$ and the sampling time $\tau = 0.8$[1] to solve the optimal control problem $\Sigma_1^{\mathrm{Reach}} = [\mathcal{I}_1, \mathcal{T}_1, \mathcal{O}_1]$ with $\mathcal{I}_1 = [6.5, 7.5]^2$, $\mathcal{T}_1 = [6.5, 7.5]^2$, $\mathcal{O}_1 = [22, 25] \times [21, 32]$, $\mathcal{T}_2 = [44, 49]^2$, and a transition cost function $c_1(x, u) = 0.5$. △

We define $\mathcal{S}_2$ with $\mathcal{X}_2 = [\eta\mathbb{Z}^2] \cap \mathcal{X}_1$ (4.31) with $\eta = (0.5, 0.5)^\top$ and $\mathcal{U}_2 \subseteq \mathcal{U}_1$ such that $|\mathcal{X}_2| = 14400$ and $|\mathcal{U}_2| = 80$. We define $\mathcal{S}_3$ with $\mathcal{X}_3 = [\eta'\mathbb{Z}^2] \cap \mathcal{X}_1$ with $\eta' = (1.5, 1.5)^\top$ such that $|\mathcal{X}_3| = 1600$. We define $\mathcal{S}_4$ with $\mathcal{X}_4 = [\eta''\mathbb{Z}^2] \cap \mathcal{X}_1$ with $\eta'' = (10, 10)^\top$ such that $|\mathcal{X}_4| = 36$.

The results of Algorithm 7.6 for a specific node of the branch-and-bound algorithm are illustrated in Figure 7.4b. The blue trajectory illustrates the closed-loop system with the initial state $x_0 = (7, 7)^\top \in \mathcal{I}_1$.

We compare the performance through the number of evaluations of the transition map F_2 for Algorithm 3.1, Algorithm 7.5, and Algorithm 7.6. Algorithm 3.1 computes $\mathcal{S}_2$ entirely, with $n_e(\mathcal{S}_2) = |\mathcal{X}_2| \cdot |\mathcal{U}_2|$ evaluations of F_2. Algorithm 7.5 computes $\mathcal{S}_3$ entirely and $\mathcal{S}_2$ lazily (see Figure 7.4a),

[1] Note that different time steps can be used for $\mathcal{S}_2$, $\mathcal{S}_3$, and $\mathcal{S}_4$, often with larger steps for coarser abstractions.



(a) Algorithm 7.5. **(b)** Algorithm 7.6.

Fig. 7.4 Outputs of Algorithm 7.5 and Algorithm 7.6 for Example 7.7, with performance metrics provided in Table 7.1. Left: Algorithm 7.5 constructs $\mathcal{S}_3$ over the entire state space and $\mathcal{S}_2$ lazily as $\mathcal{S}_2^\ell$, with the suboptimal value function $\overleftarrow{b_2}$ shown in orange. Right: Algorithm 7.6 constructs $\mathcal{S}_3$ and $\mathcal{S}_2$ lazily as $\mathcal{S}_3^\ell$ and $\mathcal{S}_2^\ell$, with local objectives and suboptimal value functions illustrated in red and orange, respectively. $\mathcal{S}_4$ is constructed over the entire state space.

	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	O
Algorithm 3.1	**1152000**	\	\	1152000
Algorithm 7.5	294457	**128000**	\	422457
Algorithm 7.6	74787	49770	**2880**	127437

Table 7.1 Number of evaluations of F_2 for the different abstractions generated by the algorithms for Example 7.7, as illustrated in Figure 7.4. The bold quantities correspond to $n_e(\mathcal{S}_2)$, $n_e(\mathcal{S}_3)$, and $n_e(\mathcal{S}_4)$. The total number of evaluations, $\mathrm{O}_c(\mathcal{S}_1)$, $\mathrm{O}_l(\mathcal{S}_1)$, and $\mathrm{O}_h(\mathcal{S}_1)$, are given in the last column.

with $n_e(\mathcal{S}_3) = |\mathcal{X}_3| \cdot |\mathcal{U}_2|$ plus $n_e(\mathcal{S}_2^\ell)$ evaluations. Algorithm 7.6 computes $\mathcal{S}_4$ entirely and $\mathcal{S}_3$ and $\mathcal{S}_2$ lazily (see Figure 7.4b), with $n_e(\mathcal{S}_4) = |\mathcal{X}_4| \cdot |\mathcal{U}_2|$ plus $n_e(\mathcal{S}_3^\ell)$ plus $n_e(\mathcal{S}_2^\ell)$ evaluations. These values are summarized in Table 7.1.

As noted in Section 7.2.3, the lazy abstraction approach (Algorithm 7.5) is more efficient than the classical approach (Algorithm 3.1), as it constructs the finer abstraction on a reduced part of the state space. However, computing a good heuristic over the entire state space can still require a significant number of evaluations of F_2, as referenced in Table 7.1. Therefore, using local objectives with their own locally computed heuristics (Algorithm 7.6) allows for a drastic reduction in the number of evaluations of F_2.

7.4 Summary

In this chapter, we formulated modular foundations for the construction of hierarchical abstractions. We formalized the modular foundations between alternating simulations and the Bellman operator. This provides a modular methodology to transfer bounds on the optimal cost from abstractions from different levels of nested partitions. We developed a branch and bound algorithm that leverages the bounds gathered from the the constructed abstractions. Finally, numerical experiments illustrate the practical significance of these results.

8

Lazy abstraction based on non-uniform optimized cells

The classical abstraction-based technique described in Section 3.2 relies on discretizing both the state and input spaces using uniform hyperrectangles. However, the curse of dimensionality, significantly affects the uniform discretization of the entire state space due to the exponential growth of the number of states with respect to the dimension. Additionally, to account for the quantization error between the actual state and the quantized state, it is required to over-approximate the forward image of the cells under constant discretized inputs. One of the main drawbacks of this approach is that, in the absence of incremental stability (δ-GAS, Definition 2.22), over-approximation increases the level of non-determinism in the symbolic system, which could result in an intractable or even unsolvable symbolic problem.

In Chapter 7, while we provide a strategy to lazily (i.e., postponing heavier numerical operations) build an abstraction along the optimal trajectory, the local control design is performed combinatorially on a discretized input set, which is likely a source of non-determinism. In [ELJ22], while the authors propose to design local feedback controllers between nearby cells to eliminate the non-determinism in the abstraction (i.e., the abstraction is a weighted digraph instead of a hypergraph), it relies on a predefined ellipsoidal cover of the entire state space, which suffers from

the curse of dimensionality.

In light of these shortcomings, we propose in this chapter a new approach that leverages the lazy construction of Chapter 7 and the optimal control solution proposed in [ELJ22] by optimizing not only the transitions but also the positioning and shape of the cells along the optimal trajectory (see Figure 8.1). Precisely, our solution relies on a Rapidly-Exploring Random Trees (RRT) algorithm that *lazily* constructs the abstraction on the basis of an *ellipsoidal covering* of the state space and a finite set of *local affine controllers*. Unlike [RZ16] and [HMMS19], one of the key advantages of our approach is that, instead of discretizing the input space, we use a finite set of local affine controllers as our symbolic input set. For L-smooth nonlinear systems, the combination of linearization and the Lipschitz constant allows us to locally optimize such controllers using semi-definite programming without the need for extensive discretization. Moreover, the proposed approach differs from classical techniques as the partitioning is designed smartly, building the abstraction iteratively, instead of adopting a predefined uniform partition, which is sub-optimal and prone to the curse of dimensionality.

The RRT algorithm [L+98] is a sampling-based path planning algorithm [KL00, LKJ01] that generates open-loop trajectories for nonlinear systems. Therefore, these techniques are not robust to perturbations and can only provide probabilistic guarantees [KF11]. In [MN21], the authors propose a safety-critical path planner based on RRT and relying on control barrier functions. In this same vein, the authors of [RD19] propose an abstraction-based technique relying on the RRT algorithm limited to deterministic systems with additional stability assumptions (δ-GAS). In [WLS+22], the authors propose a reachable set-based RRT planning algorithm that provides a formal guarantee in the presence of bounded perturbations, does not take into account the optimal control problem and is limited to piecewise constant controllers. Furthermore, their framework does not allow to optimize the shape of new cells. Finally, the authors of [AGS13] propose a forward control strategy for reachability based on ellipsoids, an affine linear approximation with an affine controller via the resolution of linear matrix inequalities (LMI).

This chapter differs from the previously mentioned works by the following points. Firstly, our approach not only provides a controller to solve a specific control problem, but also an abstraction of the original system that can be reused as a basis for solving other optimal control problems. Secondly, our RRT algorithm does not act on states but on sets of states,

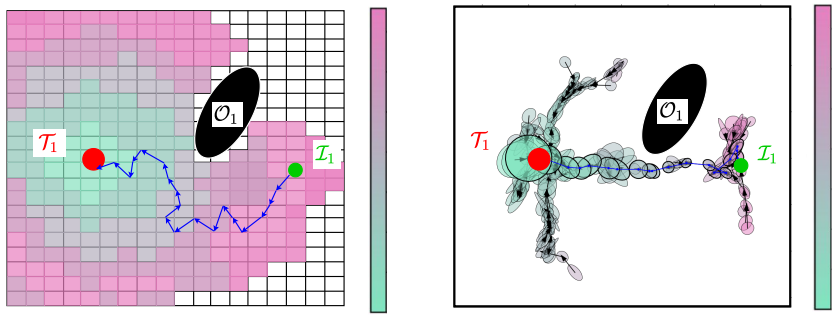


Fig. 8.1 Comparison between the classical approach and the smart state-feedback abstraction for a planar system with state trajectory (blue line) and superoptimal value function (color map) obtained for the optimal control problem of departing from $\mathcal{I}_1$ and reaching $\mathcal{T}_1$ while avoiding obstacles $\mathcal{O}_1$. Left: Classical abstraction approach based on predefined grid discretization of the entire state-space (Algorithm 3.1). Non-colored represents a region where no controller could be designed. Right: Smart deterministic abstraction lazily constructed with locally optimized ellipsoids and controllers (Algorithm 8.9).

which makes it possible to build a robust controller that provides formal guarantees even in the presence of noise. Thirdly, in contrast with [AGS13], our RRT algorithm operates in a backward manner. In this configuration, the target set serves as the root of the growing tree, offering the advantages of 1) maximizing the volume of newly added ellipsoids, thereby covering more of the state space with a single cell in the abstraction, and 2) effectively handling obstacles in the state space, as discussed later.

In Section 8.1, we introduce the specific type of abstraction relation used throughout this chapter. Section 8.2 details a general method for lazily constructing an abstraction using non-uniform cells, without restricting the class of systems or cost function templates. From Section 8.3 onward, we present a specialized and optimized implementation tailored for a specific class of L-smooth nonlinear systems with quadratic cost functions. This approach leverages linearization to address nonlinearities and uses the Lipschitz constant to reformulate the problem as a convex optimization with LMI constraints. In Section 8.4, we demonstrate this approach with a planar nonlinear dynamical system, and in Section 8.5, we summarize the contributions of this algorithm.

The results presented in this chapter are published in [CEJ24] and results from a collaboration with Lucas N. Egidio. The corresponding numerical experiments presented in Section 8.4 are available in `Dionysos.jl` at `https://github.com/dionysos-dev/Dionysos.jl/releases/tag/2024_LCSS` in the subfolder `utils/CDC2024`.

8.1 State-feedback abstraction

The abstraction-based approach presented in this chapter involves computing a deterministic abstraction, referred to as a *state-feedback abstraction* [ELJ22, Definition 1], that is related to the concrete system through a feedforward abstraction relation (Definition 4.4).

Definition 8.1 (State-feedback abstraction)**.** Given a simple system $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$, a *state-feedback abstraction* $\mathcal{S}_2$ of $\mathcal{S}_1$ is defined as a simple system $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$ that satisfies the following conditions: $\mathcal{X}_2 \subseteq 2^{\mathcal{X}_1}$, $\mathcal{U}_2 \subseteq \mathscr{F}(\mathcal{X}_1, \mathcal{U}_1)$, and

$$\forall \xi \in \mathcal{X}_2, \forall \kappa \in \mathcal{U}_{\mathcal{S}_2}(\xi) : F_2(\xi, \kappa) = \{\xi^+\}$$

where

$$\xi^+ \supseteq \{x^+ \in F_1(x, \kappa(x)) \mid x \in \xi\}. \tag{8.1}$$

△

The state-feedback abstraction is a specific instance of the feedforward abstraction relation (Definition 4.4).

Proposition 8.2. *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two simple systems where $\mathcal{S}_2$ is a state-feedback abstraction of $\mathcal{S}_1$. Then, $\mathcal{S}_1 \preceq_R^{\text{FAR}} \mathcal{S}_2$ with $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ defined as*

$$(x, \xi) \in R \Leftrightarrow x \in \xi. \tag{8.2}$$

Proof. By [ELJ22, Lemma 1], we have $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$. By Proposition 4.7, $\mathcal{S}_1 \preceq_R^{\text{FAR}} \mathcal{S}_2$ since $\mathcal{S}_2$ is deterministic. □

Note that for the rest of this chapter, we will use $x \in \xi$ instead of $x \in R^{-1}(\xi)$ to simplify the notations.

8.2 Algorithm

Consider a simple system $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$ with $\mathcal{X}_1 \subseteq \mathbb{R}^{n_x}$ and $\mathcal{U}_1 \subseteq \mathbb{R}^{n_u}$, and an optimal control problem with a reach-avoid specification $\Sigma_1^{\text{Reach}} = [\mathcal{I}_1, \mathcal{T}_1, \mathcal{O}_1]$ and a transition cost function $c_1 : \mathcal{X}_1 \times \mathcal{U}_1 \to \mathbb{R}_{>0}$.

The nonlinear nature of the system and of the reach-avoid specification makes it extremely difficult to solve the optimal control problem directly. Therefore, to approximately solve this problem our abstraction-based approach will split it up into several convex subproblems. The approach involves using a RRT (Rapidly-Exploring Random Trees) algorithm that grows a state-feedback abstraction $\mathcal{S}_2$ of $\mathcal{S}_1$ with an underlying *tree* structure from the target set to the initial set. The nodes of this tree represent abstract states, each of which is a *set* $\xi \in 2^{\mathcal{X}_1}$ and transitions from child nodes to their parents are handled by *local state-feedback controllers* $\kappa \in \mathscr{F}(\mathcal{X}_1, \mathcal{U}_1)$. While the strategy presented is theoretically applicable to any set templates for $\mathcal{X}_2$ and function templates for $\mathcal{U}_2$, we propose a practical implementation (Algorithm 8.9) using *ellipsoids* and *affine functions* templates, respectively, to take advantage of the power of LMIs.

The functions that implement Algorithm 8.9 (illustrated in Figure 8.2) are described as follows:

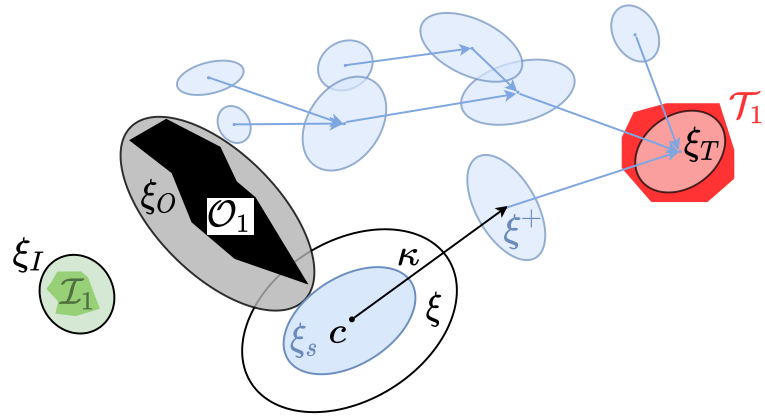


Fig. 8.2 Illustration of Algorithm 8.9. First, Line 1 returns abstract specifications satisfying $\mathcal{I}_1 \subseteq \xi_I, \xi_T \subseteq \mathcal{T}_1$, and $\mathcal{O}_1 \subseteq \xi_O$. Then, given a point $c \in \mathcal{X}_1$ generated in Line 5, the algorithm selects the closest abstract state ξ^+ (Line 6). In Line 7, it maximizes the volume of $\xi = \mathrm{E}(c, \mathbf{P})$ according to $\mathbf{P}$ and κ to ensure that $F_1(x, \kappa(x)) \subseteq \xi^+$ for all $x \in \xi$. Finally, in Line 9, the ellipsoid ξ is shrunk into ξ_s to ensure that $\xi_s \cap \xi_O = \emptyset$.

—`getAbsSpecification(`Σ_1^{Reach}`)`: returns the abstract specification $\Sigma_2^{\text{Reach}} = [\xi_I, \xi_T, \xi_O]$, where ξ_I, ξ_T, ξ_O are conservative ellipsoids satisfying Proposition 3.12.
—`getConState(`$\mathcal{X}_1$`)`: returns a candidate concrete state $c \in \mathcal{X}_1$.
—`getKClosestAbsStates(`$\mathcal{X}_2, \mathcal{A}, K$`)`: returns the K abstract states of $\mathcal{X}_2$ closest to the set $\mathcal{A}$ according to Euclidean distance.
—`solveLocalProblem(`c, ξ^+`)`: returns $\xi = \mathrm{E}(c, \mathbf{P})$ (1.4), a controller κ such that $\forall x \in \xi : F(x, \kappa(x)) \subseteq \xi^+$ and $\kappa(\xi) \subseteq \mathcal{U}_1$, and an upper bound $\widetilde{\mathcal{J}}$ on the transition cost from ξ to ξ^+, i.e., $\widetilde{\mathcal{J}} \geq \max_{x \in \xi} c_1(x, \kappa(x))$.
—`handleObstacles(`ξ, ξ_O`)`: given $\xi = \mathrm{E}(c, \mathbf{P})$, returns a new ellipsoid $\xi_s = \mathrm{E}(c, \gamma \mathbf{P})$ where $\gamma \geq 1$ is the smallest value such that ξ_s does not intersect any obstacles, i.e. $\xi_s \cap \xi_O = \emptyset$.
—`getAbsValueFunction(`$\mathcal{S}_2$, ξ_T`)`: returns a superoptimal value function l_2 for $\mathcal{S}_2$ such that $l_2(\xi_T) = 0$ and which is strictly positive for the other abstract states.
—`newTransition(`ξ', ξ`)`: returns a controller κ that maps points from a given ellipsoid ξ' to another given ellipsoid ξ, and the associated transition costs $\widetilde{\mathcal{J}}'$.

Algorithm 8.9: Lazy construction of an ellipsoid-based state-feedback abstraction $\mathcal{S}_2$ of $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$ and of an abstract superoptimal value function l_2 that solves $(\mathcal{S}_1, \Sigma_1^{\text{Reach}})$ with $\Sigma_1^{\text{Reach}} = [\mathcal{I}_1, \mathcal{T}_1, \mathcal{O}_1]$.

1 $\Sigma_2^{\text{Reach}} \leftarrow$ `getAbsSpecification`$(\Sigma_1^{\text{Reach}})$;
2 $\xi_I, \xi_T, \xi_O \leftarrow \Sigma_2^{\text{Reach}}$;
3 $\mathcal{X}_2 \leftarrow \{\xi_T\}$;
4 **while** not exists $\xi \in \mathcal{X}_2 : \xi_I \subseteq \xi$ **do**
5 $\quad c \leftarrow$ `getConState`$(\mathcal{X}_1)$;
6 $\quad \xi^+ \leftarrow$ `getKClosestAbsStates`$(\mathcal{X}_2, \{c\}, K{=}1)$;
7 $\quad \xi, \kappa, \widetilde{\mathcal{J}} \leftarrow$ `solveLocalProblem`(c, ξ^+);
8 $\quad$ **if** not *feasible* **then return** to Line 5 ;
9 $\quad \xi_s \leftarrow$ `handleObstacles`(ξ, ξ_O);
10 $\quad \mathcal{X}_2, \mathcal{U}_2 \leftarrow \mathcal{X}_2 \cup \{\xi_s\}, \mathcal{U}_2 \cup \{\kappa\}$;
11 $\quad F_2(\xi_s, \kappa) \leftarrow \{\xi^+\}$;
12 $\quad c_2(\xi_s, \kappa) \leftarrow \widetilde{\mathcal{J}}$;
13 $\quad \mathcal{S}_2 \leftarrow (\mathcal{X}_2, \mathcal{U}_2, F_2)$;
14 $\quad l_2 \leftarrow$ `getAbsValueFunction`$(\mathcal{S}_2, \xi_T)$;
15 $\quad$ `improveAbs`$(\mathcal{S}_2, l_2, \xi_s, \widetilde{\mathcal{J}})$;
$\quad$ **end**
16 **return** $\mathcal{S}_2, l_2$;

Algorithm 8.10: Update of the abstraction for the RRT* variant.

1 **function** `improveAbs`$(\mathcal{S}_2, l_2, \xi, \widetilde{\mathcal{J}})$**:**
2 $\quad \xi_{\text{list}} \leftarrow$ `getKClosestAbsStates` $(\mathcal{X}_2, \xi, K{\geq}1)$;
3 $\quad$ **for** $\xi' \in \xi_{\text{list}}$ **do**
4 $\quad\quad \kappa', \widetilde{\mathcal{J}}' \leftarrow$ `newTransition`(ξ', ξ);
5 $\quad\quad$ **if** $\widetilde{\mathcal{J}}' + l_2(\xi) < l_2(\xi')$ **then**
6 $\quad\quad\quad \mathcal{U}_2 \leftarrow \mathcal{U}_2 \cup \{\kappa'\}$;
7 $\quad\quad\quad F_2(\xi', \kappa') = \{\xi\}$;
8 $\quad\quad\quad c_2(\xi', \kappa') = \widetilde{\mathcal{J}}'$;
$\quad\quad$ **end**
$\quad$ **end**
end

As mentioned in the introduction of the chapter, our RRT algorithm works backwards in the sense that the abstraction is initialized with the target set ξ_T, and when designing a new transition, the optimized cells are preceding ones (in terms of the controlled system trajectory). This allows us to maximize the volume of the preceding ellipsoid, covering a larger portion of the state space with a single abstract state—a feature not achievable in the forward approach, which focuses on minimizing the volume of the destination ellipsoid [AGS13]. In addition, the backward approach makes it possible to decouple the design of the preceding ellipsoid ξ and the controller κ from obstacle handling. Indeed, when a preceding ellipsoid ξ returned by `solveLocalProblem` intersects an obstacle, it can be shrunk into $\xi_s \subseteq \xi$ and the same controller κ still ensures a transition to ξ^+ (see Figure 8.2). This decoupling is not feasible in the forward approach.

The function `improveAbs` implements the RRT* variant. When a new abstract state ξ_s is added, new transitions are computed from the closest existing abstract states to ξ_s in order to eventually reduce the path cost to the target set ξ_T.

The function `getConState` is a *heuristic* used to generate a concrete state from which a new ellipsoid is added to the abstraction. Although the specific implementation of this function does not affect the correctness of Algorithm 8.9, it can be used to guide the exploration of the state space and the construction of the abstraction in order to reduce the number of abstract states. For example, a simple but effective strategy involves sampling points predominantly in the direction of the initial set $\mathcal{I}_1$.

The following theorem guarantees the validity of the approach.

Theorem 8.3. *Given a simple system $\mathcal{S}_1$, consider the simple system $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$ and the value function l_2 returned by Algorithm 8.9. The following statements hold: (1) $\mathcal{S}_2$ is a state-feedback abstraction of $\mathcal{S}_1$. (2) The controller $\mathcal{C}_2$ associated with l_2 solves $(\mathcal{S}_2, \Sigma_2^{Reach})$. (3) The controller $\mathcal{C}_1^{\mathrm{FAR}}$, derived from $\mathcal{C}_2$ and R (8.2), according to Definition 4.16, solves $(\mathcal{S}_1, \Sigma_1^{Reach})$. (4) The function $l_1(x) = \min_{\xi \in R(x)} l_2(\xi)$ is a superoptimal value function for $\mathcal{S}_1$ with costs c_1.*

Proof. (1) The transition map F_2 is constructed according to $F_2(\xi_s, \kappa) = \{\xi^+\}$, where `solveLocalProblem` guarantees that $\forall x \in \xi : F_1(x, \kappa(x)) \subseteq \xi^+$, and `handleObstacles` ensures that $\xi_s \subseteq \xi$, which implies $\forall x \in \xi_s : F_1(x, \kappa(x)) \subseteq \xi^+$. Consequently, F_2 satisfies the condition (8.1).

(2) Firstly, for all $\xi \in \mathcal{X}_2$, there exists a path from ξ to ξ_T since $\mathcal{X}_2$ is initialized with ξ_T and newly added transitions are always directed to an existing abstract state in $\mathcal{X}_2$. In addition, these paths avoid obstacle

ξ_O thanks to `handleObstacles`. Secondly, the ending condition of Algorithm 8.9 guarantees that $\xi_I \subseteq \xi$ for some $\xi \in \mathcal{X}_2$. As a result, the controller $\mathcal{C}_2$ associated with l_2 solves $(\mathcal{S}_2, \Sigma_2^{\text{Reach}})$ where $\Sigma_2^{\text{Reach}} = [\xi_I, \xi_T, \xi_O]$.

(3) Since $\mathcal{S}_1 \preceq_R^{\text{FAR}} \mathcal{S}_2$ (Proposition 4.7) and Σ_2^{Reach} satisfies Proposition 3.12 according to `getAbsSpecification`, then, by Corollary 4.19, the controller $\mathcal{C}_1^{\text{FAR}}$ solves $(\mathcal{S}_1, \Sigma_1^{\text{Reach}})$.

(4) This follows directly from Proposition 2.17 since the abstract transition cost function c_2 satisfies (6.8) with c_1, as it is equal to the upper bound cost of the local transition according to `solveLocalProblem`. □

The abstract superoptimal value function l_2 is derived in Line 14 of Algorithm 8.9 (`getAbsValueFunction`) by computing the shortest path to ξ_T, which can be done efficiently since ξ_T is the root of a tree. The optimization problem in `handleObstacles` and the inclusion test in Line 4 of Algorithm 8.9 can be efficiently addressed by solving a convex scalar optimization problem, as outlined in Corollary C.8 and Algorithm C.1, respectively. Finally, the functions `solveLocalProblem` and `newTransition` can be implemented efficiently by solving a convex optimization problem as described in the following section.

8.3 Local controller design

In this section, we provide a practical and efficient implementation of the function `solveLocalProblem` in Line 7 of Algorithm 8.9 for the following general class of systems.

We consider nonlinear simple system $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$ with bounded disturbances, i.e.,

$$F_1(x, u) = \{f(x, u, w) \mid w \in \mathcal{W}\} \tag{8.3}$$

where $\mathcal{X}_1 \subseteq \mathbb{R}^{n_x}$, $\mathcal{U}_1 \subseteq \mathbb{R}^{n_u}$, $f : \mathcal{X}_1 \times \mathcal{U}_1 \times \mathcal{W} \to \mathcal{X}_1$ is a continuous nonlinear function whose Jacobian is globally Lipschitz continuous [1], and $\mathcal{W} \subseteq \mathbb{R}^{n_w}$ is a bounded polytopic set. The control input takes values in the intersection of (possibly degenerated) ellipsoids

$$\mathcal{U}_1 = \bigcap_{k \in [1; N_u]} \mathcal{U}_{1,k} \tag{8.4}$$

[1] Note that similar results can be adapted for the locally Lipschitz continuous case.

with $\mathcal{U}_{1,k} = \{u \in \mathbb{R}^{n_u} : \|\mathbf{U}_k u\|_2 \leq 1\}$ for some matrix $\mathbf{U}_k$ of appropriate dimensions. The exogenous input takes values inside the convex hull of a set of points

$$\mathcal{W} = \text{conv}\{w_1, \ldots, w_{N_w}\} \tag{8.5}$$

with $0 \in \mathcal{W}$ and introduces a level of non-determinism to the system, which can either capture non-modeled behaviors or adversarial disturbances. We consider a general quadratic transition cost function of the form

$$c_1(x,u) = \begin{pmatrix} x^\top & u^\top & 1 \end{pmatrix} \mathbf{Q} \begin{pmatrix} x^\top & u^\top & 1 \end{pmatrix}^\top \tag{8.6}$$

defined for some given matrix $\mathbf{Q} \succ \mathbf{0}$.

Therefore, the objective of this section is to design a controller κ to map all the states of $\xi \in 2^{\mathcal{X}_1}$ into $\xi^+ \in 2^{\mathcal{X}_1}$ despite exogenous noise, i.e.,

$$\forall x \in \xi \; \forall w \in \mathcal{W} : \; f(x, \kappa(x), w) \in \xi^+, \tag{8.7}$$

and such that the inputs of the closed loop system lie within the admissible input set $\mathcal{U}_1$, i.e.,

$$\kappa(\xi) \subseteq \mathcal{U}_1. \tag{8.8}$$

8.3.1 Specific approximation scheme

Since optimizing directly on the nonlinear dynamics f is a challenging problem, we rely on the linearized function $\tilde{f}$ around a point $\bar{p} = (\bar{x}, \bar{u}, \bar{w}) \in \mathcal{X}_1 \times \mathcal{U}_1 \times \mathcal{W}$:

$$\tilde{f}(x,u,w) = \mathbf{A}x + \mathbf{B}u + \mathbf{E}w + g \tag{8.9}$$

with $\mathbf{A} = J_{f,x}(\bar{p})$, $\mathbf{B} = J_{f,u}(\bar{p})$, $\mathbf{E} = J_{f,w}(\bar{p})$, $g = f(\bar{x}, \bar{u}, \bar{w}) - \mathbf{A}\bar{x} - \mathbf{B}\bar{u} - \mathbf{E}\bar{w}$, where $J_{f,x}(\bar{p})$ is the Jacobian matrix of f with respect to the variables x evaluated at $\bar{p}$.

We can derive a component-wise bound on the linearization error [N+18, Lemma 1.2.3], i.e., $\forall (x,u,w) \in \mathcal{X}_1 \times \mathcal{U}_1 \times \mathcal{W}$:

$$f(x,u,w) \in \{\tilde{f}(x,u,w)\} \oplus \Omega(x,u,w) \tag{8.10}$$

where $\Omega(x,u,w) = \text{H}(0, \frac{1}{2} L r_{\bar{p}}(x,u,w)^2)$ with $L \in \mathbb{R}^{n_x}$ representing the vector of Lipschitz constants of component functions f_i and

$$r_{\bar{p}}(x,u,w)^2 = \|x - \bar{x}\|_2^2 + \|u - \bar{u}\|_2^2 + \|w - \bar{w}\|_2^2. \tag{8.11}$$

Therefore, to enforce the condition (8.7), we can impose the following stronger condition

$$\forall x \in \xi \; \forall w \in \mathcal{W} : \{\tilde{f}(x, \kappa(x), w)\} \oplus \Omega(x, \kappa(x), w) \subseteq \xi^+. \tag{8.12}$$

In order to keep the future optimization problem tractable (convex), we use a common error bound corresponding to the "worst case" to impose the more conservative condition

$$\forall x \in \xi \; \forall w \in \mathcal{W} : \{\tilde{f}(x, \kappa(x), w)\} \oplus \mathcal{H}_r \subseteq \xi^+, \tag{8.13}$$

with $\mathcal{H}_r \coloneqq \mathrm{H}(0, \frac{1}{2} L r^2)$ and $r \coloneqq \max_{(x,w)\in\xi\times\mathcal{W}} \; r_{\bar{p}}(x, \kappa(x), w)$.

Condition (8.13) can be reformulated as

$$q(\xi) \oplus \mathbf{E}\mathcal{W} \oplus \mathcal{H}_r \subseteq \xi^+ \tag{8.14}$$

where $q(x) \coloneqq \tilde{f}(x, \kappa(x), \bar{w})$. This provides a geometric interpretation of the different sources of conservatism. The term $q(\xi)$ corresponds to the closed-loop image of ξ in the linearized model with nominal noise ($\bar{w}$), the second term $\mathbf{E}\mathcal{W}$ corresponds to the exogenous noise, and the third term $\mathcal{H}_r$ is responsible for the linearization error.

We first propose the following lemma, which provides sufficient conditions to design κ satisfying (8.7).

Lemma 8.4. *Let a system (8.3) with exogenous input set (8.5), a controller κ, two sets ξ and ξ^+ and a linearization point $\bar{p}$. If the following conditions are satisfied*

$$\xi \subseteq \mathrm{B}(\bar{x}, \sqrt{\delta_{\mathcal{X}}}), \tag{8.15}$$

$$\kappa(\xi) \subseteq \mathrm{B}(\bar{u}, \sqrt{\delta_{\mathcal{U}}}), \tag{8.16}$$

$$\forall x \in \xi \; \forall w \in \mathcal{W} : \{\tilde{f}(x, \kappa(w), w)\} \oplus \mathcal{H}_r \subseteq \xi^+, \tag{8.17}$$

with $r^2 = \delta_{\mathcal{X}} + \delta_{\mathcal{U}} + \delta_{\mathcal{W}}$ for some $\delta_{\mathcal{X}}, \delta_{\mathcal{U}} \geq 0$ and $\delta_{\mathcal{W}} \coloneqq \max_{w\in\mathcal{W}} \|w - \bar{w}\|_2^2$, then (8.7) holds.

Proof. The result follows directly from this sequence of implications

$$(8.15), (8.16), (8.17) \Rightarrow (8.13) \Rightarrow (8.12) \Rightarrow (8.7).$$

□

However, the converse result does not generally hold. The conser-

vatism arises from two sources. Firstly, we use the continuity property yielding the upper-bound (8.10). Secondly, we consider the worst linearization error for all points of ξ, which corresponds to the error of the farthest point from the linearization point as shown in (8.15) and (8.16).

The linearization point that minimizes the linearization error (r) is determined by $\bar{p}^* = \mathrm{CC}(\xi \times \kappa(\xi) \times \mathcal{W})$, the Chebyshev center of $\xi \times \kappa(\xi) \times \mathcal{W}$. However, given that the controller κ is part of the optimization process, we opt for the choice: $\bar{p} = \mathrm{CC}(\xi \times \mathcal{U}_1 \times \mathcal{W})$ which represents the optimal choice for minimizing the linearization error when κ is arbitrary.

8.3.2 Discretization and controller templates

We discretize the state space using ellipsoids, i.e.,

$$\xi = \mathrm{E}(c, \mathbf{P}),\ \xi^+ = \mathrm{E}(c_+, \mathbf{P}_+),$$

with $c, c_+ \in \mathbb{R}^{n_x}$ and $\mathbf{P}, \mathbf{P}_+ \in \mathbb{S}^n_{>0}$, and we consider affine controllers of the form

$$\kappa(x) = \mathbf{K}(x - c) + l, \tag{8.18}$$

where $\mathbf{K} \in \mathbb{R}^{n_u \times n_x}$ and $l \in \mathbb{R}^{n_u}$.

Therefore, in this setting, we obtain $\bar{x} = \mathrm{CC}(\xi) = c$, and, for the sake of clarity throughout the rest of the chapter, we assume that $\bar{w} = \mathrm{CC}(\mathcal{W}) = 0$.

The following theorem provides conditions based on Lemma 8.4 to guarantee the existence of a valid controller satisfying (8.7) and (8.8).

Theorem 8.5. *The system* (8.3) *with control input set* (8.4) *and exogenous input set* (8.5) *under the constrained affine control law* κ (8.18) *satisfies the condition that* $F_1(x, \kappa(x)) \subseteq \xi^+ = \mathrm{E}(c_+, \mathbf{P}_+)$ *for all* $x \in \xi = \mathrm{E}(c, \mathbf{P})$, *if, given the linearized system* (8.9) *around the point* $\bar{p} = (c, \bar{u}, 0)$ *with* $\bar{u} \in \mathcal{U}_1$, *there exist* $\mathbf{L} \in \mathbb{S}^{n_x}_{>0}, \mathbf{F} \in \mathbb{R}^{n_x \times n_u}$ *and scalars* $\delta_{\mathcal{X}} \geq 0, \delta_{\mathcal{U}} \geq 0, \phi \geq 0, \beta_{ij} \geq 0, \tau_k \geq 0$ *such*

that

$$\begin{pmatrix} \mathbf{I} & \mathbf{L} \\ \bullet & \delta_{\mathcal{X}}\mathbf{I} \end{pmatrix} \succeq \mathbf{0}, \tag{8.19}$$

$$\begin{pmatrix} \phi\mathbf{I} & 0 & \mathbf{F}^\top \\ \bullet & \delta_{\mathcal{U}} - \phi & (l - \bar{u})^\top \\ \bullet & \bullet & \mathbf{I} \end{pmatrix} \succeq \mathbf{0}, \tag{8.20}$$

$$\begin{pmatrix} \beta_{ij}\mathbf{I} & 0 & (\mathbf{AL} + \mathbf{BF})^\top \\ \bullet & 1 - \beta_{ij} & \mu^\top + V_i^\top + (\mathbf{E}w_j)^\top \\ \bullet & \bullet & \mathbf{P}_+^{-1} \end{pmatrix} \succeq \mathbf{0},$$

$$\textit{for } i \in [1; 2^{n_x}],\ j \in [1; N_w] \tag{8.21}$$

$$\begin{pmatrix} \tau_k\mathbf{I} & 0 & \mathbf{F}^\top\mathbf{U}_k^\top \\ \bullet & 1 - \tau_k & l^\top\mathbf{U}_k^\top \\ \bullet & \bullet & \mathbf{I} \end{pmatrix} \succeq \mathbf{0},\ \textit{for } k \in [1; N_u] \tag{8.22}$$

where $\mu = g + \mathbf{A}c + \mathbf{B}l - c_+$, V_i *for* $i \in [1; 2^{n_x}]$ *are the vertices of* $\mathrm{H}(0, \frac{1}{2}Lr^2)$ *with* $r^2 = \delta_{\mathcal{X}} + \delta_{\mathcal{U}} + \delta_{\mathcal{W}}$ *and* $\delta_{\mathcal{W}} = \max_{i \in [1; N_w]}\{\|w_i\|_2^2\}$. *From which we have*

$$\mathbf{P} = \mathbf{L}^{-2},\ \mathbf{K} = \mathbf{F}\mathbf{L}^{-1}.$$

Proof. Given Lemma 8.4, it is sufficient to establish the following equivalences

$$(8.15) \Leftrightarrow (8.19),\ (8.16) \Leftrightarrow (8.20),\ (8.17) \Leftrightarrow (8.21),\ (8.8) \Leftrightarrow (8.22).$$

The closed-loop dynamics can be expressed as

$$\tilde{f}(x, \kappa(x), w) = \bar{\mathbf{A}}x + \bar{b} + \mathbf{E}w$$

with

$$\bar{\mathbf{A}} = \mathbf{A} + \mathbf{BK},\ \bar{b} = g + \mathbf{B}l - \mathbf{BK}c.$$

We define the matrix $\mathbf{\Theta}$ as

$$\mathbf{\Theta} = \begin{pmatrix} \mathbf{L} & c & \mathbf{0} \\ \bullet & 1 & 0^\top \\ \bullet & \bullet & \mathbf{I} \end{pmatrix}.$$

Using respectively the S-procedure (Lemma A.2), the fact that $\mathbf{P} = \mathbf{L}^{-2}$

with $\mathbf{L} \succ \mathbf{0}$ and the Schur Complement Lemma (Lemma A.1), we have

$$(8.15) \Leftrightarrow \delta_{\mathcal{X}}\mathbf{P} - \mathbf{I} \succeq \mathbf{0} \Leftrightarrow \mathbf{I} - \delta_{\mathcal{X}}^{-1}\mathbf{L}^2 \succeq \mathbf{0} \Leftrightarrow (8.19).$$

By using the S-procedure, equation (8.16) can be expressed as the existence of $\phi \geq 0$ that satisfies the inequality

$$\phi \begin{pmatrix} \mathbf{P} & -\mathbf{P}c \\ \bullet & c^\top \mathbf{P} c - 1 \end{pmatrix} \succeq \begin{pmatrix} \mathbf{K}^\top \mathbf{K} & \mathbf{K}^\top (l - \bar{u} - \mathbf{K}c) \\ \bullet & (l - \bar{u} - \mathbf{K}c)^\top (l - \bar{u} - \mathbf{K}c) - \delta_{\mathcal{U}} \end{pmatrix}.$$

By further algebraic manipulations and using Lemma A.1, this inequality can be written as

$$\begin{pmatrix} \phi\mathbf{P} & -\phi\mathbf{P}c & \mathbf{K}^\top \\ \bullet & \phi(c^\top \mathbf{P} c - 1) + \delta_{\mathcal{U}} & (l - \bar{u} - \mathbf{K}c)^\top \\ \bullet & \bullet & \mathbf{I} \end{pmatrix} \succeq \mathbf{0}.$$

Then, by applying the congruent transformation of matrix $\mathbf{\Theta}$, i.e., multiplying the above inequality to the right by $\mathbf{\Theta}$ and to the left by $\mathbf{\Theta}^\top$, we obtain equation (8.20).

Since a polytope is contained in a convex set if and only if its vertices are contained in this set, equation (8.17) is satisfied $\forall w \in \mathcal{W}$ if and only if it is satisfied for the vertices of $\mathcal{W}$. Hence, (8.17) is equivalent to

$$\forall j \in [1; N_w]\ \forall x \in \xi : \{\tilde{f}(x, \kappa(x), w_j)\} \oplus \mathcal{H}_r \subseteq \xi^+$$

where w_j are the vertices of $\mathcal{W}$. By using the S-procedure, equation (8.17) can be expressed as the existence of $\beta_{ij} \geq 0$ that satisfies the inequality

$$\beta_{ij} \begin{pmatrix} \mathbf{P} & -\mathbf{P}c \\ \bullet & c^\top \mathbf{P} c - 1 \end{pmatrix} \succeq \begin{pmatrix} \bar{\mathbf{A}}^\top \mathbf{P}_+ \bar{\mathbf{A}} & \bar{\mathbf{A}}^\top \mathbf{P}_+ (\bar{b} + \mathbf{E}w_j + V_i - c_+) \\ \bullet & (\bar{b} + \mathbf{E}w_j + V_i - c_+)^\top \mathbf{P}_+ (\bar{b} + \mathbf{E}w_j + V_i - c_+) - 1 \end{pmatrix},$$

for $i \in [1; 2^{n_x}]$ and $j \in [1; N_w]$. By further algebraic manipulations and using the Schur Complement Lemma, this inequality can be written as

$$\begin{pmatrix} \beta_{ij}\mathbf{P} & -\beta_{ij}\mathbf{P}c & \bar{\mathbf{A}}^\top \\ \bullet & \beta_{ij}(c^\top \mathbf{P} c - 1) + 1 & (\bar{b} + \mathbf{E}w_j + V_i - c_+)^\top \\ \bullet & \bullet & \mathbf{P}_+^{-1} \end{pmatrix} \succeq \mathbf{0}.$$

Then, by applying the congruent transformation of matrix $\mathbf{\Theta}$, we obtain equation (8.21).

By using the S-procedure, equation (8.8) can be expressed as the existence

of $\tau_k \geq 0$ that satisfies the inequality

$$\tau_k \begin{pmatrix} \mathbf{P} & -\mathbf{P}c \\ \bullet & c^\top \mathbf{P} c - 1 \end{pmatrix} \succeq \begin{pmatrix} \mathbf{U}_k^\top \mathbf{K}^\top \mathbf{K} \mathbf{U}_k & \mathbf{K}^\top \mathbf{U}_k^\top \mathbf{U}_k (l - \mathbf{K}c) \\ \bullet & (l - \mathbf{K}c)^\top \mathbf{U}_k^\top \mathbf{U}_k (l - \mathbf{K}c) - 1 \end{pmatrix},$$

for $k \in [1; N_u]$. By further algebraic manipulations and using the Schur Complement Lemma, this inequality can be written as

$$\begin{pmatrix} \tau_k \mathbf{P} & -\tau_k \mathbf{P}c & \mathbf{K}^\top \mathbf{U}_k^\top \\ \bullet & \tau_k (c^\top \mathbf{P} c - 1) + 1 & (l - \mathbf{K}c)^\top \mathbf{U}_k^\top \\ \bullet & \bullet & \mathbf{I} \end{pmatrix} \succeq \mathbf{0}.$$

Finally, by applying the congruent transformation of matrix $\mathbf{\Theta}$, we obtain equation (8.22). □

The extension presented in Theorem 8.5 builds upon [ELJ22, Theorem 2] by incorporating the linearization error while maintaining convexity when considering the parameters of the initial ellipsoids ξ as optimization variables. It is worth noting that in [ELJ22, Theorem 2], the problem is no longer an LMI if we consider $\mathbf{P}$ as a variable. However, as demonstrated by Theorem 8.5, the optimization problem can be rendered convex through a congruent transformation when considering the square root of $\mathbf{P}^{-1}$ as the variable of interest. If f is an affine function, the LMIs from Theorem 8.5 can be simplified to the ones presented in [ELJ22, Theorem 2] by applying a congruent transformation.

We can visualize the terms of the Minkowski sum in (8.14) for our specific setting in Figure 8.3. As the diameter of ξ increases (i.e., $2\sqrt{\delta_{\mathcal{X}}}$), the half-lengths of $\mathcal{H}_r$ also increase. Consequently, the controller κ must become more aggressive to ensure that the diameter of $\widetilde{\xi}^+ := q(\xi)$, defined after (8.14), decreases. However, this might cause an increase in the magnitude of the control input, yielding a larger value of $\delta_{\mathcal{U}}$ (see Figure 8.4), which, in turn, amplifies the half-lengths of $\mathcal{H}_r$.

8.3.3 Objective function

Since our aim is not only to design κ to minimize a cost function, but also to optimize the shape $\mathbf{P}$ of the starting ellipsoid ξ, we introduce a performance objective that involves minimizing a cost function (8.6) and maximizing the hypervolume of the preceding ellipsoid ξ. We can maximize the volume of an ellipsoid $\xi = \mathrm{E}(c, \mathbf{P})$ by minimizing the convex function

$-\log(\det(\mathbf{L}))$ where $\mathbf{L}$ is the square root of $\mathbf{P}^{-1}$, i.e., $\mathbf{P}^{-1} = \mathbf{L}^2$ (Corollary A.4).

The correctness and efficiency of the global algorithm (Algorithm 8.9) are guaranteed by the following result.

Corollary 8.6. *For a given point c and target set $\xi^+ = \mathrm{E}(c_+, \mathbf{P}_+)$, with a transition cost function c_1 (8.6) where $\mathbf{Q} = \mathbf{S}^\top \mathbf{S}$ and $\lambda \in [0,1]$, the solution of the convex optimization problem*

$$\inf_{\substack{\mathbf{L}\succ\mathbf{0},\mathbf{F},l,\delta_\mathcal{X}\geq 0,\delta_\mathcal{U}\geq 0,\\ \phi\geq 0,\beta_{ij}\geq 0,\tau_k\geq 0,\gamma\geq 0,\widetilde{\mathcal{J}}}} \lambda\widetilde{\mathcal{J}} + (1-\lambda)(-\log(\det(\mathbf{L}))) \tag{8.23}$$

$$\text{s.t. (8.19), (8.20), (8.21), (8.22),} \quad \begin{pmatrix} \gamma\mathbf{I} & 0 & [\mathbf{L}, \mathbf{F}^\top, 0]\mathbf{S}^\top \\ \bullet & \widetilde{\mathcal{J}} - \gamma & [c^\top, l^\top, 1]\mathbf{S}^\top \\ \bullet & \bullet & \mathbf{I} \end{pmatrix} \succeq \mathbf{0} \tag{8.24}$$

satisfies

$$\widetilde{\mathcal{J}} \geq \max_{x\in\xi} c_1(x, \kappa(x)) \tag{8.25}$$

where the controller $\kappa(x) \coloneqq \mathbf{K}(x-c) + l \in \mathcal{U}_1$ with $\mathbf{K} = \mathbf{F}\mathbf{L}^{-1}$ ensures a transition from $\xi = \mathrm{E}(c, \mathbf{P})$ with $\mathbf{P} = \mathbf{L}^{-2}$ to ξ^+, as in Theorem 8.5.

Proof. Same as in [ELJ22, Corollary 1]. □

Note that, due to the linearization error (8.13), the inequality (8.25) is generally not tight. This contrasts with [ELJ22, Corollary 1], which was limited to affine dynamical systems.

The parameter λ governs the weight assigned to each criterion. Increasing its value will prioritize cost minimization, leading to a preference for reducing the volume of ξ. Conversely, decreasing λ will prioritize maximizing the volume of the starting ellipsoid ξ, requiring larger control gains while mapping all states of ξ into ξ^+. This also represents an exploitation/exploration trade-off, as maximizing the volume allows us to explore a larger portion of the state space, and minimizing the cost provides better (finer) solutions for already explored areas. Because of that, λ can be chosen adaptively throughout the execution.

A comprehensive study on how to fine-tune this meta-parameter is beyond the scope of this chapter. However, a basic approach is to start with

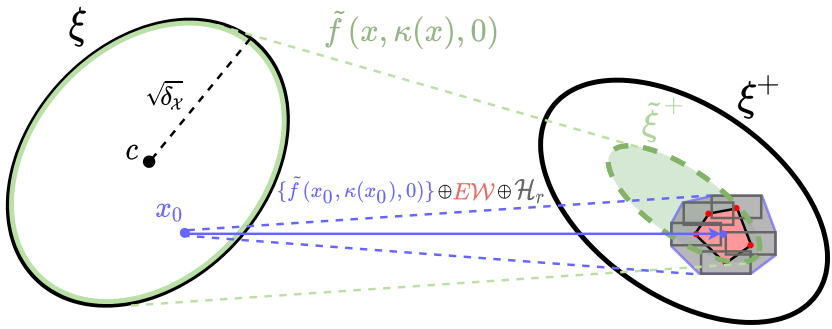


Fig. 8.3 Local transition from ellipsoid ξ to ellipsoid ξ^+. The affine controller κ enforces that $\forall x_0 \in \xi$ and $\forall w \in \mathcal{W}$: $f(x_0,\kappa(x_0),w) \in \{\tilde{f}(x_0,\kappa(x_0),0)\} \oplus \mathbf{E}\mathcal{W} \oplus \mathcal{H}_r \subseteq \xi^+$ where $\mathcal{W}$ is the polytopic noise and $\mathcal{H}_r = \mathrm{H}(0, \frac{1}{2}Lr^2)$ is the noise resulting from the linearization (8.13) where $r^2 = \delta_{\mathcal{X}} + \delta_{\mathcal{U}} + \delta_{\mathcal{W}}$. The set $\tilde{\xi}^+ := q(\xi)$ with $q(x) = \tilde{f}(x,\kappa(x),0)$ (see (8.14)) is an ellipsoid since an affine transformation of an ellipsoid is also an ellipsoid and the set $\mathbf{E}\mathcal{W}$ is a polytope since the linear transformation of a polytope is a polytope.

a smaller λ to encourage designing larger cells (potentially leading faster to a feasible controller), and then increase its value to favor smaller cells (with eventually lower transition costs).

8.4 Numerical experiments

8.4.1 One single transition

In this example we study aspects of determining a single transition for a given target set $\xi^+ = \mathrm{E}(c_+, \mathbf{P}_+)$ and initial point c with

$$c_+ = \begin{pmatrix} 4 \\ 4 \end{pmatrix}, \ \mathbf{P}_+ = \begin{pmatrix} 2 & 0.2 \\ 0.2 & 0.55 \end{pmatrix}, \ c = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Example 8.7. Consider the nonlinear system (8.3) given by

$$f(x,u,w) = \begin{pmatrix} 1.1x_1 - 0.2x_2 - \rho x_2^3 + u_1 + w_1 \\ 0.2x_1 + 1.1x_2 + \rho x_1^3 + u_2 + w_2 \end{pmatrix}, \tag{8.26}$$

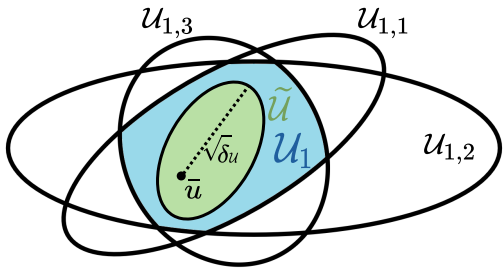


Fig. 8.4 Input space of the system $\mathcal{S}_1$ (blue) and the set of inputs actually used by the controller κ on the ellipsoid ξ (green). The input space is denoted by $\mathcal{U}_1 = \mathcal{U}_{1,1} \cap \mathcal{U}_{1,2} \cap \mathcal{U}_{1,3}$ where $\mathcal{U}_{1,1}, \mathcal{U}_{1,2}, \mathcal{U}_{1,3}$ are ellipsoids, and $\widetilde{\mathcal{U}} := \kappa(\xi)$ represents the set of inputs used by κ on ξ. The controller κ is designed such that $\widetilde{\mathcal{U}} \subseteq \mathcal{U}_1$ (8.8).

where $\rho \geq 0$. The control input u is constrained by the set $\mathcal{U}_1 = \mathcal{U}_{1,1} \cap \mathcal{U}_{1,2} \cap \mathcal{U}_{1,3}$ with

$$\mathcal{U}_{1,1} = \mathrm{H}\left(0, \begin{pmatrix} 4 \\ 5 \end{pmatrix}\right), \mathcal{U}_{1,2} = \mathrm{B}(0,5), \mathcal{U}_{1,3} = \mathrm{E}\left(0, \begin{pmatrix} 0.05 & 0 \\ 0 & 0.033 \end{pmatrix}\right),$$

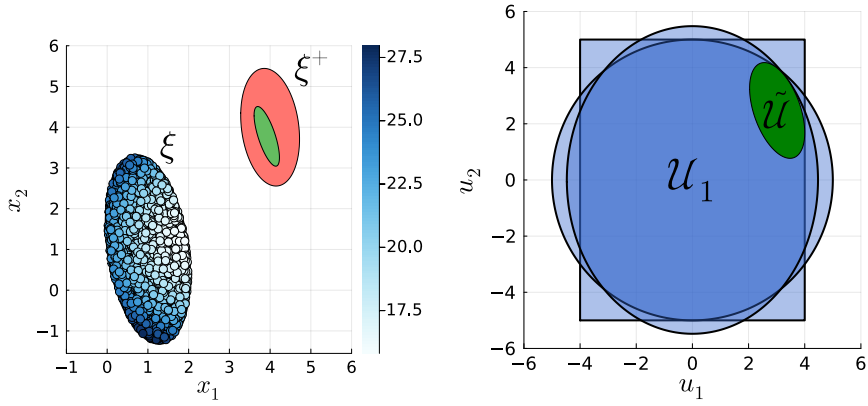
and the exogenous input w by the set $\mathcal{W} = \mathrm{H}(0, (\omega_{\max}, \ \omega_{\max})^\top)$ with $\omega_{\max} \geq 0$. △

The level of non-linearity in the system is determined by the parameter ρ, where $\rho = 0$ corresponds to an affine dynamical system. On the other hand, the presence of noise is controlled by the parameter $\omega_{\max}$, where $\omega_{\max} = 0$ corresponds to a deterministic system.

For several different values of ρ, $\omega_{\max}$ and λ, we solved the optimization problem of Corollary 8.6 considering the quadratic cost function (8.6) with $\mathbf{Q} = \mathrm{diag}(\mathbf{I}, \ \mathbf{I}, \ 1)$, i.e., $c_1(x,u) = x^\top x + u^\top u + 1$.

On average, each solution to Corollary 8.6 was found in 0.0176 seconds on an Intel® Core™ i7-10610U CPU 1.80 GHz ×8 with 16 GB of memory and using the Julia JuMP [DHL17] interface with the Mosek solver on Windows 10.

Firstly, observing the left-hand side of Figure 8.5, we can deduce that while $\widetilde{\mathcal{J}}$ is an upper bound for the cost of the transition from ξ to ξ^+, in practice this cost is considerably lower for a significant part of the cell. As expected, by comparing Figure 8.5 and Figure 8.6, the worst case cost $\widetilde{\mathcal{J}}$



(a) State space. **(b)** Input space.

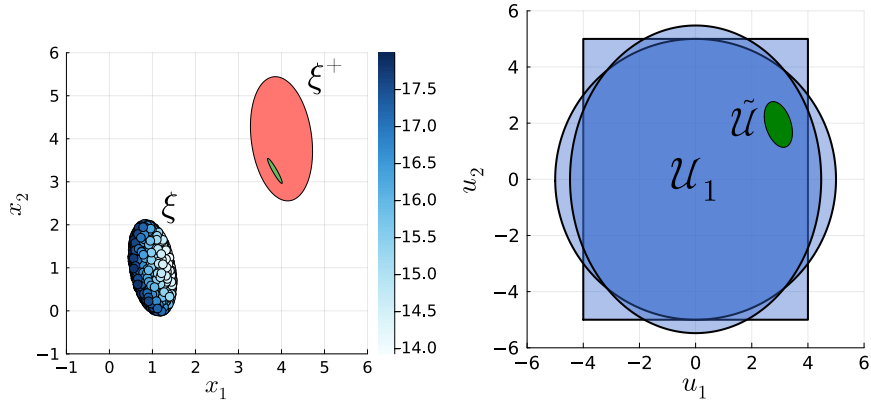
Fig. 8.5 (Section 8.4.1) Solution provided by Corollary 8.6 for $\omega_{\max} = 0.1$, $\rho = 0.0005$ and $\lambda = 0.01$. Left: Value of the cost function $c_1(x, \kappa(x))$ for the closed-loop system (color map) and the ellipsoids ξ^+ (red) and $\tilde{\xi}^+$ (green). Right: The sets $\mathcal{U}_{1,1}$, $\mathcal{U}_{1,2}$, and $\mathcal{U}_{1,3}$ (blue), and their intersection $\mathcal{U}_1$ (dark blue). The set $\tilde{\mathcal{U}} := \kappa(\xi)$ (green) represents the inputs actually employed by the controller κ on ξ. The upper bound on the transition cost $\tilde{\mathcal{J}} = 28.1$ and the volume of the starting ellipsoid $\mathrm{vol}(\xi) = 6.65$.

of the controller decreases as λ increases. This is achieved by reducing both the size of the initial ellipsoid ξ and the maximum magnitude of the inputs actually used by the controller κ on ξ. Finally, comparing the results of Figure 8.5 and Figure 8.7, we notice that when we increase the nonlinearity (ρ) and the noise bound ($\omega_{\max}$), the controller has to produce a smaller ellipsoid $\tilde{\xi}^+$ more centered in ξ^+, and to do so, it must reduce the size of the initial ellipsoid.

Figure 8.8 illustrates that as the values of $\omega_{\max}$ and ρ increase, the Pareto front of the bi-objective function (8.23) consistently exhibits worse outcomes, i.e., higher costs and smaller volumes of the initial ellipsoid, across all values of the weighting parameter λ.

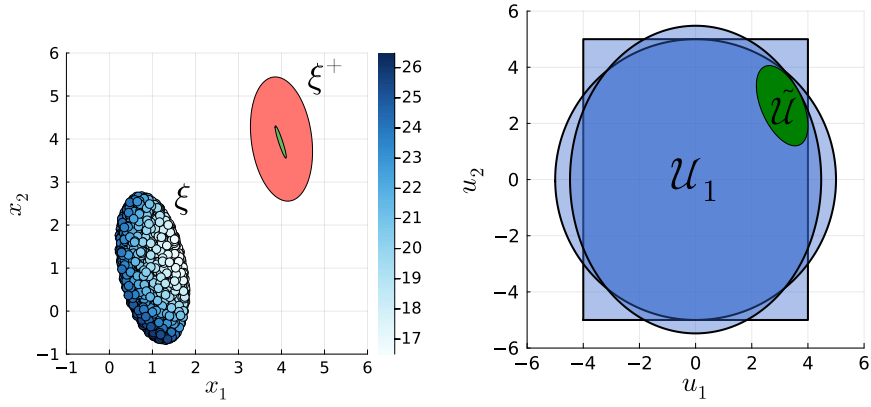
8.4.2 Optimal control

In this example, we apply Algorithm 8.9 to the optimal control of the two-dimensional dynamical system $\mathcal{S}_1$ introduced in Example 8.7. We use the parameters $\rho = 0.005$, $\omega_{\max} = 0.1$, and the same transition cost function,



(a) State space. **(b)** Input space.

Fig. 8.6 (Section 8.4.1) Solutions provided by Corollary 8.6 for $\omega_{\max} = 0.1$, $\rho = 0.0005$ and $\lambda = 0.3$. We have $\widetilde{\mathcal{J}} = 18$, $\mathrm{vol}(\xi) = 1.44$.



(a) State space. **(b)** Input space.

Fig. 8.7 (Section 8.4.1) Solutions provided by Corollary 8.6 for $\omega_{\max} = 0.15$, $\rho = 0.001$ and $\lambda = 0.01$. We have $\widetilde{\mathcal{J}} = 26.56$, $\mathrm{vol}(\xi) = 3.82$.

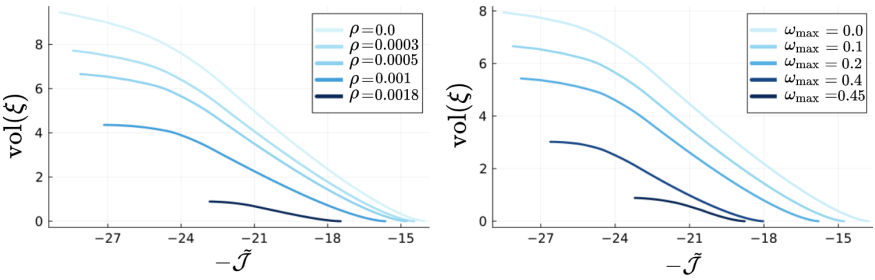


Fig. 8.8 (Section 8.4.1) Pareto front of the bi-objective function (8.23) for different values of ρ and $\omega_{\max}$. Each curve decreases as the value of λ increases from 0 to 1. Left: $\omega_{\max} = 0.1$. Right: $\rho = 0.0005$.

which penalizes states and inputs that are far from the origin.

A first feasible solution $\mathcal{S}_2$ (see Figure 8.9) was found after 18 seconds with only 8 cells and 7 transitions created, in the same computational setup as in the previous example. We can continue to expand the tree to explore other paths in the state-space as illustrated with abstraction $\mathcal{S}_2'$ in Figure 8.9 (right). As anticipated, the controller exhibits a preference for solutions that pass near the origin while circumventing the obstacle.

Let l_1 and l_1' be the superoptimal value functions derived from the superoptimal value functions l_2 and l_2' of $\mathcal{S}_2$ and $\mathcal{S}_2'$, respectively (Theorem 6.2).

Given $x_0 = (-10, -10) \in \mathcal{I}_1$, the guaranteed total cost by $\mathcal{S}_2$ is $l_1(x_0) = 1732$ whereas the true total cost of this specific trajectory is 1337. For $\mathcal{S}_2'$, the guaranteed total cost is $l_1(x_0) = 992$ whereas the true total cost is 875. The control cost guaranteed by $\mathcal{S}_2'$ is better than by $\mathcal{S}_2$ for two main reasons: 1) we continued to improve the abstraction using the RRT* variant, and 2) because by considering smaller cells, the cost of transitioning from one cell to another in the worst case is lower. Nevertheless, a similar part of the state space is covered by both abstractions, which is achieved with far fewer cells for $\mathcal{S}_2$.

8.5 Summary

We provided a tractable algorithm for the optimal control of L-smooth nonlinear dynamical systems. Firstly, the proposed algorithm circumvents the

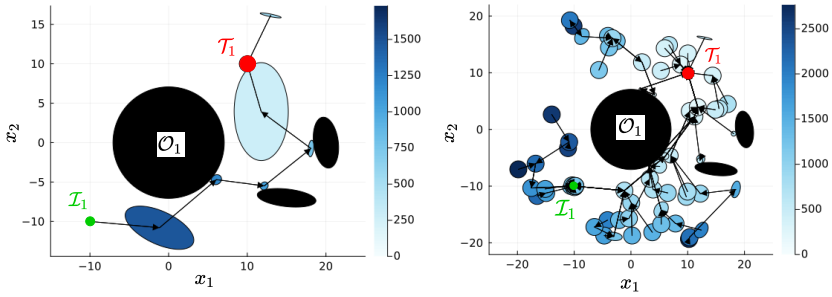


Fig. 8.9 (Section 8.4) Solution obtained from Algorithm 8.9 (left) and from the improved RRT* version (right). The initial set $\mathcal{I}_1$, target set $\mathcal{T}_1$ and obstacles $\mathcal{O}_1$ are respectively in green, red and black. The color map illustrates the value function l_1 for $\mathcal{S}_1$. Left: Construction of $\mathcal{S}_2$ and l_2 stops as soon as a feasible solution is found. Right: Continue to extend and improve the abstraction $\mathcal{S}_2'$ and the value function l_2'. We added a bound on the cell volume to easily visualize the entire tree.

necessity of discretizing the input space by employing a set of local feedback controllers. This is done to ensure deterministic transitions, thereby eliminating the non-determinism associated with abstraction, a common limitation in classical approaches. Secondly, the combined use of ellipsoid-based covering and affine local controllers leverages the power of LMIs and convex optimization, allowing the creation of larger and non-standard cells. Thirdly, the use of a lazy approach and non-uniform cells significantly reduces the complexity of the abstraction, i.e., the number of abstract states, when addressing a specific control problem.

9

Dionysos.jl: a modular platform for smart symbolic control

THE practical implementation of abstraction-based controllers has gained traction through several open-source toolboxes [MJDT10, RTM11, RZ16], including recent tools for stochastic systems analysis and synthesis [MLL24, WL24]. These efforts underscore the growing need in industry for techniques that offer robust safety guarantees. However, abstraction suffers from the *curse of dimensionality*, and none of these toolboxes can solve non-academic problems of dimension, say, larger than 3. These limitations of scale explain why abstraction-based techniques have not yet been successful, particularly in the field of robotics.

In this chapter, we introduce `Dionysos.jl`[1] [CBLJ24a], a modular package for solving optimal control problems for complex dynamical systems using state-of-the-art and experimental techniques from symbolic control, optimization, and learning. It is built on top of different Julia packages such as `JuMP.jl` and `MathOptInterface.jl`, and features optimal control problem definitions and several abstraction-based methods to solve them.

[1]The package is open source and available on GitHub at: `https://github.com/dionysos-dev/Dionysos.jl`.

`Dionysos.jl` is the software of the ERC project Learning to control (L2C)[2]. It implements new fundamental techniques that have been designed for smart abstraction [CLEJ21, LBJ21, ELJ22, BRAJ23, CJ23] as part of the L2C project, with the goal of designing control techniques that would come with guarantees of safety and efficiency, while at the same time being able to take into account non-standard constraints or information, e.g. coming from first principles in physics, a logical specification translating some legal regulations, or some recommendation from a human being.

In recent years, several groups have introduced ideas to mitigate the computational challenges of the abstraction-based approach, such as the concept of *lazy abstractions* [CGG11a, GGM15, TI09, HMMS18b]. The package `Dionysos.jl` offers a versatile platform for implementing these non-standard approaches within a unified environment. Notably, all algorithms presented in this thesis are implemented within `Dionysos.jl` (see Table 9.1 and Table 9.2). As a result, `Dionysos.jl` features

—abstractions that are covers of the concrete space (i.e., not only partitions);

—discretization templates, such as hyperrectangles and ellipsoids;

—controller templates, including piecewise constant controllers or piecewise affine controllers;

—different and novel types of abstraction relations such as alternating simulation relation (ASR), feedback refinement relation (FRR), or feedforward abstraction relation (FAR).

The techniques and software solutions presented have been validated on academic examples from [GPT09, MGG13, RWR16, GLB14], which are available in the toolbox documentation.

This chapter is organized as follows: Section 9.1 summarizes the key concepts for constructing smart abstractions. Section 9.2 details the features of `Dionysos.jl`. The package structure and its main modules are discussed in Section 9.3. Finally, Section 9.4 and Section 9.5 present numerical examples and benchmark comparisons with existing toolboxes, respectively.

This work has been published in [CBLJ24a, CBLJ24b].

[2]European Reasearch Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement No 864017 - L2C.

9.1 Smart abstraction

`Dionysos.jl` extends the classical approach (Section 3.2) by enabling lazy construction of abstractions (Chapter 7), and by utilizing overlapping cells and state-dependent controllers defined in a piecewise manner (Chapter 8). The design of low-level controllers within cells, in combination with high-level abstraction-based controllers, opens up new possibilities when co-creating the abstraction and the controller.

By designing these local state-dependent controllers, for example by solving an optimization problem (e.g., Section 8.3), we can ensure deterministic transitions in the abstraction, thereby eliminating the nondeterminism imposed by the discretization of the concrete system (see Figure 3.7). In addition, contrary to the classical approach (Section 3.2.2), this technique avoids discretizing the input-space and uses all the available inputs, enabling the design of more effective control solutions based on a given cost metric.

To reduce the number of cells in the abstraction, `Dionysos.jl` co-designs the abstraction and controller by optimizing the shape of the cells during the construction of the abstraction. For example, the combined use of ellipsoid-based covering and affine local feedback controllers can leverage the power of linear matrix inequalities (or LMI) and convex optimization to create larger/non-standard cells (see Figure 8.1).

9.2 `Dionysos.jl` features

In this section, we present the features currently supported by `Dionysos.jl`.

Systems: `Dionysos.jl` supports simple systems with bounded disturbances as described in (8.3) and returns static and dynamical controllers (see Definition 2.4) for both the abstract and concrete problems.

Specifications: `Dionysos.jl` supports either *reach-avoid* or *invariance* specifications (Definition 2.10). In addition to the specified specifications, given a simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$, a *state cost function* $s : \mathcal{X} \to \mathbb{R}_{\geq 0}$ evaluating the cost of being in a state x, and a *transition cost function* $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{\geq 0}$ quantifying the cost of transitioning from one state x to another with con-

trol input u, can be supplied to the control problem. The objective is then to design a controller that satisfies the specification while minimizing the cumulative cost.

Discretization Templates: The *quantizer* $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$, which defines the relation between the concrete system $\mathcal{S}_1$ and the abstract system $\mathcal{S}_2$, can be either a (partial or complete) partition or a cover of the concrete state space. For a continuous state space $\mathcal{X}_1 \subseteq \mathbb{R}^n$, `Dionysos.jl` provides support for two types of sets for state-space discretization

—*Hyperrectangle* (1.3);
—*Ellipsoid* (1.4).

9.3 Package structure

In this section we describe the architecture of `Dionysos.jl`. It is composed of seven root modules. This section will cover the first three modules—`System`, `Problem`, and `Optim`—along with a brief overview of the `Utils` module. For the sake of conciseness, the three other principal modules, namely `Domain`, `Mapping` and `Symbolic`, are skipped but can be found in the package documentation. A summary of this section is given in Figure 9.1.

9.3.1 The `System` module

The `System` module contains the mathematical descriptions of simple systems, controllers and trajectories (see Definition 2.4). It is an extension of `MathematicalSystems.jl` and `HybridSystems.jl` [LFS+24], it complements the latter with more specific system definitions.

The systems are described as structures implementing the abstract type `ControlSystem{N, T}`, where `N` and `T` are respectively the dimension and type of the state-space (e.g. `N = 3` and `T = Float64` for a three-dimensional continuous state-space). For example,

```
ControlSystemLinearized{N, T, F1, F2, F3, F4} <:  ControlSystem{N,T}
```

where `F1`, `F2`, `F3` and `F4` are subtypes of `Function`, implements a control

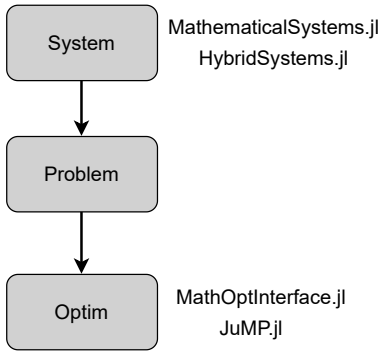


Fig. 9.1 Summary of Section 9.3. The `System` module is an extension of `MathematicalSystems.jl` and `HybridSystems.jl`, and implements mathematical definitions of dynamical systems. The `Problem` module contains mathematical definitions of control problems. All problem structures have a system as a field. The `Optim` module contains the control strategies to solve the problems. It is built on top of the optimization packages `MathOptInterface.jl` and `JuMP.jl`.

system whose transition function has been linearized. It has the form

$$\dot{x}(t) \in \widetilde{F}(x, u), \tag{9.1}$$

where $\widetilde{F}(x, u)$ is such as in (8.3) with an additive noise, that is

$$\widetilde{F}(x, u) = \left\{ \widetilde{f}(x, u) + w \,|\, w \in \mathcal{W} \right\}, \tag{9.2}$$

where $\mathcal{W} = [-W, W]^{n_x}$, and where $\widetilde{f}$ is the linearized version of some possibly nonlinear function f. The system is considered to be sampled with a given time-step, and the corresponding discrete-time transition function is computed with a numerical derivation scheme such as the fourth-order Runge Kutta method, implemented as `RungeKutta4` in `Dionsysos.jl`. The structure `ControlSystemLinearized{N,T,F1,F2,F3,F4}` contains the fields

—`tstep::Float64`, the sampling time-step;
—`measnoise::SVector{N,T}`, the bound W on the disturbance;
—`sys_map::F1`, the sampled possibly nonlinear transition function;
—`linsys_map::F2`, the sampled linearized transition function;
—`error_map::F3`, the difference between `linsys_map` and `sys_map`;
—`sys_inv_map::F4`, the inverse of `sys_map`.

The controllers implement the abstract type `Controller` and have a field `c_eval` that corresponds to its the set-valued output map $H_{\mathcal{C}}(x_{\mathcal{C}}, x)$. For example, the structure `ConstantController{T,VT}` implements the static controllers of the form $H_{\mathcal{C}}(0, x) = \{c\}$, where c is a constant. It contains the fields `c::VT` that is the constant c, where `VT<:AbstractVector{T}` and `T<:Real`, and `c_eval`.

Finally, the module `System` contains descriptions of trajectories. For example, the structure

```
ContinuousTrajectory{T, XVT<:AbstractVector{T}, UVT<:AbstractVector{T}},
```

contains the fields `x::Vector{XVT}` and `u::Vector{UVT}`, and implements closed-loop trajectories of the form $(\boldsymbol{u}, \boldsymbol{x}) \in \mathcal{B}(\mathcal{C} \times \mathcal{S})$ such as described in Definition 2.6.

9.3.2 The `Problem` module

The `Problem` module contains the two structures that respectively define the reach-avoid and invariance specification problems in `Dionysos.jl`. Both implement the abstract type `ProblemType`.

`OptimalControlProblem{S,XI,XT,XC,TC,T<:Real}` is the first structure, and implements a reach-avoid problem $\Sigma^{\text{Reach}} = [\mathcal{I}, \mathcal{T}, \mathcal{O}]$. Its fields are

—`system::S`, the system to be controlled,
—`initial_set::XI`, the initial set $\mathcal{I}$,
—`target_set::XT`, the target set $\mathcal{T}$,
—`state_cost::XC`, the state cost function,
—`transition_cost::TC`, the transition cost function, and
—`time::T`, the number of allowed steps.

Note that, in `Dionysos.jl`, the obstacles are encoded as part of the domain of the system. The second structure is `SafetyProblem{S,XI,XS,T<:Real}`, and implements an invariance problem. Its fields are

—`system::S`, the system to be controlled,
—`initial_set::XI`, the initial set $\mathcal{I}$,
—`safe_set::XS`, the safe set $\mathcal{Z}$,
—`time::T`, the number of allowed steps.

For both structures, the type `S` is typically a system type from the packages `MathematicalSystems.jl` and `HybridSystems.jl` or from the `System` module.

9.3.3 The `Optim` module

The `Optim` module contains both abstraction-based and classical control strategies. Table 9.1 gathers the implemented strategies. All strategies are

Module	Algorithm	Reference
`BemporadMorari`	-	[BM99]
`BranchAndBound`	-	[LBJ21]
`AB.EllipsoidsAbstraction`	-	[ELJ22]
`AB.UniformGridAbstraction`	Algorithm 3.1	[RWR16]
`AB.LazyAbstraction`	Algorithm 7.5	[CLEJ21]
`AB.HierarchicalAbstraction`	Algorithm 7.6	[CLEJ21]
`AB.LazyEllipsoidsAbstraction`	Algorithm 8.9	[CEJ24]

Table 9.1 Modules, algorithms and corresponding references of all the control strategies implemented in `Dionysos.jl`, where `AB = Abstraction` in the codebase.

viewed as *solvers* inheriting from `JuMP.jl` [LDD+23], a powerful optimization framework embedded in Julia. More precisely, they are all subtypes of the abstract type `MOI.AbstractOptimizer`, and implement the function `MOI.optimize!` from the package `MathOptInterface.jl` [LDGL22].

A simple example of an implementation of a classical abstraction-based method is given in the following. In Code 9.2, the steps given in Figure 1 are followed. In Code 9.1, the structure for the optimizer is defined. To be initialized, it needs a concrete problem to solve, as well as a discretization for the state space and the input space.

```
mutable struct Optimizer{T}<:MOI.AbstractOptimizer
    concrete_problem::Union{
        Nothing,
        PR.OptimalControlProblem,
        PR.SafetyProblem
    }
    abstract_problem::Union{
        Nothing,
        PR.OptimalControlProblem,
        PR.SafetyProblem
    }
    abstract_controller::Any
    concrete_controller::Any
    state_grid::Any
    input_grid::Any
    function Optimizer{T}(cp,sg,ig) where {T}
        return new{T}(
            cp,
```

```
                nothing,
                nothing,
                nothing,
                nothing,
                sg,
                ig
            )
    end
    end
```

Code 9.1: Definition of an abstraction-based strategy structure implementing the abstract type `MOI.AbstractOptimizer`.

```
function MOI.optimize!(opt::Optimizer)
    # Build the abstraction
    abstract_system = build_abs_system(
        opt.concrete_problem.system,
        opt.state_grid,
        opt.input_grid,
    )
    # Build the abstract problem
    abstract_problem = build_abs_problem(
        opt.concrete_problem,
        abstract_system
    )
    opt.abstract_problem = abstract_problem
    # Solve the abstract problem
    abstract_controller = solve_abs_problem(
        abstract_problem
    )
    opt.abstract_controller = abstract_controller
    # Solve the concrete problem
    opt.concrete_controller = solve_conc_problem(
        abstract_system,
        abstract_controller
    )
end
```

Code 9.2: Definition of the `MOI.optimize!` function for the strategy `Optimizer` defined in Code 9.1.

9.3.4 The `Utils` module

This module contains useful functions, data structures, classical search algorithms, file management tools, plotting utilities, and more. Notably, the algorithms for ellipsoid inclusion tests introduced in Appendix C are implemented within the `Utils` module (see Table 9.2).

Module	Algorithm	Reference
`Base.in(E1::Ellipsoid, E2::Ellipsoid)`	Algorithms C.1 and C.2	[CEJ23]
`UT.scale_for_inclusion_contact_point`	Corollary C.8	[CEJ23]

Table 9.2 Modules, algorithms, and corresponding references for the ellipsoid-related subroutines introduced in this thesis and implemented in `Dionysos.jl`, where `UT = Utils` in the codebase.

9.4 Numerical example

In this section, we provide an example of how `Dionysos.jl` is used to control a nonlinear system with a smart abstraction method. We first define the problem, then solve it with `Dionysos.jl`, and finally we provide visualization results, as recipes are implemented for all visualizable structures[3]. For the sake of conciseness, the code presented in this section is slightly simplified[4].

Example 9.1. We consider a reach-avoid problem $\Sigma_1^{\text{Reach}} = [\mathcal{I}_1, \mathcal{T}_1, \mathcal{O}_1]$ where $\mathcal{I}_1 = \text{E}((-10,-10), 10\mathbf{I}_2)$, $\mathcal{T}_1 = \text{E}((10,10), \mathbf{I}_2)$, and $\mathcal{O}_1 = \text{E}(0, 0.02\mathbf{I}_2)$. The dynamical system under study is $\mathcal{S}_1 = (\mathcal{X}_1 \setminus \mathcal{O}_1, \mathcal{U}_1, F_1)$, with the state space $\mathcal{X}_1 = [-20, 20]^2$, the control input space $\mathcal{U}_1 = [-10, 10]^2$, and the dynamics given by $F_1(x,u) = \{f(x,u,w) \mid w \in \mathcal{W}\}$, where the function f is defined in Example 8.7 with parameters $\rho = 0.005$ and $\omega_{\max} = 0$. The transition cost function is $c_1(x_1, u_1) = x_1^\top x_1 + u_1^\top u_1 + 1$, and there is no additional state cost. △

The system in Example 9.1, as well as other examples, are already defined in `Dionysos.jl/problems`. The `OptimalControlProblem` defining the problem in Example 9.1 can be found in the `NonLinear` module in `problems/non_linear.jl`. In Code 9.3, we import this problem.

```
concrete_problem = NonLinear.problem()
concrete_system = concrete_problem.system
```

Code 9.3: The problem stated in Example 9.1 is imported from the `problems` directory.

[3] See https://docs.juliaplots.org/latest/recipes/ for an introduction on recipes.
[4] See https://dionysos-dev.github.io/Dionysos.jl/stable/generated/Lazy-Ellipsoids-Abstraction/ for the full example.

We choose to use `AB.LazyEllispoidsAbstraction` (see Table 9.1) to solve this problem, a smart abstraction method that constructs a conrollable abstraction relation (FAR) that partially covers the state space with ellipsoids. In Code 9.4, we instantiate the corresponding `Optimizer`, then set all the needed fields, including `concrete_problem`, the reach-avoid problem stated above. For the sake of conciseness, we do not state the purpose of the other fields, but gather them in the variable `other_parameters` in the code.

```
optimizer = MOI.instantiate(
    AB.LazyEllipsoidsAbstraction.Optimizer
)
AB.LazyEllipsoidsAbstraction.set_optimizer!(
    optimizer,
    concrete_problem,
    other_parameters...
)
```

Code 9.4: The `Optimizer` is instantiated, and the solver parameters are set.

In Code 9.5, we solve the problem and retrieve the corresponding abstract system, abstract problem and concrete controller (see Figure 1). Note that every operation follows the `MathOptInterface.jl` syntax.

```
MOI.optimize!(optimizer)
abstract_system = MOI.get(
    optimizer,
    MOI.RawOptimizerAttribute("abstract_system")
)
abstract_problem = MOI.get(
    optimizer,
    MOI.RawOptimizerAttribute("abstract_problem")
    )
abstract_controller = MOI.get(
    optimizer,
    MOI.RawOptimizerAttribute("abstract_controller")
)
concrete_controller = MOI.get(
    optimizer,
    MOI.RawOptimizerAttribute("concrete_controller")
)
```

Code 9.5: The problem is solved, and the result is extracted from the solver.

In Code 9.6, we simulate a concrete closed-loop trajectory starting from an initial point x_0 sampled from the initial set $\mathcal{I}_1$.

```
x0 = UT.sample(concrete_problem.initial_set)
reached(x) = x \in concrete_problem.target_set
cost_control_trajectory = ST.get_closed_loop_trajectory(
    concrete_system,
    concrete_controller,
    concrete_problem.transition_cost,
    x0;
    stopping = reached,
)
```

Code 9.6: Simulate a closed-loop trajectory.

In `Dionysos.jl`, we can also generate visualizations thanks to the implemented recipes. In Code 9.7, we visualize the constructed abstract system, and it gives the plot in Figure 9.2.

```
fig = plot(aspect_ratio=:equal)
plot!(abstract_system;
      arrowsB = true,
      cost = true)
plot!(concrete_problem.target_set;
      color = :red)
plot!(cost_control_trajectory;
      color = :blue)
plot!(concrete_problem.initial_set;
      color = :green)
```

Code 9.7: The recipe implemented for the abstraction is used to visualize the abstract system. The `cost` is set to true to plot the superoptimal value function (see Algorithm 8.9).

9.5 Benchmarking

In order to evaluate the performance of `Dionysos.jl`, we compare the performance of our package against other similar packages, namely SCOTS [RZ16] and CoSyMA [MGG13]. We exclude PESSOA [MJDT10, RTM11] from the comparison as it is outperformed by SCOTS [RZ16]. The code for comparing these packages is published on CodeOcean [CBLJ24b], and is

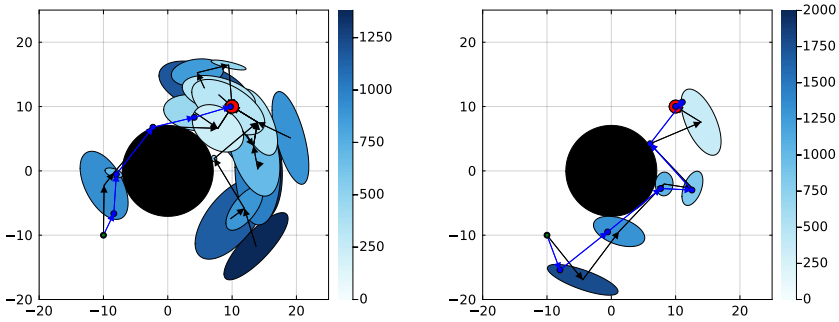


Fig. 9.2 Results of Code 9.7 for different meta-parameters (`other_parameters`) of the solver `AB.LazyEllipsoidsAbstraction`. The initial set $\mathcal{I}_1$, target set $\mathcal{T}_1$, and obstacle set $\mathcal{O}_1$ are shown in green, red, and black, respectively. The superoptimal value function is depicted using a blue color map, with the trajectory of the closed-loop system also plotted in blue.

entirely reproducible. For more information, we invite the reader to read the `README.md` file in the CodeOcean capsule.

We reproduced the two numerical experiments of [RZ16]. First, the DC-DC converter example (Example 3.14) presented in [RZ16, Section 4.2] is reproduced with `Dionysos.jl`, SCOTS and CoSyMA. Thanks to the modularity of `Dionysos.jl`, we can specify to the package that the system is incrementally stable, resulting in sped-up abstraction and synthesis procedures [CGG11b]. For the sake of completeness, we also provide the performance of `Dionysos.jl` in the setting where no prior knowledge on the stability is given. The results can be found in Table 9.3.

	Abstraction [s]	Synthesis [s]	Total [s]
`Dionysos.jl` (no prior)	**1.24**	**3.53**	**4.77**
`Dionysos.jl` (prior)	**0.63**	**2.76**	**3.39**
SCOTS	19.05	74.01	93.06
CoSyMA	—	—	5.31

Table 9.3 Comparaison between SCOTS, CoSyMA and `Dionysos.jl` (`AB.UniformGridAbstraction` solver from Table 9.1) for the DC-DC converter example. `Dionysos.jl` outperforms SCOTS and CoSyMA with and without prior knowledge of the system's incrementally stable property.

Second, the path planning problem (Example 3.13) presented in [RZ16, Section 4.1] is executed with `Dionysos.jl` and SCOTS. We exclude CoSyMA because this system is not incrementally stable. The results can be found in Table 9.4.

	Abstraction [s]	Synthesis [s]	Total [s]
`Dionysos.jl`	**8.58**	**6.45**	**15.03**
SCOTS	117.52	480.44	597.96

Table 9.4 Comparaison between SCOTS, CoSyMA and `Dionysos.jl` (`AB.UniformGridAbstraction` solver from Table 9.1) for the path planning example. `Dionysos.jl` outperforms SCOTS.

We see that `Dionysos.jl` outperforms the other packages for these two examples. We also reproduced [RZ16, Figures 3 and 4] in Figure 3.6a and Figure 3.6b, which shows that they compute the same controller. The visualizations of the DC-DC converter controller with and without prior knwoledge are identical, as it can be verified in [CBLJ24b].

The reason for such a difference between SCOTS, written in C++, and `Dionysos.jl` does not lie in the programming language used to write the package but in the synthesis algorithm itself. For example, unlike SCOTS, our package does not make use of *Binary Decision Diagrams* (*BDDs*) [Bry92], which as recognized in [RZ16] results in substantially longer execution times compared to tools that use alternative data structures.

9.6 Summary

In this chapter, we introduced `Dionysos.jl`, a new software package that provides both a new abstract symbolic representation of the system and controllers with safety guarantees. It generalizes existing toolboxes limited to classical abstractions by allowing the construction of abstractions with a simple concretization step and the design of low-level controllers. The package provides a modular platform for implementing these non-standard approaches in a unified environment, and facilitates seamless interaction for end-users with the advanced black-box algorithms developed in the L2C project and integrated into the toolbox. This interface allows end-users to guide the algorithms with control-specific insights, such as heuristics, ensuring more tailored and effective solutions.

We provided a description of the structure of the package, and described further the main modules of it. We then showed how `Dionysos.jl` can be used in practice by providing an optimal control problem example. Finally, we demonstrated the performance of our package compared to existing similar toolboxes.

Conclusions and perspectives

THROUGHOUT this thesis, we have pursued two main objectives. Firstly, we have developed novel theoretical frameworks for characterizing system relations. And secondly, we have introduced practical algorithms that address classical challenges of abstraction-based approach, such as non-determinism and computational complexity. Additionally, we contributed to the creation of software tools that enable the practical application of these methods in real-world scenarios.

⋆ Brief summary of the contributions

In what follows, we briefly summarize our contributions throughout this thesis.

Part II: Theory

Our contributions in this part focus on the characterization of abstraction-based control techniques. In particular, this is done through a systematic approach to tailoring system relations based on the desired properties of the concretization step and the final concrete controller. Furthermore, this characterization is carried out independently of the synthesis of the abstract controller.

Key contributions include the exhaustive enumeration and classification of system relations that refine alternating simulation relations, with each evaluated for its advantages and limitations. This work demonstrates

that when a concrete system is related to its abstraction via any of these relations, the abstract controller can be applied directly to the original system through a suitable *interface*, regardless of the specification being enforced. As a by-product, we introduce new simulation relations that are useful in practice for developing smart abstractions in control systems.

Moreover, we established that the existence of a system relation between the original system and its abstraction is not only sufficient but also necessary for ensuring this control architecture. To achieve this, we introduced the concept of the *augmented system*, a system transformation that maintains a feedback refinement relation with the abstract system. This allows to generalize the plug-and-play property of the feedback refinement control architecture to other types of system relations.

Part III: Algorithms

Our contributions in this part focus on tailoring the abstraction construction to the specific control problems by merging the abstraction construction and the controller synthesis. By co-designing the abstraction and the abstract controller based on the optimal control problem, the abstraction is constructed incrementally, focusing only on the relevant regions of the state space.

We first established the connection between alternating simulations and the Bellman operator, demonstrating how suboptimal and superoptimal value functions can be translated from the abstract system to the concrete system.

Next, we formalized modular foundations for constructing hierarchical abstractions, enabling the transfer of optimal cost bounds between different levels of nested partitions. This included the development of a branch-and-bound algorithm that leverages these bounds, with numerical experiments illustrating its practical significance.

Additionally, we provided a tractable algorithm for the optimal control of L-smooth nonlinear dynamical systems. By employing local feedback controllers, we circumvented the need for input space discretization, ensuring deterministic transitions. The combined use of ellipsoid-based covering and affine local controllers further leveraged convex optimization to create larger, non-standard cells.

Finally, we introduced `Dionysos.jl`, a software package offering a modular platform for the abstract symbolic representation of systems and controllers with safety guarantees. This package extends beyond traditional

toolboxes by allowing the construction of abstractions with a simplified concretization step and facilitating the design of low-level controllers. It integrates advanced algorithms and enables user interaction with control-specific insights, ensuring more tailored and effective solutions.

⋆ Perspectives

This work has opened several paths for future research. At the end of each chapter, we have already sketched several perspectives for future work. In addition, we present below a few general ideas for future research.

Application to output-feedback control: In this thesis, we focus on the design of state-feedback abstraction-based controllers, assuming full and exact information of the system states. However, this assumption can be quite restrictive in real-world applications, where only output variables or sensor measurements are available. As a future direction, we plan to address a more realistic scenario where the system state cannot be directly observed. To achieve this, we propose introducing a new system relation, termed the *output-feedback relation* (OFR), which generalizes the alternating simulation relation and enables the synthesis of output-feedback controllers to enforce state specifications. In this approach, while the state of the concrete system is only accessible through an output map, the abstraction state is directly observable. This allows state-feedback controllers to be applied to the abstraction, avoiding to solve symbolic synthesis problems with partial information, which can be a challenging task.

Furthermore, we believe that such a relation can be employed to reformulate various existing abstraction-based techniques within a common framework, including those based on:

- *observers* [Lue71, ST99], which estimate the system's state by using measurements of the inputs and outputs along with a model of the system dynamics, such as in [APGCZ20, HAVdH15, MOM14, MU18, PDBB19]. For example, in [APGCZ20], the authors propose an abstraction that enables the synthesis of output-feedback controllers using observers derived from the original dynamics for nonlinear systems with partial information and bounded disturbances.

- *memory*, such as *l-complete* abstractions [STR15, dAGMJ21], where the

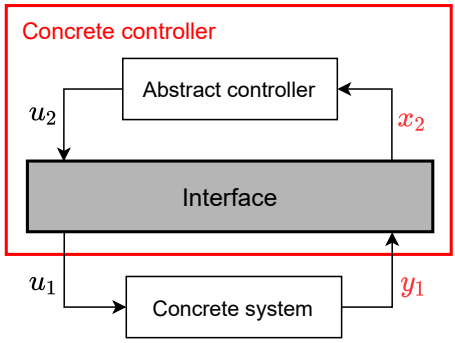


Fig. ⋆.1 Closed-loop system resulting from the concretization of an abstract state-feedback controller (operating on the abstract state x_2) into a concrete output-feedback controller (operating on the concrete output y_1).

states correspond to the system's outputs. The process of constructing an abstraction for a dynamical system can introduce properties that were not present in the original system due to discretization. This can lead to erroneous conclusions when inferring transitions in the original system. By increasing memory, the abstraction becomes more precise, reducing non-determinism and eliminating spurious behaviors, thus providing a more accurate representation of the system's behavior and facilitating analysis and verification of its properties.

Then, we plan to extend the framework introduced in Part II to characterize system relations that refine the output-feedback relation through their concretization procedures. This will include specifying the interface, similar to the approach in Chapter 4, as illustrated in Figure ⋆.1.

Data-driven abstraction-based control: When the model of the original dynamics is too complex or incomplete, over-approximating the behaviors of a system can hinder the applicability of the abstraction-based control techniques presented in this thesis. For this reason, data-driven methods are commonly used, where initial conditions are sampled, and trajectories of fixed length ℓ are observed from these sampled points [DSA21, CPMJ22, ALZ23, KMS+24, CPMJ24]. Consequently, the resulting abstraction, often modeled via a Markov model, may contain not only spurious but also missing behaviors of the original system [BRAJ23].

As a potential future research direction, we could adapt Algorithm 8.9

to a data-driven context for the lazy construction of an interval Markov decision process (IMDP) abstract system. This would follow the approach in [JLFL21], where a data-driven framework is used for partially known stochastic systems. In this approach, the abstract synthesis maximizes the probability of satisfying the given specification, while the concrete controller is derived by combining it with the low-level controllers. This approach could enable the design of abstractions with fewer cells, trading deterministic transitions for a stochastic setting with larger cells. Moreover, it allows for relaxing the assumption of bounded noise by considering stochastic noise distributions.

Long-term vision:

1. **The limitations of the classical 'model and control' paradigm.**
   A common feature of many modern systems is that they do not fit with the classical paradigm of 'model and control.' The data-driven nature of these systems, along with their versatility, makes traditional modeling approaches conservative and limited. Examples include smart grids, which constantly evolve, or modular robotic networks with interchangeable parts. As a result, industries typically resort to developing in-house solutions for specific control problems, which demands substantial resources, dedicated engineering teams, and extended development timelines.

   To address this challenge, we aim to extend our framework to more general dynamical systems while improving both numerical complexity and theoretical conservatism. Our approach is well-suited for an iterative or active learning process, where high-level solutions and bottlenecks inform low-level controllers design, and vice versa.

2. **The need for efficient and robust control tools.**
   The historical significance of effective control solutions cannot be overstated. Control theory remains a key technological bottleneck in deploying (semi-)autonomous systems in aerospace, automotive, and robotics. Advances in control theory have led to major societal changes—such as the development of laser controllers, which played a crucial role in the semiconductor manufacturing revolution. These breakthroughs highlight the importance of finding robust control solutions for increasingly complex systems.

   We believe that the development of a tool capable of automatically generating controllers with formal guarantees, by abstracting both the con-

trol problem and the system through a systematic procedure, represents a paradigm shift in industrial control practices. Such a tool would allow a single engineer to design a controller rapidly, unlike the current industry standard, which requires weeks of work from dedicated engineering teams for specific problems. The `Dionysos.jl` software package introduced in this thesis is positioned to serve as a pioneering platform for these future advancements. Given the complexity of control problems, we envision a meta-solver within `Dionysos.jl` that dynamically combines these modules in an ad-hoc and opportunistic way using Machine Learning and Artificial Intelligence techniques. This approach would leverage the specific structure of each problem, helping to mitigate the curse of dimensionality and further streamline the control design process.

Appendix A: Mathematical review

In this appendix, we provide a list of mathematical results that are used frequently in the text.

Lemma A.1 (Schur Complement Lemma [BEGFB94]). *Consider the matrices* $\mathbf{A} \in \mathbb{R}^{n_1 \times n_1}, \mathbf{B} \in \mathbb{R}^{n_1 \times n_2}, \mathbf{C} \in \mathbb{R}^{n_2 \times n_1}, \mathbf{D} \in \mathbb{R}^{n_2 \times n_2}$. *Let*

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}.$$

If $\mathbf{D}$ *is invertible, the* Schur complement *of the block* $\mathbf{D}$ *of* $\mathbf{M}$ *is defined by*

$$\mathbf{M}/\mathbf{D} = \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}.$$

If $\mathbf{A}$ *is invertible, the* Schur complement *of the block* $\mathbf{A}$ *of* $\mathbf{M}$ *is defined by*

$$\mathbf{M}/\mathbf{A} = \mathbf{D} - \mathbf{B}\mathbf{A}^{-1}\mathbf{C}.$$

We have the following properties of the Schur complement: If $\mathbf{D}$ *is invertible then the matrix* $\mathbf{M}$ *is positive definite if and only if* $\mathbf{D}$ *and* $\mathbf{M}/\mathbf{D}$ *are positive definite. If* $\mathbf{A}$ *is invertible then the matrix* $\mathbf{M}$ *is positive definite if and only if* $\mathbf{A}$ *and* $\mathbf{M}/\mathbf{A}$ *are positive definite. Moreover, if* $\mathbf{A} \in \mathbb{S}^{n_1}, \mathbf{D} \in \mathbb{S}^{n_2}$ *and* $\mathbf{C} = \mathbf{B}^\top$, *we have the following properties. If* $\mathbf{D}$ *is positive definite then the matrix* $\mathbf{M}$ *is positive semidefinite if and only if* $\mathbf{M}/\mathbf{D}$ *is positive semidefinite. If* $\mathbf{A}$ *is positive definite then the matrix* $\mathbf{M}$ *is positive semidefinite if and only if* $\mathbf{M}/\mathbf{A}$ *is positive semidefinite.*

Lemma A.2 (S-procedure [PT07, Theorem 2.2]). *Let* $q_i : \mathbb{R}^n \to \mathbb{R} : x \to q_i(x) = x^\top \mathbf{P}_i x + 2g_i^\top x + s_i$ *for* $i = 0, 1$ *be two quadratic functions and suppose*

that there is a $\bar{x}$ such that $q_1(\bar{x}) < 0$. Then the following two statements are equivalent

$$(i)\ \forall x \in \mathbb{R}^n :\ q_1(x) \leq 0 \Rightarrow q_0(x) \leq 0$$
$$(ii)\ \exists \lambda \geq 0 :\ \lambda \begin{pmatrix} \mathbf{P}_1 & g_1 \\ g_1^\top & s_1 \end{pmatrix} \succeq \begin{pmatrix} \mathbf{P}_0 & g_0 \\ g_0^\top & s_0 \end{pmatrix}.$$

Proposition A.3. *The* volume *of an ellipsoid $\xi = \mathrm{E}(c, \mathbf{P})$ is*

$$\mathrm{vol}(\xi) = \frac{\pi^{n/2}}{\Gamma(n/2+1)} \cdot \frac{1}{\sqrt{\det(\mathbf{P})}},$$

where Γ is the Gamma function, a generalization of the factorial to real numbers

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt, \quad z > 0.$$

For positive integer n, $\Gamma(n) = (n-1)!$.

Corollary A.4 ([BEGFB94, Section 2.2.4])**.** *Let $\xi = \mathrm{E}(c, \mathbf{P})$ be an ellipsoid. Minimizing the volume of ξ with respect to $\mathbf{P}$ is equivalent to*

$$\min_{\mathbf{P} \succ \mathbf{0}} \mathrm{vol}(\mathrm{E}(c, \mathbf{P})) \Leftrightarrow \min_{\mathbf{P} \succ \mathbf{0}} \left(-\log(\det(\mathbf{P}))\right).$$

This is a convex optimization problem.
Maximizing the volume of ξ with respect to $\mathbf{P}$ is equivalent to

$$\max_{\mathbf{P} \succ \mathbf{0}} \mathrm{vol}(\mathrm{E}(c, \mathbf{P})) \Leftrightarrow \min_{\mathbf{L} \succ \mathbf{0}} \left(-\log(\det(\mathbf{L}))\right),$$

where $\mathbf{L}$ is the square root of $\mathbf{P}^{-1}$, i.e., $\mathbf{P}^{-1} = \mathbf{L}^2$ with $\mathbf{P} \succ \mathbf{0}$. This is a convex optimization problem.

Appendix B: A^* algorithm

In this appendix, we present the A^* algorithm [HNR68] to solve an optimal control problem (Definition 2.11) with a reach-avoid specification $\Sigma^{\mathrm{Reach}} = [\mathcal{I}, \mathcal{T}, \mathcal{O}]$, where $\mathcal{I} = \{q_I\}$ is a singleton, and a transition cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{>0}$, for a *deterministic finite* simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$.

The A^* algorithm, outlined in Algorithm B.1 and illustrated in Figure B.1, is an extension of Dijkstra's algorithm [Dij22], designed to efficiently find the shortest path by using heuristics to guide its search and avoid exploring all possible trajectories starting from q_I.

The algorithm maintains two sets: $\mathcal{N}$ and $\mathcal{M}$. The set $\mathcal{N}$ contains states q for which a control policy to solve the reach-avoid problem $[\{q_I\}, \{q\}, \mathcal{O}]$ is available but have not yet been *expanded*, meaning their successors have not yet been computed. The algorithm incrementally extends this set one step at a time until the target set is reached. At each iteration, Algorithm B.1 determines which state q to expand based on the cost incurred so far to reach q from q_I and an estimate of the remaining cost to reach the target set $\mathcal{T}$ from q. Specifically, Algorithm B.1 selects the state that minimizes the evaluation function

$$f(q) = g(q) + h(q), \tag{B.1}$$

where $g(q)$ is the cost from q_I to q, and $h(q)$ is a *heuristic function* estimating the remaining cost from q to the target set $\mathcal{T}$. Once a state q is expanded, it is removed from $\mathcal{N}$ and added to $\mathcal{M}$.

To ensure that the paths maintained by A^* are optimal, the heuristic function must be *admissible* [HNR68, Section II.C], meaning it never overestimates the actual cost to reach the target. If the heuristic is admissible, the set $\mathcal{M}$ contains the states q for which we have the cost of traveling from q_I to q along the optimal path from q_I to $\mathcal{T}$ that passes through q.

Additionally, the heuristic function h is *consistent* [HNR68, Section III.B] if, for all $q \in \mathcal{X}$ and $u \in \mathcal{U}(q)$,

$$h(q) \leq c(q, u) + h(q^+), \tag{B.2}$$

where $q^+ \in F(q, u)$. A consistent heuristic guarantees that A* will find an optimal path without processing any node with $f(q) > c^*$, where c^* is the optimal cost from q_I to $\mathcal{T}$. The tighter the heuristic h, the smaller the portion of the state space that needs to be explored. Additionally, at each iteration

$$\mathcal{N} = \{q \in \mathcal{X} \mid f(q) < \infty,\ g(q) = \infty\}, \quad \mathcal{M} = \{q \in \mathcal{X} \mid g(q) < \infty\}. \tag{B.3}$$

Finally, in Algorithm B.1, the *parent* map is used to track predecessors and reconstruct the controller.

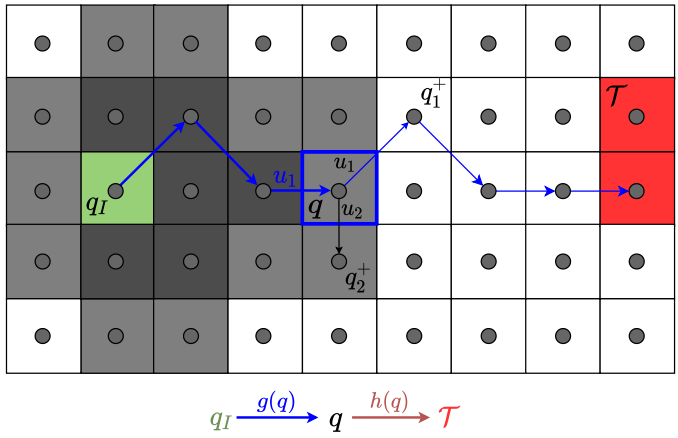


Fig. B.1 Illustration of Algorithm B.1. The set $\mathcal{N}$ and $\mathcal{M}$ are shown in light and dark grey, respectively. When q is selected from $\mathcal{N}$ in Line 16, it is added to $\mathcal{M}$ and expanded, meaning its reachable states q_1^+ and q_2^+ are added to $\mathcal{N}$. The state q_1^+ is added to $\mathcal{N}$ with a cost evaluation $f(q_1^+) = g(q) + c(q, u_1) + h(q_1^+)$ as it was not yet in $\mathcal{N}$, and the parent is set to (q, u_1). As q_2^+ is already in $\mathcal{N}$, its cost evaluation and parent are updated to $g(q) + c(q, u_2) + h(q_2^+)$ and (q, u_2) if $g(q) + c(q, u_2) + h(q_2^+) < f(q_2^+)$. The optimal path from q_I to $\mathcal{T}$ passing through q is highlighted in blue. As shown, once $q \in \mathcal{M}$, $g(q)$ represents the cost of traveling from q_I to q along the optimal path from q_I to $\mathcal{T}$ that passes through q, while $h(q)$ provides a lower bound on the cost to reach $\mathcal{T}$ from q.

Algorithm B.1: A* algorithm for solving an optimal control problem with reach-avoid specification $\Sigma^{\text{Reach}} = [\mathcal{I}, \mathcal{T}, \mathcal{O}]$, where $\mathcal{I} = \{q_I\}$ is a singleton, and transition cost function c, for a deterministic finite simple system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$. Returns a static system $\mathcal{C}$ that solves the control problem $(\mathcal{S}, \Sigma^{\text{Reach}})$ if a solution exists.

Function `constructController`(parent, q^+)**:**
  while $q^+ \notin \mathcal{I}$ **do**
    $(q, u) \leftarrow \text{parent}(q^+)$;
    $H_{\mathcal{C}}(0, q) \leftarrow \{(u, q)\}$;
    $F_{\mathcal{C}}(0, q) \leftarrow \{0\}$;
    $q^+ \leftarrow q$;
  end
  return $\mathcal{C} = (\{0\}, \mathcal{X}, \mathcal{X}, \mathcal{U}, F_{\mathcal{C}}, H_{\mathcal{C}})$;
end
$g(q) \leftarrow \infty$, $\forall q \in \mathcal{X}$;
$f(q) \leftarrow \infty$, $\forall q \in \mathcal{X} \setminus \mathcal{I}$;
$\text{parent}(q) \leftarrow \varnothing$, $\forall q \in \mathcal{X}$;
$f(q) \leftarrow h(q)$, $\forall q \in \mathcal{I}$;
$\mathcal{N} \leftarrow \mathcal{I}$;
$\mathcal{M} \leftarrow \varnothing$;
while $\mathcal{N} \neq \varnothing$ **do**
  $q \leftarrow \text{argmin}\{f(q) \mid q \in \mathcal{N}\}$;
  if $q \in \mathcal{T}$ **then**
    return `constructController`(parent, q);
  end
  $\mathcal{N} \leftarrow \mathcal{N} \setminus \{q\}$;
  $g(q) \leftarrow f(q) - h(q)$;
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{q\}$;
  for $u \in \mathcal{U}(q)$ **do**
    $\{q^+\} \leftarrow F(q, u)$;
    if $q^+ \notin \mathcal{O}$ **then**
      if $g(q) + c(q, u) + h(q^+) < f(q^+)$ **then**
        $f(q^+) \leftarrow g(q) + c(q, u) + h(q^+)$;
        $\text{parent}(q^+) \leftarrow (q, u)$;
        $\mathcal{N} \leftarrow \mathcal{N} \cup \{q^+\}$;
      end
    end
  end
end
return $(\mathcal{S}, \Sigma^{\text{Reach}})$ is infeasible;

Appendix C: Efficient method to verify the inclusion of ellipsoids

For many problems in control and estimation theory, ellipsoidal sets represent a sensible compromise between expressiveness power and numerical tractability. Their simple characterization by a convex quadratic function allows the expression of involved control objectives and constraints as optimization problems that can be handled relatively efficiently by convex optimization. Also, they are suitable for characterizing uncertainties and disturbances, particularly when Gaussian noise is assumed. For a non-exhaustive list of classical applications on control, we refer to [BEGFB94, KV97] and, for estimation, a few instances are [Sch68, BR71, Zol96].

More recently, with the development of modern control techniques, such as abstraction-based control design, neural-network-based control, and data-driven control, among others, the representation of mathematical concepts through ellipsoidal sets has been shown to be also useful in these contexts, e.g., developing barrier functions and local controllers [HLTS20, ELJ22, CEJ24], assessing the safety of neural-networks [FMP19], and capturing data uncertainties in data-driven control methods [BDPT22].

In view of all these applications and the growing necessity for efficient methods to perform numerical operations with ellipsoids, we present a novel approach to verify whether one n-ellipsoid $\xi \subset \mathbb{R}^n$ is a subset of another ellipsoid $\xi_0 \subset \mathbb{R}^n$. Our contributions in this section are listed as follows

- we write the problem of inclusion of two ellipsoids as a concave mini-

mization problem for which strong duality holds. Then, one can decide the inclusion by computing the maximum of the *dual function*, which is a scalar concave function that can be evaluated in $\mathcal{O}(n)$ floating-point operations (FLOPs), as well as its derivative. Nevertheless, note that constructing the dual function in the desired form requires $\mathcal{O}(n^3)$ FLOPs for the computation of the Cholesky factorization of the shape matrix of ξ_0.

- we prove that the dual search domain for the maximum of the dual function can be restricted to a compact set contained in the interval $[0, 1]$, which makes it suitable to be handled by bisection algorithms. Additionally, an early stop criterion is presented, which is triggered within a finite number of iterations of the bisection algorithm when strict inclusion holds.

- we generalize our algorithm to compute the smallest level set of a positive definite quadratic function containing a finite number of n-ellipsoids. This is a problem with applications in control theory, and we present an example, namely, calculating control forward invariant sets.

- we show that, in a benchmark consisting of randomly generated ellipsoids, when compared to the classical semidefinite programming-based method, our approach performs, on average, about 27 times faster for ellipsoids of dimension $n = 3$ and 2294 times faster in dimension $n = 100$.

The most simple case of testing the ellipsoid inclusion happens when they share the same center. As it will be discussed, this can be solved by comparing the eigenvalues of the Hessian matrices of the quadratic functions defining each of them. In the general case, when the ellipsoids do not share the same center, the inclusion problem can be reformulated (see [BEGFB94, p. 43]) as a linear matrix inequality (LMI) problem using the S-lemma [PT07, Thm. 2.2].

This LMI problem yields a semidefinite program (SDP) and, thus, can be solved by one of many readily available SDP solvers. Nonetheless, the performance and accuracy of these solvers are often not ideal as they do not leverage the specific structure of the problem being solved. Therefore, in this work, we design a method that, by exploiting the structure of the ellipsoidal inclusion problem, outperforms general-purpose SDP solvers such as SDPA [YFN+10] and Mosek [ApS19]. In a similar fashion, the authors of [GH12, Prop. 2] reformulate the ellipsoid *intersection* problem as the minimization of a convex scalar function in a bounded interval. Their method requires $\mathrm{O}(n^4)$ FLOPs and is based on algebraic geometry. For

this same problem of intersection of n-ellipsoids, in [RST02], the authors present an algorithm to compute, among all ellipsoids that are convex combinations of two given ones, ξ and ξ_0, the one with minimal volume and that contains $\xi \cap \xi_0$. Their method relies on the computation of the root of a polynomial of degree $2n-1$ defined in a bounded interval, see [RST02, Thm. 3]. For the problem of *inclusion* of ellipsoids, however, no tailored procedure is available in the literature to the best of our knowledge. Finally, we note the similarities of our approach with the secular equations of the trust-region subproblem [WN+99, Section 4.3]. However, we adapt these techniques to the ellipsoid inclusion problem by providing an algorithm with an early stopping criterion that is triggered in a finite number of iterations.

This section is structured as follows. Section C.1 includes the mathematical background needed throughout this section. Section C.2 is devoted to our main result: the optimization formulation of the inclusion test. In Section C.3 we provide the details of the practical implementation, benchmarking with off-the-shelf solvers, and an example of application in control theory.

Most of the results presented in this section are published in [CEJ23]. The results of the numerical experiments presented in Section C.3 are available in `https://github.com/egidioln/EllipsoidInclusion.jl`.

C.1 Preliminary results

The inclusion of ellipsoids

Before presenting our main results, we first state some definitions and present existing results regarding the problem of verifying the inclusion of ellipsoids. For two ellipsoids (1.4) $\xi = \mathrm{E}(c, \mathbf{P})$ and $\xi_0 = \mathrm{E}(c_0, \mathbf{P}_0)$, the *inclusion*, the *strict inclusion* and the *non-inclusion* are denoted by the standard mathematical symbols $\subseteq$, $\subset$, and $\not\subseteq$. Besides these, an additional relation $\subseteq_0$ is considered in this section and defined as follows

$$\xi \subseteq_0 \xi_0 \iff \xi \subseteq \xi_0 \text{ and } \partial\xi \cap \partial\xi_0 \neq \emptyset, \tag{C.1}$$

which means that ξ is included in ξ_0 and both ellipsoids have common points in their boundaries that we will call *contact points*. For studying the inclusion in our context, this situation denotes an extreme case and yields

a particular interpretation of our algorithm to be presented.

Verifying whether one ellipsoid is included in another can be equivalently rewritten as verifying if a surrogate ellipsoid is included in a unit Euclidean ball centered at the origin. The next lemma formalizes this equivalence.

Lemma C.1. *Let matrices $\mathbf{P}, \mathbf{P}_0 \in \mathbb{S}^n_{>0}$ and vectors $c,\ c_0 \in \mathbb{R}^n$ be given. The following equivalences hold*

$$\mathrm{E}(c, \mathbf{P}) \subset \mathrm{E}(c_0, \mathbf{P}_0) \Leftrightarrow \mathrm{E}(\tilde{c}, \tilde{\mathbf{P}}) \subset \mathrm{B}(0, 1);$$
$$\mathrm{E}(c, \mathbf{P}) \subseteq_0 \mathrm{E}(c_0, \mathbf{P}_0) \Leftrightarrow \mathrm{E}(\tilde{c}, \tilde{\mathbf{P}}) \subseteq_0 \mathrm{B}(0, 1);$$

with

$$\tilde{c} = \mathbf{L}_0^\top (c - c_0), \quad \tilde{\mathbf{P}} = \mathbf{L}_0^{-1} \mathbf{P} \mathbf{L}_0^{-\top} \tag{C.2}$$

and $\mathbf{L}_0$ defines the Cholesky factorization of $\mathbf{P}_0 = \mathbf{L}_0 \mathbf{L}_0^\top$.

Proof. As $\mathbf{P}_0 \succ \mathbf{0}$, we have that $\mathbf{L}_0$ is regular. By applying the change of variables $\tilde{x} = \mathbf{L}_0^\top (x - c_0)$, we have that $\mathrm{E}(c, \mathbf{P})$ and $\mathrm{E}(c_0, \mathbf{P}_0)$ becomes respectively $\mathrm{E}(\tilde{c}, \tilde{\mathbf{P}})$ and $\mathrm{B}(0, 1)$ in the space of $\tilde{x}$. □

As a consequence, in this section, we will equivalently study the problem of verifying if an ellipsoid is included in a Euclidean n-ball of radius 1 given that an appropriate change of variables transforming the original problem into this one always exists. Notice that this can be done under $\mathcal{O}(n^3)$ arithmetic operations because of the required Cholesky factorization [Hig09].

For the sake of completeness, before presenting the necessary and sufficient condition for inclusion on which our method is based, we will discuss other simpler criteria that allow us to sufficiently determine whether one ellipsoid is included in the other or not. These can be used as preliminary tests before running our algorithm to further speed up the execution time of an inclusion verification routine.

Proposition C.2. *Let matrices $\mathbf{P}, \mathbf{P}_0 \in \mathbb{S}^n_{>0}$ and vectors $c,\ c_0 \in \mathbb{R}^n$ be given. The following are necessary conditions for $\mathrm{E}(c, \mathbf{P}) \subseteq \mathrm{E}(c_0, \mathbf{P}_0)$:*

1. $c \in \mathrm{E}(c_0, \mathbf{P}_0)$,
2. $\mathbf{P} \succeq \mathbf{P}_0$.

Moreover, if $c = c_0$ then condition (2) is also sufficient.

Proof. Let us demonstrate each of these two statements.

1. This trivially holds from the fact that $c \in \mathrm{E}(c, \mathbf{P})$.
2. To show a contradiction, assume that $\mathrm{E}(c, \mathbf{P}) \subseteq \mathrm{E}(c_0, \mathbf{P}_0)$ but also that there exists $v \in \mathbb{R}^n$ such that $v^\top \mathbf{P}_0 v > v^\top \mathbf{P} v = 1$, without loss of generality. Therefore, $x_+, x_- \in \mathrm{E}(c, \mathbf{P})$, with $x_+ = c + v$, $x_- = c - v$. On the other hand, x_+ and/or x_- are not in $\mathrm{E}(c_0, \mathbf{P}_0)$, which can be verified by developing the left-hand side expression in the definition (1.4) as

$$\begin{aligned}(x_\pm - c_0)^\top \mathbf{P}_0 (x_\pm - c_0) =& (c - c_0 \pm v)^\top \mathbf{P}_0 (c - c_0 \pm v) \\ >& (c - c_0)^\top \mathbf{P}_0 (c - c_0) \\ & \pm 2 v^\top \mathbf{P}_0 (c - c_0) + 1 \\ >& \pm 2 v^\top \mathbf{P}_0 (c - c_0) + 1. \end{aligned} \tag{C.3}$$

   This shows that at least $x_+ \notin \mathrm{E}(c_0, \mathbf{P}_0)$ or $x_- \notin \mathrm{E}(c_0, \mathbf{P}_0)$, which is a contradiction.

To show the sufficiency of (2) for $c = c_0$, note that $(x - c)^\top \mathbf{P} (x - c) \geq (x - c)^\top \mathbf{P}_0 (x - c)$ for all $x \in \mathbb{R}^n$. Therefore, for all $x \in \mathrm{E}(c, \mathbf{P})$ we have $1 \geq (x - c)^\top \mathbf{P} (x - c) \geq (x - c)^\top \mathbf{P}_0 (x - c)$, which implies that $x \in \mathrm{E}(c_0, \mathbf{P}_0)$. □

Although simple, these tests allow us to efficiently decide about the inclusion of ellipsoids for some cases. Additionally, for the case $c = c_0$, one can show in a similar fashion that $\mathbf{P} \succ \mathbf{P}_0$ implies strict inclusion.

An optimization approach

Let us introduce the following optimization problem

$$\begin{aligned} p^* = \min_{x \in \mathbb{R}^n} \; & -x^\top x \\ \text{s.t.} \quad & (x - c)^\top \mathbf{P} (x - c) \leq 1, \end{aligned} \tag{C.4}$$

which finds the point x of maximum Euclidean norm within $\mathrm{E}(c, \mathbf{P})$ with $\mathbf{P} \in \mathbb{S}^n_{>0}$. Therefore, the optimal value p^* is related to the problem of verifying the inclusion of an ellipsoid inside the euclidean unit ball as follows:

$$\mathrm{E}(c, \mathbf{P}) \subseteq \mathrm{B}(0, 1) \Leftrightarrow p^* \geq -1. \tag{C.5}$$

It is also straightforward to show that $p^* = -1$ if and only if $\mathrm{E}(c, \mathbf{P}) \subseteq_0 \mathrm{B}(0, 1)$. Although (C.4) is a non-convex optimization problem, it has some

noteworthy properties. First, there always exists a (strictly) feasible point, given that the ellipsoid $\mathrm{E}(c, \mathbf{P})$ contains its center in its interior. Also, it is always bounded, given that it is a minimization of a concave function within a convex bounded set [Roc70, p. 342]. Finally, the *Lagrangian* of this optimization problem is a quadratic function on x, given as

$$\mathcal{L}(x, \beta) = -x^\top x + \beta \left((x - c)^\top \mathbf{P} (x - c) - 1 \right) \tag{C.6}$$

where β is the Lagrange multiplier associated with the unique constraint of (C.4). Naturally, the *Lagrange dual function* $g : \mathbb{R} \to \mathbb{R}$ is defined by

$$g(\beta) = \min_{x \in \mathbb{R}^n} \mathcal{L}(x, \beta) \tag{C.7}$$

and the dual optimization problem by

$$d^* = \max_{\beta \in \mathcal{D}_g} g(\beta), \tag{C.8}$$

where the dual domain is defined as

$$\mathcal{D}_g = \{\beta \geq 0 \ : \ g(\beta) > -\infty\}. \tag{C.9}$$

Notice that, because of its quadratic nature, the lower-boundedness of the Lagrangian function is closely related to the sign of its Hessian

$$\nabla_x^2 \mathcal{L}(x, \beta) = \beta \mathbf{P} - \mathbf{I}. \tag{C.10}$$

As discussed in [BV04, p. 458], for a given $\beta \geq 0$, this function is bounded from below if $\nabla_x^2 \mathcal{L}(x, \beta) \succ \mathbf{0}$ or if $\nabla_x^2 \mathcal{L}(x, \beta) \succeq \mathbf{0}$ and there exists $x \in \mathbb{R}^n$ such that $\nabla_x \mathcal{L}(x, \beta) = 0$. This implies that the domain of the dual function is either $\mathcal{D}_g = [1/\lambda_{\min}(\mathbf{P}), \infty)$ or $\mathcal{D}_g = (1/\lambda_{\min}(\mathbf{P}), \infty)$, depending on the matrix $\mathbf{P}$ and the vector c defining the Lagrangian (C.6).

The next section clarifies how the dual function (C.7) can be used to construct an efficient algorithm for verifying the inclusion of ellipsoids.

C.2 Main results

An Algorithm to Test the Inclusion of Ellipsoids

Corresponding to the problem of verifying whether the inclusion $\mathrm{E}(c, \mathbf{P}) \subseteq \mathrm{B}(0, 1)$ holds, we define the infinitely differentiable function $\ell_{c,\mathbf{P}} : \mathcal{I}_{\mathbf{P}} \to \mathbb{R}$ as

$$\ell_{c,\mathbf{P}}(\beta) := -\beta - \sum_{i \in \mathrm{support}(\bar{c})} \bar{c}_i^2 \frac{\lambda_i \beta}{\lambda_i \beta - 1} \tag{C.11}$$

and its domain

$$\mathcal{I}_{\mathbf{P}} := \begin{cases} (\lambda_{\min}(\mathbf{P})^{-1}, \infty), & \text{if } \exists i \in \mathrm{support}(\bar{c}),\ \lambda_i = \lambda_{\min}(\mathbf{P}) \\ [\lambda_{\min}(\mathbf{P})^{-1}, \infty), & \text{otherwise} \end{cases} \tag{C.12}$$

where $\bar{c} = \mathbf{V}^\top c$, $\lambda_i = D_{ii}$ and $(\mathbf{V}, \mathbf{D})$ constitute the spectral decomposition of $\mathbf{P} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$. Notice that, within the domain of $\ell_{c,\mathbf{P}}(\beta)$, the only case where the expression (C.11) is undefined is when $\beta = 1/\lambda_{\min}(\mathbf{P})$ and there exists $i \in \mathrm{support}(\bar{c})$ such that $\lambda_i = \lambda_{\min}(\mathbf{P})$. In this case, we remove this lower limit from its domain. More precisely, we will show in the proof of Theorem C.4 that $\ell_{c,\mathbf{P}}$ is the dual function of (C.4).

In the following lemma, we present some important properties of $\ell_{c,\mathbf{P}}$.

Lemma C.3. *Let $\mathbf{P} \in \mathbb{S}^n_{>0}$ and $c \in \mathbb{R}^n$ be given. The function $\ell_{c,\mathbf{P}}(\beta)$, given in (C.11), has the following properties:*

1. *$\ell_{c,\mathbf{P}}(\beta)$ is concave in its domain $\mathcal{I}_{\mathbf{P}}$;*
2. *$\forall \beta \in \mathcal{I}_{\mathbf{P}}$, $\ell_{c,\mathbf{P}}(\beta) < 0$.*

Proof. For $\beta \in \mathcal{I}_{\mathbf{P}}$, we decompose $\ell_{c,\mathbf{P}}$ as

$$\ell_{c,\mathbf{P}}(\beta) = h_0(\beta) + \sum_{i \in \mathrm{support}(\bar{c})} \bar{c}_i^2 \lambda_i h_i(\beta)$$

with $h_0(\beta) = -\beta$ and $h_i(\beta) = -\beta/(\lambda_i \beta - 1)$, $i \in \mathrm{support}(\bar{c})$. Note that, $\bar{c}_i^2 \lambda_i \geq 0$ for all i, given that $\mathbf{P} \succ \mathbf{0}$. Below we proof each property in the statement.

1. Evaluating the first and second derivatives of $\ell_{c,\mathbf{P}}(\beta)$ for $\beta \in \mathcal{D}_g$, we

have

$$\ell'_{c,\mathbf{P}}(\beta) = -1 + \sum_{i \in \text{support}(\bar{c})} \bar{c}_i^2 \frac{\lambda_i}{(\lambda_i \beta - 1)^2}, \tag{C.13}$$

$$\ell''_{c,\mathbf{P}}(\beta) = -2 \sum_{i \in \text{support}(\bar{c})} \bar{c}_i^2 \frac{\lambda_i^2}{(\lambda_i \beta - 1)^3}. \tag{C.14}$$

Since $\mathbf{P} \succ \mathbf{0}$, we have $\lambda_i > 0$, which implies that the function $\ell''_{c,\mathbf{P}}(\beta)$ is strictly negative for all $\beta > \lambda_{\min}(\mathbf{P})^{-1}$. Thus, $\ell_{c,\mathbf{P}}(\beta)$ is concave in $\mathcal{I}_{\mathbf{P}}$.

2. The function $h_0(\beta) < 0$ in $\mathcal{I}_{\mathbf{P}}$ and, for all $i \in \text{support}(\bar{c})$, the function $h_i(\beta) < 0$ in $(\lambda_i^{-1}, \infty) \supseteq \mathcal{I}_{\mathbf{P}}$. We conclude that $\ell_{c,\mathbf{P}}(\beta) < 0$ in $\mathcal{I}_{\mathbf{P}}$.

The proof is concluded. □

As demonstrated in the previous lemma, inside its domain $\mathcal{I}_{\mathbf{P}}$ this function is concave. Hence, we can obtain

$$\ell^*_{c,\mathbf{P}} \coloneqq \sup_{\beta \in \mathcal{I}_{\mathbf{P}}} \ell_{c,\mathbf{P}}(\beta) \tag{C.15}$$

by classical optimization algorithms such as Newton's method, which guarantees a quadratic convergence rate to $\ell^*_{c,\mathbf{P}}$ [NW99, Thm. 3.5], or by a bisection algorithm, which avoids the computation of the second derivative. The following theorem connects the function (C.11) with the ellipsoidal inclusion problem.

Theorem C.4. *Let an ellipsoid* $\mathrm{E}(c, \mathbf{P})$ *be given. Define the function* $\ell_{c,\mathbf{P}}(\beta)$ *as in* (C.11) *and consider its supremum* $\ell^*_{c,\mathbf{P}}$ *over the domain* $\mathcal{I}_{\mathbf{P}}$*. The following equivalences always hold*

$$\mathrm{E}(c, \mathbf{P}) \subset \text{int}(\mathrm{B}(0,1)) \Leftrightarrow \ell^*_{c,\mathbf{P}} > -1;$$
$$\mathrm{E}(c, \mathbf{P}) \subseteq_0 \mathrm{B}(0,1) \Leftrightarrow \ell^*_{c,\mathbf{P}} = -1.$$

Proof. First, let us recall that the optimal solution p^* of the primal optimization problem (C.4) allows us to decide whether $\mathrm{E}(c, \mathbf{P})$ is included or not inside $\mathrm{B}(0,1)$, i.e., the inclusion $\mathrm{E}(c, \mathbf{P}) \subset \text{int}(\mathrm{B}(0,1))$ (resp. $\mathrm{E}(c, \mathbf{P}) \subseteq_0 \mathrm{B}(0,1)$) holds if and only if $p^* > -1$ (resp. $p^* = -1$). Let us show that this problem has no duality gap, that is, $p^* = d^*$ where d^* is the optimal solution of the dual problem (C.8). Notice that Slater's constraint qualification [BV04, p. 226] holds given that $x = c$ is a strictly feasible point.

Although the primal problem is not convex due to the concave objective function, Slater's condition implies strong duality in this case, given that this problem is the minimization of a quadratic function subject to a single quadratic inequality, see [BV04, Sec. B4] for a detailed proof.

Therefore, we must now show that $\ell^*_{c,\mathbf{P}} = d^*$. Notice that, the minimization problem within the definition of the dual function (C.7) is convex for $\beta \in \mathcal{D}_g$ (see discussion following (C.10)) and, therefore, its global minimizers x^* for a given $\beta \in \mathcal{D}_g$ are all points satisfying the first-order optimality condition

$$\nabla_x \mathcal{L}(x^*, \beta) = 2(\beta\mathbf{P} - \mathbf{I})x^* - 2\beta\mathbf{P}c = 0. \tag{C.16}$$

If $\beta > 1/\lambda_{\min}(\mathbf{P})$, then the matrix $(\beta\mathbf{P} - \mathbf{I})$ can be inverted and there is a unique minimizer $x^*(\beta) = (\beta\mathbf{P} - \mathbf{I})^{-1}\beta\mathbf{P}c$. However, when $\beta = 1/\lambda_{\min}(\mathbf{P})$, two cases may occur: 1) there is no x^* such that (C.16) holds and therefore, $1/\lambda_{\min}(\mathbf{P})$ does not belong to the domain $\mathcal{D}_g$ of the dual function $g(\beta)$; 2) there exist infinitely many x^* satisfying (C.16), which are given by

$$x^*(\beta) = (\beta\mathbf{P} - \mathbf{I})^\dagger\beta\mathbf{P}c + \mathbf{N}v \tag{C.17}$$

where $\mathbf{N} \in \mathbb{R}^{n\times m}$ has columns spanning the m-dimensional nullspace of $\big(\mathbf{P} - \lambda_{\min}(\mathbf{P})\mathbf{I}\big)$ and $v \in \mathbb{R}^m$. As each of these minimizers is a global minimizer, without loss of generality, let us choose $x^*(\beta) = (\beta\mathbf{P} - \mathbf{I})^\dagger\beta\mathbf{P}c$ (i.e., $v = 0$). Naturally, this minimizer must satisfy (C.16) by assumption. Substituting $x^*(\beta)$ into (C.16) and performing a few algebraic manipulations yields

$$-\beta(\beta\mathbf{P} - \mathbf{I})^\dagger\mathbf{P}c = \beta\mathbf{P}c - \beta^2\mathbf{P}(\beta\mathbf{P} - \mathbf{I})^\dagger\mathbf{P}c. \tag{C.18}$$

Hence, the dual function (C.7) can be rewritten as

$$\begin{aligned} g(\beta) &= \mathcal{L}(x^*(\beta), \beta) \\ &= \beta(c^\top\mathbf{P}c - 1) - \beta^2 c^\top\mathbf{P}(\beta\mathbf{P} - \mathbf{I})^\dagger\mathbf{P}c \\ &= -\beta + c^\top(\beta\mathbf{P} - \beta^2\mathbf{P}(\beta\mathbf{P} - \mathbf{I})^\dagger\mathbf{P})c \\ &= -\beta - \beta c^\top(\beta\mathbf{P} - \mathbf{I})^\dagger\mathbf{P}c \\ &= -\beta - \beta\bar{c}^\top(\beta\mathbf{D} - \mathbf{I})^\dagger\mathbf{D}\bar{c}, \end{aligned}$$

where (C.18) was used to obtain the before last equation and the last equation uses the spectral decomposition $\mathbf{P} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$ and the transformation $\bar{c} = \mathbf{V}^\top c$. Moreover, due to the fact that all matrices in this last expression

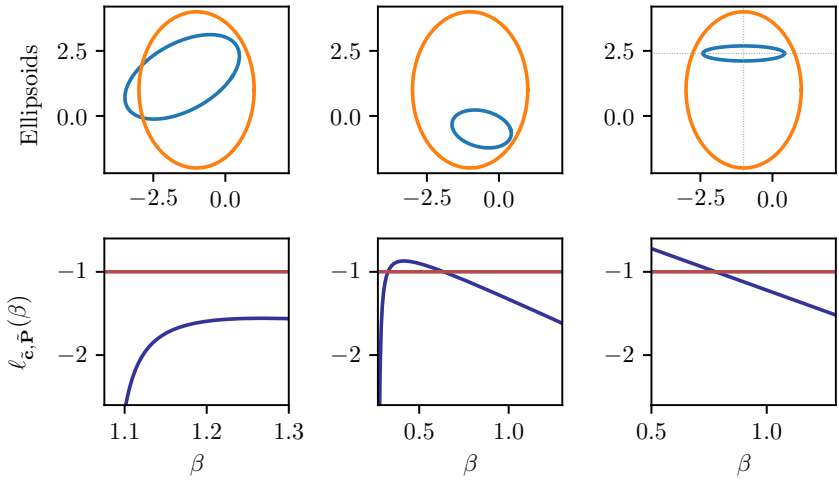


Fig. C.1 Illustration of results of Theorem C.4. The ellipsoid $\tilde{\xi}$ (blue) is contained in $\tilde{\xi}_0$ (orange) if and only if the maximum of $\ell_{\tilde{c},\tilde{\mathbf{P}}}(\beta)$ is greater than -1.

are diagonal, one can rewrite

$$g(\beta) = -\beta - \sum_{i \in \text{support}(\bar{c})} \bar{c}_i^2 \frac{\lambda_i \beta}{\lambda_i \beta - 1}. \tag{C.19}$$

Therefore the dual function $g(\beta) = \ell_{c,\mathbf{P}}(\beta)$ for all $\beta \in \mathcal{D}_g = \mathcal{I}_{\mathbf{P}}$, and, hence, $d^* = \ell^*_{c,\mathbf{P}}$, concluding the proof. □

Remark C.5. The inclusion $\mathrm{E}(c, \mathbf{P}) \subseteq \mathrm{B}(0, 1)$ is equivalent to

$$\max_{\beta \geq 0} \min_{x \in \mathbb{R}^n} \mathcal{L}(x, \beta) \geq -1 \tag{C.20}$$

which, in turn, is equivalent to

$$\exists \beta \geq 0,\ \forall x \in \mathbb{R}^n :\ \mathcal{L}(x, \beta) \geq -1. \tag{C.21}$$

We can rewrite $\mathcal{L}(x, \beta) + 1 = [x^\top \ \ 1]\mathbf{F}(\beta)[x^\top \ \ 1]^\top$ with

$$\mathbf{F}(\beta) := \begin{pmatrix} \beta\mathbf{P} - \mathbf{I} & -\beta\mathbf{P}c \\ -\beta c^\top \mathbf{P} & \beta(c^\top \mathbf{P} c - 1) + 1 \end{pmatrix} \tag{C.22}$$

to show that (C.21) is equivalent to

$$\exists \beta \geq 0 : \ \mathbf{F}(\beta) \succeq \mathbf{0}. \tag{C.23}$$

Note that this last inequality is the LMI condition proposed by [BEGFB94, Sec. 3.7.1] characterizing the inclusion $\mathrm{E}(c, \mathbf{P}) \subseteq \mathrm{B}(0, 1)$. This shows the equivalence and the connection between this approach and ours. A numerical comparison, in terms of computational time and memory required to verify inclusions by both methods, is presented in Section C.3. △

Theorem C.4 is the foundation of our algorithm to verify the inclusion of ellipsoids. We recall that, although one ellipsoid is considered to be a Euclidean n-ball $\mathrm{B}(0, 1)$, Lemma C.1 provides a change of variables that always allows us to transform the general problem into the one tackled in Theorem C.4.

In Figure C.1, an illustration of the results of Theorem C.4 is provided. There, three cases of ellipsoids $\xi = \mathrm{E}(c, \mathbf{P})$ (blue) and $\xi_0 = \mathrm{E}(c_0, \mathbf{P}_0)$ (orange) are depicted, along with the corresponding functions $\ell_{\tilde{c},\tilde{\mathbf{P}}}$ in the interval $[1/\lambda_{\min}(\tilde{\mathbf{P}}), 1.3]$, where $\tilde{c}$ and $\tilde{\mathbf{P}}$ are given in (C.2). Notice that, for the first case (left) the inclusion of ξ (blue ellipsoid) within ξ_0 (orange) does not hold and the corresponding function $\ell_{\tilde{c},\tilde{\mathbf{P}}}$ is always strictly below -1. For the second case (middle), the inclusion holds, and therefore, the maximum of $\ell_{\tilde{c},\tilde{\mathbf{P}}}$ is greater than -1. Finally, a third case (right) illustrates the case when $1/\lambda_{\min}(\tilde{\mathbf{P}}) \in \mathcal{D}_g$. This happens because, after transforming ξ_0 into the unit Euclidean ball centered at the origin, the center $\tilde{c}$ of the transformed ξ is perpendicular to its greatest semi-axis (or semi-axes), which is the eigenvector associated to $\lambda_{\min}(\tilde{\mathbf{P}})$.

Before introducing our general algorithm, let us present an additional property of the scalar function $\ell_{c,\mathbf{P}}(\beta)$, defined in (C.11), which will be important for implementation purposes.

Proposition C.6. *Let $\mathbf{P} \in \mathbb{S}^n_{>0}$ and $c \in \mathbb{R}^n$ be given. The function $\ell_{c,\mathbf{P}}(\beta)$, defined in (C.11) satisfies*

$$\ell_{c,\mathbf{P}}(\beta) < -1, \quad \forall \beta > \max\left\{\lambda_{\min}(\mathbf{P})^{-1}, 1 - c^\top c\right\}.$$

Proof. By contradiction, assume that there exists $\beta_0 > 1 - c^\top c$ such that $\beta_0 > \lambda_{\min}(\mathbf{P})^{-1}$ and $\ell_{c,\mathbf{P}}(\beta_0) \geq -1$. From the proof of Theorem C.4, we

have that $\ell_{c,\mathbf{P}}(\beta_0) = g(\beta_0)$, which implies

$$\begin{aligned}\ell_{c,\mathbf{P}}(\beta_0) &= \min_{x\in\mathbb{R}^n} \mathcal{L}(x,\beta_0)\\ &\leq \mathcal{L}(c,\beta_0)\\ &= -c^\top c - \beta_0 < -1\end{aligned}$$

which is a contradiction. □

Besides providing a useful upper bound on the interval on which $\ell_{c,\mathbf{P}}(\beta) \geq -1$, reducing the search space for β that maximizes $\ell_{c,\mathbf{P}}(\beta)$, Proposition C.6 also provides a sufficient condition for $\mathrm{E}(c,\mathbf{P}) \not\subseteq \mathrm{B}(0,1)$. Indeed, if $1 - c^\top c < 1/\lambda_{\min}(\mathbf{P})$, then $\ell_{c,\mathbf{P}}(\beta) < -1$ for all $\beta \geq 1/\lambda_{\min}(\mathbf{P})$ and, therefore, the inclusion does not hold.

Algorithm C.1: Test the inclusion of an ellipsoid $\xi = \mathrm{E}(c,\mathbf{P})$ in another ellipsoid $\xi_0 = \mathrm{E}(c_0,\mathbf{P}_0)$.

```
1  if isPreTestConclusive() then
2  |  return PreTestConclusion();
3  L_0 ← Cholesky factorization of P_0 = L_0 L_0^⊤;
4  (c̃, P̃) ← (L_0^⊤ (c − c_0), L_0^{−1} P L_0^{−⊤});
5  V, D ← Spectral decomposition of P̃ = V D V^⊤;
6  ℓ*_{c̃,P̃} ← max_{β∈I_{c̃,P̃}} ℓ_{c̃,P̃}(β);
7  if ℓ*_{c̃,P̃} > −1 then
8  |  return ξ ⊂ int(ξ_0);
9  else if ℓ*_{c̃,P̃} = −1 then
10 |  return ξ ⊆_0 ξ_0;
11 else
12 |  return ξ ⊈ ξ_0;
```

Finally, these results can be joined together into Algorithm C.1. It is important to highlight that this algorithm starts at Line 1-Line 2 by verifying the tests of Proposition C.2 and Proposition C.6 through the function `isPreTestConclusive()`. Whenever these tests are not conclusive, the algorithm proceeds to evaluate the necessary and sufficient condition from Theorem C.4. To do so, notice that Line 3-Line 5 comprise the initialization of the algorithm and can be computed within $\mathcal{O}(n^3)$ FLOPs, due to the Cholesky Factorization [Hig09] and the spectral decomposition [BGVKS22]. Line 6 consists in the maximization of a concave scalar

function $\ell_{\tilde{c},\tilde{\mathbf{P}}}(\beta)$, defined in (C.11), on the interval

$$\mathcal{I}_{\tilde{c},\tilde{\mathbf{P}}} \coloneqq [\lambda_{\min}(\tilde{\mathbf{P}})^{-1}, 1 - \tilde{c}^\top \tilde{c}]. \tag{C.24}$$

The upper bound of the interval $\mathcal{I}_{\tilde{c},\tilde{\mathbf{P}}}$ is determined by Proposition C.6. This maximization problem can be solved without difficulty by a bisection algorithm or, more efficiently, by Newton's method. Notice that, for the later, one needs to compute the first and second derivatives of $\ell_{\tilde{c},\tilde{\mathbf{P}}}(\beta)$, defined respectively in (C.13) and (C.14), which can be done in $\mathcal{O}(n)$ FLOPs.

Algorithm C.1 can be early-stopped whenever a $\beta \in \mathcal{I}_{\tilde{c},\tilde{\mathbf{P}}}$ is found such that $\ell_{\tilde{c},\tilde{\mathbf{P}}}(\beta) > -1$. However, as discussed in the next subsection, computing this maximum is an efficient way to obtain a distance between the boundaries of the two ellipsoids (when the inclusion holds) or by how much ξ_0 must be inflated so it contains ξ.

Consequences of Theorem C.4

The first consequence of Theorem C.4 that we discuss in this section is the fact that whenever $\mathrm{E}(c, \mathbf{P}) \subseteq_0 \mathrm{B}(0, 1)$, knowing $\beta > 0$ that maximizes $\ell_{c,\mathbf{P}}(\beta)$ allows us to fully characterize contact points between $\mathrm{E}(c, \mathbf{P})$ and $\partial\,\mathrm{B}(0, 1)$.

Corollary C.7. *If* $\mathrm{E}(c, \mathbf{P}) \subseteq_0 \mathrm{B}(0, 1)$ *then, all contact points*

$$\bar{x} \in \{x \,:\, (x - c)^\top \mathbf{P}(x - c) = x^\top x = 1\} \tag{C.25}$$

satisfy

$$\bar{x} = (\beta^* \mathbf{P} - \mathbf{I})^\dagger \beta^* \mathbf{P} c + \mathbf{N} v \tag{C.26}$$

for some $v \in \mathbb{R}^m$ *where* $\mathbf{N} \in \mathbb{R}^{n \times m}$ *has columns spanning the m-dimensional nullspace of* $(\beta^* \mathbf{P} - \mathbf{I})$ *and*

$$\beta^* = \arg\max_{\beta \in \mathcal{I}_{c,\mathbf{P}}} \ell_{c,\mathbf{P}}(\beta).$$

Proof. By the fact that strong duality holds, the contact points $\bar{x}$ are optimal solutions of the primal problem (C.4) and, thus, satisfy the equation (C.16) for the optimal β^* of the dual problem and, thus, satisfy (C.26). □

Corollary C.7 provides a complete characterization of the contact points between the boundary of the two ellipsoids. Notice that, there exists a unique contact point whenever $\beta^* > 1/\lambda_{\min}(\mathbf{P})$, which ensures that the

matrix $(\beta^*\mathbf{P} - \mathbf{I})$ has full rank. However, if $\beta^* = 1/\lambda_{\min}(\mathbf{P})$, the uniqueness is no longer guaranteed. In this case, a contact point can be found by selecting $v = \alpha v_0$ with $v_0 \in \mathbb{R}^m$ and finding the scalar α that solves the quadratic equation generated by evaluating $\bar{x}^\top \bar{x} = 1$.

As shown in Theorem C.4, the maximal value $\ell^*_{c,\mathbf{P}}$ being greater or lesser than -1 provides some geometrical insights regarding the inclusion. Besides that, its magnitude yields additional information, as the next corollary shows.

Corollary C.8. *For any $c, c_0 \in \mathbb{R}^n$ and $\mathbf{P}, \mathbf{P}_0 \in \mathbb{S}^n_{>0}$, let $\tilde{c}$ and $\tilde{\mathbf{P}}$ be defined as in (C.2) and $\ell^*_{\tilde{c},\tilde{\mathbf{P}}}$, defined in (C.15). The following inclusions hold*

$$\mathrm{E}(c, \mathbf{P}) \subseteq_0 \mathrm{E}(c_0, (-\ell^*_{\tilde{c},\tilde{\mathbf{P}}})^{-1}\mathbf{P}_0) \tag{C.27}$$

$$\mathrm{E}(d, -\ell^*_{\tilde{c},\tilde{\mathbf{P}}}\mathbf{P}) \subseteq_0 \mathrm{E}(c_0, \mathbf{P}_0) \tag{C.28}$$

with $d = (-\ell^*_{\tilde{c},\tilde{\mathbf{P}}})^{-1/2}(c - c_0) + c_0$.

Proof. By Lemma C.1, we have that $\mathrm{E}(c, \mathbf{P}) \subseteq_0 \mathrm{E}(c_0, \gamma^{-1}\mathbf{P}_0)$ if and only if $\mathrm{E}(\gamma^{-1/2}\tilde{c}, \gamma\tilde{\mathbf{P}}) \subseteq_0 \mathrm{B}(0, 1)$ with $\tilde{c}$, $\tilde{\mathbf{P}}$ defined in (C.2). By its definition in (C.11) we have

$$\ell^*_{\tilde{c},\tilde{\mathbf{P}}} = -\beta^* - \sum_{i \in \mathrm{support}(\tilde{c})} \tilde{c}_i^2 \frac{\lambda_i \beta^*}{\lambda_i \beta^* - 1}. \tag{C.29}$$

Let $\beta^*_\gamma = \gamma^{-1}\beta^*$ and $\gamma = -\ell^*_{\tilde{c},\tilde{\mathbf{P}}}$. By the property (2) in Lemma C.3, we have $\beta^*_\gamma > 0$. Tedious but simple algebraic manipulations where (C.29) is used, yield $\ell_{\gamma^{-1/2}\tilde{c},\gamma\tilde{\mathbf{P}}}(\beta^*_\gamma) = -1$ and $\ell'_{\gamma^{-1/2}\tilde{c},\gamma\tilde{\mathbf{P}}}(\beta^*_\gamma) = 0$ (recall (C.13)), which shows that β^*_γ is the maximizer of $\ell_{\gamma^{-1/2}\tilde{c},\gamma\tilde{\mathbf{P}}}(\beta)$. Therefore, Theorem C.4 ensures that $\mathrm{E}(c, \mathbf{P}) \subseteq_0 \mathrm{E}(c_0, \gamma^{-1}\mathbf{P}_0)$, which is the inclusion (C.27) in the statement. As a consequence, (C.28) also holds as the ellipsoids therein are the same as those in (C.27) after a translation of $-c_0$, a uniform scaling of $(-\ell^*_{\tilde{c},\tilde{\mathbf{P}}})^{-1/2}$, and a translation of c_0. □

Notice that, different interpretations can be given to the inclusion (C.27) in Corollary C.8:

- $\ell^*_{\tilde{c},\tilde{\mathbf{P}}} > -1$: The ellipsoid $\mathrm{E}(c_0, \mathbf{P}_0)$ can be compressed by, at most, a factor of $(-\ell^*_{\tilde{c},\tilde{\mathbf{P}}})^{1/2} < 1$ and will still contain $\mathrm{E}(c, \mathbf{P})$;
- $\ell^*_{\tilde{c},\tilde{\mathbf{P}}} = -1$: Corollary C.8 becomes trivial, as we have $\gamma = 1$ and there is already a contact point between the boundaries of $\mathrm{E}(c_0, \mathbf{P}_0)$ and $\mathrm{E}(c, \mathbf{P})$;

- $\ell^*_{\tilde{c},\tilde{\mathbf{P}}} < -1$: The ellipsoid $\mathrm{E}(c_0, \mathbf{P}_0)$ must be inflated by, at least, a factor of $(-\ell^*_{\tilde{c},\tilde{\mathbf{P}}})^{1/2} > 1$ to contain $\mathrm{E}(c, \mathbf{P})$.

The same applies to (C.28) but considering inflation and compression of $\mathrm{E}(c, \mathbf{P})$ with respect to the c_0.

Intuitively, Corollary C.8 provides a simple method for determining what is the ellipsoid of minimum volume centered in c_0 and with a shape defined by the spectrum of $\mathbf{P}_0$ that contains $\mathrm{E}(c, \mathbf{P})$. Alternatively, we can also compute the least sub-level set of the quadratic function $x \mapsto (x - c_0)^\top \mathbf{P}_0 (x - c_0)$ containing the n-ellipsoid $\mathrm{E}(c, \mathbf{P})$. This interpretation will be explored in the following sections, with applications in control design problems.

C.3 Numerical experiments

Implementation details

To solve the optimization problem in Line 6 of Algorithm C.1, we propose a bisection algorithm (Algorithm C.2) with stopping criteria guaranteeing inclusion without contact point ($\xi \subset \mathrm{int}(\xi_0)$) and non-inclusion ($\xi \not\subseteq \xi_0$). Note that the inclusion with contact points ($\xi \subseteq_0 \xi_0$) cannot be decided numerically, motivating the definition of $\xi \subseteq^\epsilon_0 \xi_0$, which means that, given a machine precision ϵ of the computer, the algorithm cannot determine if $\xi \subset \mathrm{int}(\xi_0)$ or $\xi \not\subseteq \xi_0$.

Proposition C.9. *Algorithm C.2 is correct and terminates in a finite number of steps.*

Proof. Let $\beta^* = \arg\max_{\beta \in \mathcal{I}_\mathbf{P}} \ell_{c,\mathbf{P}}(\beta)$. Whenever $\beta^* \notin \mathcal{I}_{c,\mathbf{P}}$ we have $\ell'_{c,\mathbf{P}}(1 - c^\top c) > 0$ and $\ell_{c,\mathbf{P}}(1 - c^\top c) < -1$ implies, by Proposition C.6, that $\xi \not\subseteq \xi_0$, which the algorithm deals with in Line 5. Let us consider the case $\beta^* \in \mathcal{I}_{c,\mathbf{P}}$. In an arbitrary interval $[l, u] \subset \mathcal{I}_{c,\mathbf{P}}$ containing β^* for which $l > 1/\lambda_{\min}(\mathbf{P})$, the function $\ell_{c,\mathbf{P}}$ is locally L-smooth with $L = \max_{\beta \in [l,u]} |\ell''_{c,\mathbf{P}}(\beta)| = -\ell''_{c,\mathbf{P}}(l)$ since $\ell''_{c,\mathbf{P}}$ is negative and increasing in $[l, u]$. Therefore, due to [N+18, Lemma 1.2.3] we have

$$\ell_{c,\mathbf{P}}(\beta^*) - \tfrac{L}{2}(\beta - \beta^*)^2 \leq \ell_{c,\mathbf{P}}(\beta) \leq \ell_{c,\mathbf{P}}(\beta^*) \quad \forall \beta \in [l, u]$$

and since $\beta^* \in [l, u]$, we obtain the following lower and upper bounds on

Algorithm C.2: Test the inclusion of an ellipsoid $\xi = \mathrm{E}(c, \mathbf{P})$ in the ball $\xi_0 = \mathrm{B}(0, 1)$. Returns either $\xi \subset \mathrm{int}(\xi_0)$, $\xi \not\subseteq \xi_0$, or $\xi \subseteq_0^\epsilon \xi_0$.

1 $l_0, u_0 \leftarrow \lambda_{\min}(\mathbf{P})^{-1}, 1 - c^\top c$;
2 **if** $\ell_{c,\mathbf{P}}(l_0) > -1$ **or** $\ell_{c,\mathbf{P}}(u_0) > -1$ **then**
3 | **return** $\xi \subset \mathrm{int}(\xi_0)$;
4 **if** $\ell'_{c,\mathbf{P}}(u_0) > 0$ **then**
5 | **return** $\xi \not\subseteq \xi_0$;
6 $k \leftarrow 0$;
7 **while** $u_k - l_k > \epsilon$ **do**
8 | $\beta_k = \frac{l_k + u_k}{2}$;
9 | **if** $\ell_{c,\mathbf{P}}(\beta_k) > -1$ **then**
10 | | **return** $\xi \subset \mathrm{int}(\xi_0)$;
11 | **else if** $\ell_{c,\mathbf{P}}(\beta_k) < -1 + \ell''_{c,\mathbf{P}}(l_k)\frac{(u_k - l_k)^2}{2}$ **then**
12 | | **return** $\xi \not\subseteq \xi_0$;
13 | **if** $\ell'_{c,\mathbf{P}}(\beta_k) < 0$ **then**
14 | | $l_{k+1}, u_{k+1} \leftarrow l_k, \beta_k$;
15 | **if** $\ell'_{c,\mathbf{P}}(\beta_k) > 0$ **then**
16 | | $l_{k+1}, u_{k+1} \leftarrow \beta_k, u_k$;
17 | $k \leftarrow k + 1$;
end
18 **return** $\xi \subseteq_0^\epsilon \xi_0$;

$\ell^*_{c,\mathbf{P}}$ for all $\beta \in [l, u]$:

$$\ell_{c,\mathbf{P}}(\beta) \leq \ell_{c,\mathbf{P}}(\beta^*) \leq \ell_{c,\mathbf{P}}(\beta) - \frac{\ell''_{c,\mathbf{P}}(l)}{2}(u - l)^2. \tag{C.30}$$

At each iteration k, Line 14 or Line 16 ensure that $\beta^* \in [l_k, u_k]$.

If at a given iteration k, either $\ell_{c,\mathbf{P}}(\beta_k) \geq -1$, or $\ell_{c,\mathbf{P}}(\beta_k) - \ell''_{c,\mathbf{P}}(l_k)(u_k - l_k)^2/2 < -1$, one can respectively conclude by (C.30) that $\ell^*_{c,\mathbf{P}} > -1$ or $\ell^*_{c,\mathbf{P}} < -1$.

Whenever $\ell^*_{c,\mathbf{P}} > -1$, the continuity of $\ell_{c,\mathbf{P}}(\beta)$ ensures that the condition in Line 9 will be satisfied at a given iteration, given that the interval $[l_k, u_k]$ converges to β^* as $k \to \infty$. In the opposite case, when $\ell^*_{c,\mathbf{P}} < -1$, Algorithm C.2 also stops by satisfying the condition in Line 11. This holds by the same argument that $[l_k, u_k]$ converges to β^*, which also implies that $(u_k - l_k)^2 \to 0$ and $\ell''_{c,\mathbf{P}}(l_k) \to \ell''_{c,\mathbf{P}}(\beta^*) < 0$ as $k \to \infty$, concluding the proof. Finally, if $\ell^*_{c,\mathbf{P}} = 1$, the stop criterion of the main loop (i.e., $u_k - l_k \leq \epsilon$, for a

given precision $\epsilon > 0$) will be satisfied and the algorithm returns $\mathcal{E} \subseteq_0^\epsilon \mathcal{E}_0$, implying that for this precision ϵ the inclusion with contact points may hold. □

Finally, note that it is computationally inexpensive to evaluate $\ell_{c,\mathbf{P}}(\beta)$ and its derivatives. Once $\bar{c}_i^2$, $i \in \text{support}(\bar{c})$, and the eigenvalues of $\mathbf{P}$ have been computed, the exact number of FLOPs to evaluate the function $\ell_{c,\mathbf{P}}$ for a given β is $5|\text{support}(\bar{c})|$, which, in the worst case, represents $5n$ FLOPs. Similarly, by performing some preliminary computations, the evaluation of $\ell'_{c,\mathbf{P}}$ and $\ell''_{c,\mathbf{P}}$ at a given β requires respectively $5|\text{support}(\bar{c})|$ and $6|\text{support}(\bar{c})|$ FLOPs. Note that the evaluation of $\ell_{c,\mathbf{P}}$ and its derivative share several algebraic operations, which can be used to reduce the total number of computations.

Performances

Let us now compare the performance of Algorithm C.1 with those of solving the LMI condition (C.22) provided by [BEGFB94] with two conventional SDP solvers, namely, SDPA [YFN+10] and Mosek [ApS19]. The underlying SDP problem tries to find $\beta \geq 0$ such that the matrix $\mathbf{F}(\beta) \succeq \mathbf{0}$ and, although a single variable is being searched for, the problem deals with an SDP restriction of size $n + 1$.

Figure C.2 shows the average execution times (in seconds) for 200 randomly generated problems, each consisting of two ellipsoids $\mathcal{E} = \mathrm{E}(c, \mathbf{P})$ and $\mathcal{E}_0 = \mathrm{E}(c_0, \mathbf{P}_0)$. These were generated in a way that, in 100 cases, we have $\mathcal{E} \subset \text{int}(\mathcal{E}_0)$, and $\mathcal{E} \not\subset \mathcal{E}_0$ in the remaining ones. Also, we made sure that the conditions in Proposition C.2 do not hold, so running the bisection algorithm (Algorithm C.2) was required each time. The comparison with the SDP solvers was repeated for n-ellipsoids with $n \in \{3, 10, 30, 100\}$ to evaluate how these approaches scale up. These were performed in an Intel(R) Xeon(R) W-2295 CPU @ 3.00GHz with Julia 1.7.3 and using the optimization toolbox JuMP [DHL17]. We computed that, in this benchmark, the execution times for Algorithm C.1 were lesser than those for SDPA and Mosek in all cases. On average, our algorithm performs 27, 49, 162, and 2294 times faster than the competitors for ellipsoids of dimensions 3, 10, 30, and 100, respectively. In terms of memory consumption, Table C.5 shows the average allocated memory (in kilobytes) of each case. This also demonstrates that our method is not only faster but also requires considerably fewer resources than the LMI-based approach.

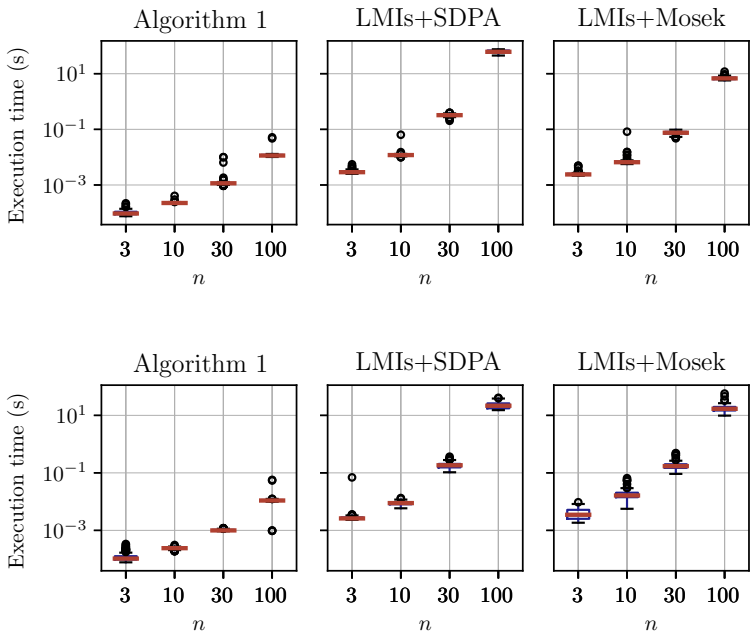


Fig. C.2 Comparison of the execution time for testing ellipsoid inclusion $\xi = \mathrm{E}(c, \mathbf{P}) \subseteq \mathrm{E}(c_0, \mathbf{P}_0) = \xi_0$ as a function of ellipsoid dimension n. Top: $\xi \subset \mathrm{int}(\xi_0)$. Bottom: $\xi \not\subseteq \xi_0$.

	$\xi \subset \mathrm{int}(\xi_0)$			$\xi \not\subseteq \xi_0$		
n	Alg. C.1	SDPA	MOSEK	Alg. C.1	SDPA	MOSEK
3	8.2	192.6	153.5	13.8	203.3	152.1
10	26.5	733.2	577.8	36.4	733.1	576.0
30	118.4	4866.6	3792.1	138.6	4866.6	3790.3
100	865.8	51822.6	39608.1	901.8	51822.6	39606.2

Table C.5 Memory Allocated on Average (in kB).

C.4 Applications in control theory

We have already discussed the necessity of efficiently solving the optimization problem in `handleObstacles` and the inclusion test in Line 4 of Algorithm 8.9, which can be addressed by Corollary C.8 and Algorithm C.1, respectively, to construct smart abstractions of systems.

In this section, we illustrate one another possible application in the control theory of linear time-invariant (LTI) systems, namely, the computation of a control forward-invariant set for additive disturbances. This problem has been extensively studied in the literature (e.g., see [Bla99, Section 4.1]) and, therefore, the goal of this section is not to provide a new method to tackle it but rather to demonstrate the results from Corollary C.8.

For that, consider an LTI system

$$\dot{x} = \mathbf{A}x + \mathbf{B}u + \mathbf{H}w \tag{C.31}$$

where $x(t) \in \mathbb{R}^2$ is the state variable, and the signals $u(t)$, $w(t) \in \mathbb{R}$ are the control and the additive disturbance. This system is defined by the matrices

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 0.1 & 0.3 \end{pmatrix}, \ \mathbf{B} = \begin{pmatrix} 0 \\ 0.5 \end{pmatrix}, \ \mathbf{H} = \begin{pmatrix} -0.3 \\ 0.6 \end{pmatrix} \tag{C.32}$$

and is controlled by a state-feedback LQR controller $u(t) = -\mathbf{K}x(t)$ synthesized for LQR parameters $\mathbf{Q} = \mathbf{I}$ and $R = 1$, see [BEGFB94, p. 114] for details on the LQR problem. The corresponding solution of the Algebraic Riccati Equation and feedback gain are

$$\mathbf{P} = \begin{pmatrix} 36.10 & 42.36 \\ 42.36 & 72.98 \end{pmatrix}, \ \mathbf{K} = \begin{pmatrix} 4.24 & 7.30 \end{pmatrix}, \tag{C.33}$$

which allows us to define the closed loop matrix $\mathbf{A}_c = \mathbf{A} - \mathbf{B}\mathbf{K}$ and a Lyapunov function $v(x) = x^\top \mathbf{P} x$.

Bounded additive disturbances: Due to the additive disturbance, the descent condition for this Lyapunov function does not hold everywhere. For an arbitrary $w \in \mathcal{W} \subset \mathbb{R}$, this can be verified as

$$\begin{aligned} \dot{v}(x, w) &:= 2x^\top (\mathbf{P}\mathbf{A}_c)x + 2x^\top \mathbf{P}\mathbf{H}w \\ &= -(x - \mathbf{G}w)^\top \mathbf{S}(x - \mathbf{G}w) + r(w) \end{aligned} \tag{C.34}$$

where $\mathbf{S} = -\mathbf{A}_c^\top \mathbf{P} - \mathbf{P}\mathbf{A}_c$, $\mathbf{G} = \mathbf{S}^{-1}\mathbf{P}\mathbf{H}$ and $r(w) = w^\top \mathbf{G}^\top \mathbf{S}\mathbf{G} w$. From (C.34), one can conclude that $\dot{v}(x,w) \geq 0$ if and only if $x \in \mathrm{E}(\mathbf{G}w, r(w)^{-1}\mathbf{S})$. Due to the linearity of $\dot{v}(x,w)$ with respect to w, when the disturbance $w(t)$ takes values from a polytope $\mathcal{W} = \mathrm{co}\{w_1, \ldots, w_N\}$, the problem of finding the smallest sublevel set of $v(x)$ that is control forward-invariant reduces to finding the smallest $\gamma \geq 0$ such that

$$\mathcal{B}_i := \mathrm{E}(\mathbf{G}w_i, r(w_i)^{-1}\mathbf{S}) \subseteq \mathcal{V} := \mathrm{E}(0, \gamma^{-1}\mathbf{P})$$

for $i \in [1; N]$. In this context, Corollary C.8 allows us to calculate γ_i such that $\mathcal{B}_i \subseteq_0 \mathcal{V}_i := \mathrm{E}(0, \gamma_i^{-1}\mathbf{P})$ by comparison with the inclusion given in equation (C.27). Hence, we have that $\gamma = \max(\gamma_1, \ldots, \gamma_N)$ is ensured to be the smallest such that $\mathcal{B}_i \subseteq \mathcal{V}$ for $i \in [1; N]$. Considering $\mathcal{W} = [-0.5,\ 0.5]$, we obtain $\gamma = 1.137$. Notice that, for obtaining all $\gamma_1, \ldots, \gamma_N$, the Cholesky decomposition and the spectral decomposition associated with the definition of the function (C.11) and the variable transformation (C.2) can be computed only once, given that the matrices defining the shape of all $\mathcal{B}_i$ are scalar multiples of $\mathbf{S}$. Moreover, the symmetry of $\mathcal{V}$ ensures that $\mathrm{E}(\mathbf{G}w_i, r(w_i)^{-1}\mathbf{S}) \subseteq \mathcal{V}$ implies that $\mathrm{E}(-\mathbf{G}w_i, r(w_i)^{-1}\mathbf{S}) \subseteq \mathcal{V}$, which helps to reduce the number of executions of the bisection algorithm to calculate $\ell^*_{c,\mathbf{P}}$ in Corollary C.8.

In Figure C.3, the ellipsoidal sets defined in this section are illustrated, along with a trajectory undergoing a random disturbance and starting at $x_0 = [-1 \quad -1]^\top$. The light blue shaded area represents the set $\mathrm{co}\{\mathcal{B}_1, \mathcal{B}_2\}$, where the decreasing property of the Lyapunov function fails for some $w \in \mathcal{W}$.

In summary, Corollary C.8 allows the verification that the Lyapunov function $v(x) = x^\top \mathbf{P} x$ strictly decreases in $\mathbb{R}^2 \setminus \mathcal{V}$ despite the persistent disturbance. This verification is done efficiently by performing the following numerical operations: one Cholesky decomposition, one spectral decomposition, and one bisection algorithm for maximizing a concave scalar function in a compact interval.

C.5 Summary

We presented a new method to verify the inclusion of n-ellipsoids, which consists in the maximization of a scalar concave and smooth function (C.11).

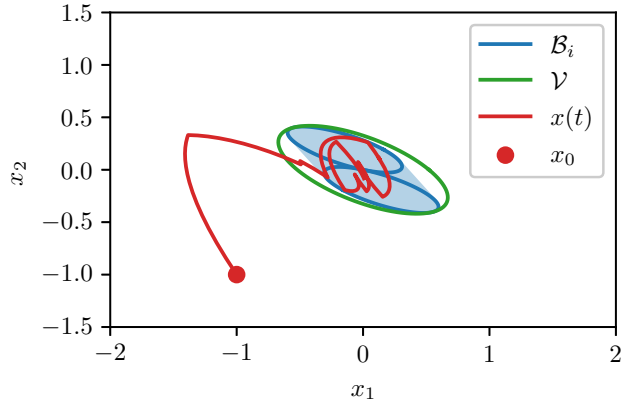


Fig. C.3 For a disturbed LTI system presented in Section C.4, we computed with the results from Corollary C.8 the smallest forward-invariant level set $\mathcal{V}$ of the Lyapunov function $v(x)$.

This function and its derivatives can be computed in $\mathcal{O}(n)$ floating-point operations and the interval (C.24) where its maximum lies is a subset of $[0, 1]$. Therefore, we proposed a bisection-based algorithm (Algorithm C.2) allowing us to decide whether the inclusion holds. A benchmark with methods based on LMI constraints tackled by two off-the-shelf SDP solvers is carried out, showing that we outperform the LMI-based approach. We also present an application in the field of control theory.

Bibliography

[AD94] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[AGS13] Leonhard Asselborn, Dominic Gross, and Olaf Stursberg. Control of uncertain nonlinear systems using ellipsoidal reachability calculus. *IFAC Proceedings Volumes*, 46(23):50–55, 2013.

[AHKV98] Rajeev Alur, Thomas A Henzinger, Orna Kupferman, and Moshe Y Vardi. Alternating refinement relations. In *CONCUR'98 Concurrency Theory: 9th International Conference Nice, France, September 8–11, 1998 Proceedings 9*, pages 163–178. Springer, 1998.

[Alt10] Matthias Althoff. *Reachability analysis and its application to the safety assessment of autonomous cars*. PhD thesis, Technische Universität München, 2010. URL: https://mediatum.ub.tum.de/963752.

[Alu15] Rajeev Alur. *Principles of cyber-physical systems*. MIT press, 2015.

[ALZ23] Daniel Ajeleye, Abolfazl Lavaei, and Majid Zamani. Data-driven controller synthesis via finite abstractions with formal guarantees. *IEEE Control Systems Letters*, 7:3453–3458, 2023.

[ÅM07] Karl Johan Åström and Richard M Murray. Feedback systems. *An Introduction for Scientists and Engineers, Karl Johan Aström and Richard M Murray*, pages 27–64, 2007.

[Ang02] David Angeli. A Lyapunov approach to incremental stability properties. *IEEE Transactions on Automatic Control*, 47(3):410–421, 2002.

[APGCZ20] W Alejandro Apaza-Perez, Antoine Girard, Christophe Combastel, and Ali Zolghadri. Symbolic observer-based controller for uncertain nonlinear systems. *IEEE Control Systems Letters*, 5(4):1297–1302, 2020.

[ApS19] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.* 2019. URL: `http://docs.mosek.com/9.0/toolbox/index.html`.

[BDPT22] Andrea Bisoffi, Claudio De Persis, and Pietro Tesi. Data-driven control via Petersen's lemma. *Automatica*, 145:110537, 2022.

[BEGFB94] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[Bel57] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957.

[Bel58] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[Ber05] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2005.

[Ber07] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume II*, volume 2. Athena scientific, 2007.

[BGVKS22] Jess Banks, Jorge Garza-Vargas, Archit Kulkarni, and Nikhil Srivastava. Pseudospectral shattering, the sign function, and diagonalization in nearly matrix multiplication time. *Foundations of Computational Mathematics*, pages 1–89, 2022.

[BJP+12] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Saar. Synthesis of reactive (1) designs. *Journal of Computer and System Sciences*, 78(3):911–938, 2012.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[Bla99] Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[BM99] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.

[BPDB18] Alessandro Borri, Giordano Pola, and Maria Domenica Di Benedetto. Design of symbolic controllers for networked control systems. *IEEE Transactions on Automatic Control*, 64(3):1034–1046, 2018.

[BPM05] A Giovanni Beccuti, Georgios Papafotiou, and Manfred Morari. Optimal control of the boost dc-dc converter. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 4457–4462. IEEE, 2005.

[BR71] Dimetri Bertsekas and Ian Rhodes. Recursive state estimation for a set-membership description of uncertainty. *IEEE Transactions on Automatic Control*, 16(2):117–128, 1971.

[BRAJ23] Adrien Banse, Licio Romao, Alessandro Abate, and Raphaël M. Jungers. Data-driven memory-dependent abstractions of dynamical systems. In *Learning for Dynamics and Control Conference*, pages 891–902. PMLR, 2023.

[Bry92] Randal E Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys (CSUR)*, 24(3):293–318, 1992.

[BT96] Dimitri Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[BYG17] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Formal methods for discrete-time dynamical systems*, volume 89. Springer, 2017.

[CBLJ24a] Julien Calbert, Adrien Banse, Benoît Legat, and Raphaël M. Jungers. Dionysos. jl: a modular platform for smart symbolic control. *arXiv preprint arXiv:2404.14114*, 2024.

[CBLJ24b] Julien Calbert, Adrien Banse, Benoît Legat, and Raphaël M. Jungers. Dionysos.jl: a modular platform for smart symbolic control, 2024. doi:10.24433/CO.6327570.V2.

[CEJ23] Julien Calbert, Lucas N. Egidio, and Raphaël M. Jungers. An efficient method to verify the inclusion of ellipsoids. *IFAC-PapersOnLine*, 56(2):1958–1963, 2023.

[CEJ24] Julien Calbert, Lucas N. Egidio, and Raphaël M. Jungers. Smart abstraction based on iterative cover and non-uniform cells. *IEEE Control Systems Letters*, 2024.

[CGG11a] Javier Camara, Antoine Girard, and Gregor Gössler. Safety controller synthesis for switched systems using multi-scale symbolic models. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 520–525. IEEE, 2011.

[CGG11b] Javier Cámara, Antoine Girard, and Gregor Gössler. Synthesis of switching controllers using approximately bisimilar multiscale abstractions. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, HSCC '11. ACM, April 2011.

[CGJ24] Julien Calbert, Antoine Girard, and Raphaël M Jungers. Classification of simulation relations for symbolic control. *arXiv preprint arXiv:2410.06083*, 2024.

[CJ23] Julien Calbert and Raphaël M. Jungers. Data-driven heuristic symbolic models and application to limit-cycle detection. In *2023 American Control Conference (ACC)*, pages 4351–4356. IEEE, 2023.

[CL08] Christos G Cassandras and Stéphane Lafortune. *Introduction to discrete event systems*. Springer, 2008.

[Cla97] Edmund M Clarke. Model checking. In *Foundations of Software Technology and Theoretical Computer Science: 17th Conference Kharagpur, India, December 18–20, 1997 Proceedings 17*, pages 54–56. Springer, 1997.

[CLEJ21] Julien Calbert, Benoît Legat, Lucas N. Egidio, and Raphaël M. Jungers. Alternating simulation on hierarchical abstractions.

In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 593–598. IEEE, 2021.

[CLRS22] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[CMGJ24] Julien Calbert, Sébastien Mattenet, Antoine Girard, and Raphaël M. Jungers. Memoryless concretization relation. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, pages 1–9, 2024.

[CPMJ22] Rudi Coppola, Andrea Peruffo, and Manuel Mazo Jr. Data-driven abstractions for verification of deterministic systems. *arXiv preprint arXiv:2211.01793*, 2022.

[CPMJ24] Rudi Coppola, Andrea Peruffo, and Manuel Mazo Jr. Data-driven abstractions for control systems. *arXiv preprint arXiv:2402.10668*, 2024.

[dAGMJ21] Gabriel de A. Gleizer and Manuel Mazo Jr. Computing the sampling performance of event-triggered control. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–7, 2021.

[DAHM01] Luca De Alfaro, Thomas A Henzinger, and Rupak Majumdar. Symbolic algorithms for infinite-state games. In *CONCUR 2001—Concurrency Theory: 12th International Conference Aalborg, Denmark, August 20–25, 2001 Proceedings 12*, pages 536–550. Springer, 2001.

[DCDVL13] Eric Dallal, Alessandro Colombo, Domitilla Del Vecchio, and Stéphane Lafortune. Supervisory control for collision avoidance in vehicular networks with imperfect measurements. In *52nd IEEE Conference on Decision and Control*, pages 6298–6303. IEEE, 2013.

[DFVR03] Daniela Pucci De Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6):850–865, 2003.

[DHL17] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM review*, 59(2):295–320, 2017.

[Dij22] Edsger W Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: his life, work, and legacy*, pages 287–290. 2022.

[DM07] Alexandre Donzé and Oded Maler. Systematic simulation using sensitivity analysis. In *International Workshop on Hybrid Systems: Computation and Control*, pages 174–189. Springer, 2007.

[Don07] Alexandre Donzé. *Trajectory-based verification and controller Synthesis for continuous and hybrid systems*. PhD thesis, Université Joseph-Fourier-Grenoble I, 2007.

[DSA21] Alex Devonport, Adnane Saoud, and Murat Arcak. Symbolic abstractions from data: A pac learning approach. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 599–604. IEEE, 2021.

[ELJ22] Lucas N. Egidio, Thiago Alves Lima, and Raphaël M. Jungers. State-feedback abstractions for optimal control of piecewise-affine systems. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 7455–7460. IEEE, 2022.

[FMP19] Mahyar Fazlyab, Manfred Morari, and George J Pappas. Probabilistic verification and reachability analysis of neural networks via semidefinite programming. In *IEEE Conference on Decision and Control (CDC)*, pages 2726–2731, 2019.

[GGM15] Antoine Girard, Gregor Gössler, and Sebti Mouelhi. Safety controller synthesis for incrementally stable switched systems using multiscale symbolic models. *IEEE Transactions on Automatic Control*, 61(6):1537–1549, 2015.

[GH12] Igor Gilitschenski and Uwe D Hanebeck. A robust computational test for overlap of two arbitrary-dimensional ellipsoids in fault-detection of Kalman filters. In *IEEE International Conference on Information Fusion*, pages 396–401, 2012.

[Gir12] Antoine Girard. Controller synthesis for safety and reachability via approximate bisimulation. *Automatica*, 48(5):947–953, 2012.

[Gir13] Antoine Girard. Low-complexity quantized switching controllers using approximate bisimulation. *Nonlinear Analysis: Hybrid Systems*, 10:34–44, 2013.

[Gir14] Antoine Girard. Approximately bisimilar abstractions of incrementally stable finite or infinite dimensional systems. In *53rd IEEE conference on decision and control*, pages 824–829. IEEE, 2014.

[GJ07] Lars Grune and Oliver Junge. Approximately optimal nonlinear stabilization with preservation of the lyapunov function property. In *2007 46th IEEE Conference on Decision and Control*, pages 702–707. IEEE, 2007.

[GLB14] Ebru Aydin Gol, Mircea Lazar, and Calin Belta. Language-guided controller synthesis for linear systems. *IEEE Transactions on Automatic Control*, 59(5):1163–1176, May 2014.

[GLPN93] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. Directed hypergraphs and applications. *Discrete applied mathematics*, 42(2-3):177–201, 1993.

[GP06] Antoine Girard and George J Pappas. Verification using simulation. In *International Workshop on Hybrid Systems: Computation and Control*, pages 272–286. Springer, 2006.

[GPT09] Antoine Girard, Giordano Pola, and Paulo Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Transactions on Automatic Control*, 55(1):116–126, 2009.

[Har02] Philip Hartman. *Ordinary differential equations*. SIAM, 2002.

[HAVdH15] Sofie Haesaert, Alessandro Abate, and Paul MJ Van den Hof. Correct-by-design output feedback of lti systems. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 6159–6164. IEEE, 2015.

[Hig09] Nicholas J Higham. Cholesky factorization. *Wiley interdisciplinary reviews: computational statistics*, 1(2):251–254, 2009.

[HJMS02] Thomas A Henzinger, Ranjit Jhala, Rupak Majumdar, and Gregoire Sutre. Lazy abstraction. In *Proceedings of the 29th*

ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pages 58–70, 2002.

[HKPV95] Thomas A Henzinger, Peter W Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 373–382, 1995.

[HLTS20] Binghan He, Jaemin Lee, Ufuk Topcu, and Luis Sentis. BP-RRT: Barrier pair synthesis for temporal logic motion planning. In *IEEE Conference on Decision and Control (CDC)*, pages 1404–1409, 2020.

[HMMS18a] Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. Lazy abstraction-based control for safety specifications. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4902–4907. IEEE, 2018.

[HMMS18b] Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. Multi-layered abstraction-based controller synthesis for continuous-time systems. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, pages 120–129, 2018.

[HMMS19] Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. Lazy abstraction-based controller synthesis. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2019.

[HNR68] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[IN89] Giuseppe F Italiano and Umberto Nanni. Online maintenance of minimal directed hypergraphs. Technical Report CUCS-435-89, Department of Computer Science, Columbia University Series, 1989.

[JLFL21] John Jackson, Luca Laurenti, Eric Frew, and Morteza Lahijanian. Strategy synthesis for partially-known switched

stochastic systems. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

[KF11] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[KG12] Ratnesh Kumar and Vijay K Garg. *Modeling and control of logical discrete event systems*, volume 300. Springer Science & Business Media, 2012.

[Kha96] H. Khalil. *Nonlinear Systems*. Englewood Cliffs, NJ: Prentice Hall, 1996.

[KK12] Kyoung-Dae Kim and Panganamala R Kumar. Cyber–physical systems: A perspective at the centennial. *Proceedings of the IEEE*, 100(Special Centennial Issue):1287–1308, 2012.

[KL00] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation. Symposia Proceedings*, volume 2, 2000.

[KMS+24] Milad Kazemi, Rupak Majumdar, Mahmoud Salamati, Sadegh Soudjani, and Ben Wooding. Data-driven abstraction-based control synthesis. *Nonlinear Analysis: Hybrid Systems*, 52:101467, 2024.

[KV97] Alexander Kurzhanski and István Vályi. *Ellipsoidal calculus for estimation and control*. Springer, 1997.

[KV01] Orna Kupferman and Moshe Y Vardi. Model checking of safety properties. *Formal methods in system design*, 19:291–314, 2001.

[L+98] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.

[LBJ21] Benoît Legat, Jean Bouchat, and Raphaël M. Jungers. Abstraction-based branch and bound approach to q-learning for hybrid optimal control. In *Proceedings of the 3rd Annual Learning for Dynamics & Control Conference*, 2021.

[LDD+23] Miles Lubin, Oscar Dowson, Joaquim Dias Garcia, Joey Huchette, Benoît Legat, and Juan Pablo Vielma. JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, 15:581–589, 2023.

[LDGL22] Benoît Legat, Oscar Dowson, Joaquim Dias Garcia, and Miles Lubin. Mathoptinterface: a data structure for mathematical optimization problems. *INFORMS Journal on Computing*, 34(2):672–689, 2022.

[LFS+24] Benoît Legat, Marcelo Forets, Christian Schilling, kpotomkin, and Julia TagBot. blegat/HybridSystems.jl: v0.4.3, 2024.

[LKJ01] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20, 2001.

[LPY01] Gerardo Lafferriere, George J Pappas, and Sergio Yovine. Symbolic reachability computation for families of linear vector fields. *Journal of Symbolic Computation*, 32(3):231–253, 2001.

[LS98] Winfried Lohmiller and Jean-Jacques E Slotine. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998.

[LS16] Edward Ashford Lee and Sanjit Arunkumar Seshia. *Introduction to embedded systems: A cyber-physical systems approach*. MIT press, 2016.

[Lue71] David Luenberger. An introduction to observers. *IEEE Transactions on automatic control*, 16(6):596–602, 1971.

[Man60] Alan S Manne. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960.

[MGG13] Sebti Mouelhi, Antoine Girard, and Gregor Gössler. CoSyMA: a tool for controller synthesis using multi-scale abstractions. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 83–88, 2013.

[MJDT10] Manuel Mazo Jr, Anna Davitian, and Paulo Tabuada. Pessoa: A tool for embedded controller synthesis. In *International conference on computer aided verification*, pages 566–569. Springer, 2010.

[MLL24] Frederik Baymler Mathiesen, Morteza Lahijanian, and Luca Laurenti. IntervalMDP.jl: Accelerated Value Iteration for Interval Markov Decision Processes, 2024.

[MN21] Aniketh Manjunath and Quan Nguyen. Safe and robust motion planning for dynamic robotics via control barrier functions. In *60th IEEE Conference on Decision and Control (CDC)*, 2021.

[MNA03] P Madhusudan, Wonhong Nam, and Rajeev Alur. Symbolic computational techniques for solving games. *Electronic Notes in Theoretical Computer Science*, 89(4):578–592, 2003.

[MOM14] Oscar Mickelin, Necmiye Ozay, and Richard M Murray. Synthesis of correct-by-construction control protocols for hybrid systems using partial state information. In *2014 American Control Conference*, pages 2305–2311. IEEE, 2014.

[MOS20] Rupak Majumdar, Necmiye Ozay, and Anne-Kathrin Schmuck. On abstraction-based controller design with output feedback. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2020.

[MU18] Masashi Mizoguchi and Toshimitsu Ushio. Deadlock-free output feedback controller design based on approximately abstracted observers. *Nonlinear Analysis: Hybrid Systems*, 30:58–71, 2018.

[N+18] Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

[NW99] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[Osi06] George Osipenko. *Dynamical systems, graphs, and algorithms*. Springer, 2006.

[OWB13] Brendan O'Donoghue, Yang Wang, and Stephen Boyd. Iterated approximate value functions. In *2013 European Control Conference (ECC)*, pages 3882–3888. IEEE, 2013.

[PDBB19] Giordano Pola, Maria Domenica Di Benedetto, and Alessandro Borri. Symbolic control design of nonlinear systems with outputs. *Automatica*, 109:108511, 2019.

[PJ04] Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *International Workshop on Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.

[Pra06] Stephen Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42(1):117–126, 2006.

[PT07] Imre Pólik and Tamás Terlaky. A survey of the S-lemma. *SIAM review*, 49(3):371–418, 2007.

[RCJ21] Wei Ren, Julien Calbert, and Raphaël M. Jungers. Zonotope-based controller synthesis for ltl specifications. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 580–585. IEEE, 2021.

[RD19] Wei Ren and Dimos V Dimarogonas. Dynamic quantization based symbolic abstractions for nonlinear control systems. In *IEEE 58th Conference on Decision and Control (CDC)*, pages 4343–4348, 2019.

[Rei11] Gunther Reißig. Computing abstractions of nonlinear systems. *IEEE Transactions on Automatic Control*, 56(11):2583–2598, 2011.

[Roc70] R Tyrrell Rockafellar. *Convex analysis*, volume 18. Princeton university press, 1970.

[RST02] Lluís Ros, Assumpta Sabater, and Federico Thomas. An ellipsoidal calculus based on propagation and fusion. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 32(4):430–442, 2002.

[RTM11] Pritam Roy, Paulo Tabuada, and Rupak Majumdar. Pessoa 2.0: a controller synthesis tool for cyber-physical systems. In

Proceedings of the 14th international conference on Hybrid systems: computation and control, HSCC '11. ACM, April 2011.

[RWR16] Gunther Reissig, Alexander Weber, and Matthias Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62(4):1781–1796, 2016.

[RZ16] Matthias Rungger and Majid Zamani. Scots: A tool for the synthesis of symbolic controllers. In *Proceedings of the 19th international conference on hybrid systems: Computation and control*, pages 99–104, 2016.

[Sch68] Fred Schweppe. Recursive state estimation: Unknown but bounded errors and system inputs. *IEEE Transactions on Automatic Control*, 13(1):22–28, 1968.

[SR14] Anne-Kathrin Schmuck and Jörg Raisch. Asynchronous l-complete approximations. *Systems & Control Letters*, 73:67–75, 2014.

[SS85] Paul J Schweitzer and Abraham Seidmann. Generalized polynomial approximations in markovian decision processes. *Journal of mathematical analysis and applications*, 110(2):568–582, 1985.

[ST99] Jeff S Shamma and Kuang-Yang Tu. Set-valued observers and optimal disturbance rejection. *IEEE Transactions on Automatic Control*, 44(2):253–264, 1999.

[STR15] Anne-Kathrin Schmuck, Paulo Tabuada, and Jörg Raisch. Comparing asynchronous l-complete approximations and quotient based abstractions. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 6823–6829. IEEE, 2015.

[Tab09] Paulo Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

[TI09] Yuichi Tazaki and Jun-ichi Imura. Discrete-state abstractions of nonlinear systems using multi-resolution quantizer. In *International Workshop on Hybrid Systems: Computation and Control*, pages 351–365. Springer, 2009.

[TRK16] Duc N Tran, Björn S Rüffer, and Christopher M Kellett. Incremental stability properties for discrete-time systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 477–482. IEEE, 2016.

[Var95] Moshe Y Vardi. An automata-theoretic approach to fair realizability and synthesis: Preliminary report. In *Computer Aided Verification: 7th International Conference, CAV'95 Liège, Belgium, July 3–5, 1995 Proceedings 7*, pages 267–278. Springer, 1995.

[Vid81] Mathukumalli Vidyasagar. *Input-output analysis of large-scale interconnected systems: decomposition, well-posedness and stability*. Springer, 1981.

[WL24] Ben Wooding and Abolfazl Lavaei. IMPaCT: Interval MDP Parallel Construction for Controller Synthesis of Large-Scale Stochastic Systems, 2024.

[WLS+22] Albert Wu, Thomas Lew, Kiril Solovey, Edward Schmerling, and Marco Pavone. Robust-rrt: Probabilistically-complete motion planning for uncertain nonlinear systems. *arXiv preprint arXiv:2205.07728*, 2022.

[WN+99] Stephen Wright, Jorge Nocedal, et al. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

[WOB15] Yang Wang, Brendan O'Donoghue, and Stephen Boyd. Approximate dynamic programming via iterated bellman inequalities. *International Journal of Robust and Nonlinear Control*, 25(10):1472–1496, 2015.

[WR14] Alexander Weber and Gunther Reissig. Classical and strong convexity of sublevel sets and application to attainable sets of nonlinear systems. *SIAM Journal on Control and Optimization*, 52(5):2857–2876, 2014.

[YFN+10] Makoto Yamashita, Katsuki Fujisawa, Kazuhide Nakata, Maho Nakata, Mituhiro Fukuda, Kazuhiro Kobayashi, and Kazushige Goto. A high-performance software package for semidefinite programs: SDPA 7. *Tokyo, Japan*, 2010.

[YTC+11] Boyan Yordanov, Jana Tumova, Ivana Cerna, Jiří Barnat, and Calin Belta. Temporal logic control of discrete-time piecewise affine systems. *IEEE Transactions on Automatic Control*, 57(6):1491–1504, 2011.

[ZAZ24] Bingzhuo Zhong, Murat Arcak, and Majid Zamani. Hierarchical control for cyber-physical systems via general approximate alternating simulation relations. *IFAC-PapersOnLine*, 58(11):13–18, 2024.

[Zol96] Ali Zolghadri. An algorithm for real-time failure detection in Kalman filters. *IEEE Transactions on Automatic Control*, 41(10):1537–1539, 1996.

[ZPMT11] Majid Zamani, Giordano Pola, Manuel Mazo, and Paulo Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, 57(7):1804–1809, 2011.