Formal synthesis of analytic controllers

An evolutionary approach

Verdier, C.F.

DOI
10.4233/uuid:70f6704f-30e4-4e1a-8c74-9fe2b699a80d

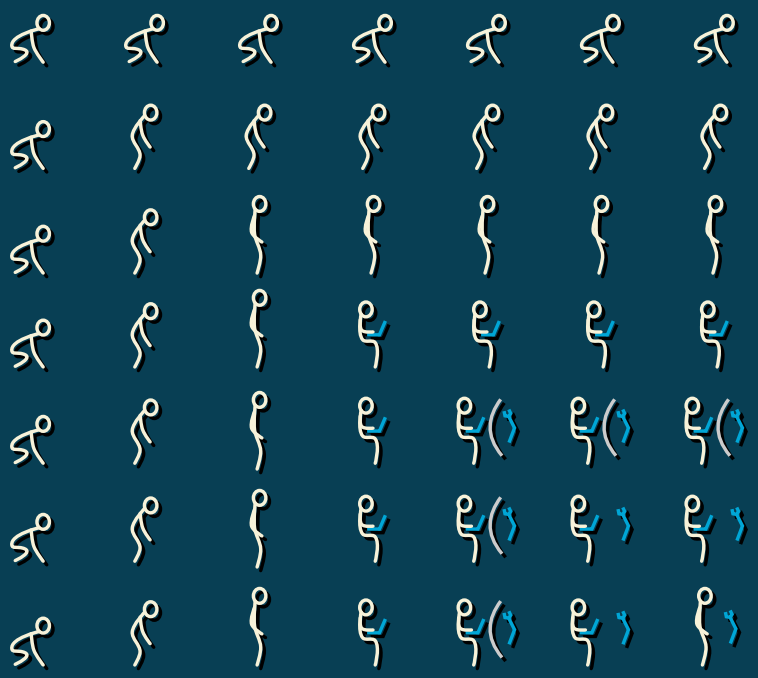
Publication date
2020

Document Version
Final published version

Citation (APA)
Verdier, C. F. (2020). *Formal synthesis of analytic controllers: An evolutionary approach*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:70f6704f-30e4-4e1a-8c74-9fe2b699a80d

Formal Synthesis of Analytic Controllers

An Evolutionary Approach



Cees Ferdinand Verdier

Formal synthesis of analytic controllers

An evolutionary approach

Formal synthesis of analytic controllers

An evolutionary approach

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus Prof. dr. ir. T.H.J.J. van der Hagen;
chair of the Board of Doctorates
to be defended publicly on
Wednesday 21 October 2020 at 15:00 o'clock

by

Cees Ferdinand Verdier

Master of Science in Systems and Control,
Delft University of Technology, the Netherlands
born in Heemskerk, the Netherlands

This dissertation has been approved by the promoters.

Composition of the doctoral committee:

Rector Magnificus	chairperson
Dr. M. Mazo Espinosa	Delft University of Technology, promotor
Prof. dr. R. Babuška	Delft University of Technology, copromotor
Independent members:	
Prof. dr. ir. B. H. K. de Schutter	Delft University of Technology
Prof. dr. P. A. N. Bosman	Delft University of Technology
Prof. dr. A. Abate	University of Oxford, United Kingdom
Prof. Dr.-Ing. M. Althoff	Technical University of Munich, Germany
Dr. ir. R. Toth	Eindhoven University of Technology

This research is supported by the Dutch Organization for Scientific Research (NWO, domain TTW, grant: 13852) which is partly funded by the Ministry of Economic Affairs.

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

Keywords: Formal controller synthesis, hybrid systems, temporal logic, genetic programming, Lyapunov methods, reachability analysis

Printed by: Print Service Ede

Front & Back: Cees F. Verdier

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

This dissertation is dedicated to my parents, for their great dedication to my education.

"Every now and then a man's mind is stretched by a new idea or sensation, and never shrinks back to its former dimensions

-Oliver Wendell Homes Sr.

Contents

Summary

Modern technology has resulted in a widespread availability of advanced hardware and computation power, enabling the increase of automation. However, control design for safety-critical cyber-physical systems still requires significant expert knowledge, hampering large scale automation. Cyber-physical systems typically exhibit both continuous and discrete behavior, and are therefore paradigmatic examples of hybrid systems. Their (safety-critical) specifications, which can be formalized using e.g. temporal logic, go beyond classical system properties such as stability. While there exist constructive design methods for certain sub-classes of systems and specifications, for general hybrid systems with temporal logic specifications, methods are still lacking. Nevertheless, in recent years multiple approaches have been proposed to automatically synthesize correct-by-construction controllers, i.e. controllers that are guaranteed to satisfy a pre-defined specification by their synthesis method. However, typically these approaches suffer from one or more of the following disadvantages: the method relies on discretization of the state space and therefore suffers from the curse of dimensionality; the resulting controllers are in the form of enormous look-up tables, hence impractical for implementation in embedded hardware; the method relies on online optimization and therefore has a high computation cost; the method is only applicable for a limited class of systems or specifications; or the methodology is highly dependent on expert knowledge.

The goal of this thesis is to propose a novel approach that overcomes these limitations. That is, our goal is to propose a framework for automatic controller synthesis, capable of synthesizing closed-form controllers for hybrid systems with temporal logic specifications, without a heavy reliance on expert knowledge. To this end, we draw inspiration from the human design process, and utilize two methods that show great similarities to it, namely evolutionary algorithms and counterexample-guided inductive synthesis.

More specifically, in this work we use genetic programming (GP), an evolutionary algorithm which is capable of evolving entire programs, in our case controllers. This makes it possible to automatically discover the structure of a solution, rather than being dependent on the user to supply an adequate template solution. Moreover, it enables the synthesis of compact closed-form controllers, circumventing the need for look-up tables or online optimization. While GP can be used to discover the controller structures from scratch, the use of expert knowledge does improve the convergence to a solution. To enable such provision of expert knowledge, we use grammar-guided genetic programming, a variant that restricts candidate solutions to adhere to a user-defined grammar. Nevertheless, the use of expert knowledge remains optional; a user can provide a very general grammar, or use their expertise to bias the search direction.

In combination with GP, we use the concept of counterexample-guided inductive synthesis (CEGIS) to refine candidate solutions based on counterexamples, until the controller is guaranteed to satisfy the desired specification. In this thesis we propose two CEGIS-based synthesis frameworks, which differ in the employed verification paradigms. The

first approach uses an *indirect* method, namely certificate functions, whereas the second approach uses a *direct* method, namely reachability analysis.

The first framework proposed in this thesis co-synthesises both controllers and certificate functions, where the latter is used to (indirectly) verify the desired system specification. We propose a novel Lyapunov barrier function (LBF) which, if it exists, implies a reach-avoid property. The LBF is defined such that its conditions are verifiable by means of a satisfiability modulo theories (SMT) solver; a tool capable of determining whether a first-order logic formula is satisfied or not. We use genetic programming to synthesize pairs of candidate controller and candidate LBF based on a finite number of samples of the state space. Synthesized pairs meeting the LBF conditions over the finite set of states are subsequently formally verified by means of an SMT solver. If the LBF conditions are not met, a counterexample is extracted, which is used to refine the synthesis procedure. This first methodology is applied to general hybrid systems modelled as jump-flow systems, subjected to reach-avoid specifications. Additionally, we propose a specialized framework for smooth nonlinear systems with sampled-data controllers, based on *control* Lyapunov barrier functions. Finally, we demonstrate how the proposed framework can be used to verify (near) optimal controllers, which are obtained by means of reinforcement learning.

The second framework relies on the reachability analysis of the system. To this end we use recent advances on model checking for signal temporal logic (STL) and counterexample generation based on reachability analysis. STL reasons over singular trajectories, whereas reachability analysis returns reachable sets. To bridge the gap between singular trajectories and reachable sets, we use a sound transformation from STL to reachset temporal logic (RTL), which directly reasons over reachability sets. To quantify the satisfaction of an RTL formula, we introduce the quantitative semantics of RTL, which provides an optimization criterion that is used in our synthesis. We use genetic programming to optimize controllers, based on a finite set of simulated trajectories. Controllers which satisfy the specification for this finite number of trajectories are subsequently verified with respect to the entire initial set by means of reachability analysis. If the specification is violated based on the reachability analysis, a corresponding initial condition resulting in the violation is extracted. This counterexample is then used to refine the controller synthesis. This second methodology is applied to nonlinear systems with a sampled-data implementation of the controller, subjected to general STL specifications.

We demonstrate the effectiveness of both approaches on multiple (academic) case studies. The proposed frameworks are best suited for different use cases; the certificate-based approach is best suited for low-dimensional systems with large initial sets, whereas the reachability-based approach is best suited for higher-dimensional systems with small initial sets, subjected to intricate specifications in the form of temporal logic. While the two presented frameworks deal with either general hybrid systems or general temporal logic specifications, we propose future extensions to general hybrid systems subjected to general temporal logic. Both frameworks result in correct-by-construction compact closed-form controllers, where the use of expert knowledge is optional. Their capability to synthesize sampled-data controllers enables easy implementation in embedded hardware with limited memory and computation power, forming a stepping stone towards faster automation.

Samenvatting

Moderne technologie heeft geresulteerd in de wijdverspreide beschikbaarheid van geavanceerde hardware en rekenkracht, waardoor de automatisering toe kan nemen. Echter, het ontwerpen van regelaars voor veiligheidskritische cyberfysieke systemen vereist nog steeds aanzienlijke expertise, waardoor automatisering op grote schaal wordt geremd. Cyberfysieke systemen vertonen doorgaans zowel continu als discreet gedrag, en zijn daarom typische voorbeelden van hybride systemen. De bijbehorende (veiligheidskritieke) specificaties, die bijvoorbeeld kunnen worden geformaliseerd met temporele logica, gaan voorbij klassieke systeemeigenschappen zoals stabiliteit. Hoewel er constructieve ontwerpmethoden bestaan voor bepaalde subklassen van systemen, zijn deze methoden voor algemene hybride systemen met temporele logica specificaties niet voorhanden. Desondanks zijn er in de afgelopen jaren meerdere methoden voorgesteld voor het automatisch synthetiseren van 'correct-door-constructie' regelaars, d.w.z. regelaars die dankzij de synthesemethode aan vooraf gedefinieerde specificaties gegarandeerd voldoen. Deze methoden hebben echter doorgaans een of meer van de volgende tekortkomingen: de methode is afhankelijk van discretisatie en leidt daarom aan de vloek van dimensionaliteit; de resulterende regelaar heeft de vorm van een enorme opzoektabel en is daarom onpraktisch om te implementeren in *embedded hardware*; de methode is afhankelijk van online optimalisatie en heeft daarom een hoge rekenkracht; de methode is enkel van toepassing op een beperkte set systemen of specificaties; of de methode is zeer afhankelijk van expertise.

Het doel van dit proefschrift is het voorstellen van een nieuwe aanpak, die deze beperkingen niet heeft. Met andere woorden, het doel is om een methode voor te stellen voor het automatisch synthetiseren van regelaars, die in staat is gesloten-vorm regelaars voor hybride systemen met temporele-logicaspecificaties te ontwerpen, zonder een sterke afhankelijkheid van expertise. Om dit te bewerkstelligen, combineren we methoden die een grote overeenkomst vertonen met het menselijke ontwerpproces, namelijk evolutionaire algoritmes en tegenvoorbeeld-gestuurde inductieve synthese.

Specifiek gezien, gebruiken we in dit werk genetisch programmeren (GP), een evolutionair algoritme, dat in staat is om gehele programma's te evolueren, in onze context zijn dat regelaars. Deze methode maakt het mogelijk om automatisch de structuur van een oplossing te ontdekken, in plaats van afhankelijk te zijn van een adequate sjabloonoplossing, die is aangedragen door de gebruiker. Daarnaast maakt GP het mogelijk om compacte gesloten-vorm regelaars te synthetiseren, waardoor het gebruik van opzoektabellen of online optimalisatie omzeild wordt. Hoewel GP gebruikt kan worden om de structuur van de regelaar te ontdekken vanaf nul, wordt door het gebruik van expertise de convergentie naar een oplossing verbeterd. Om een dergelijke voorziening van expertise mogelijk te maken, gebruiken we grammatica-gestuurd genetisch programmeren, een variant die kandidaatoplossingen beperkt om zich aan een door de gebruiker definieerde grammatica te houden. Desondanks blijft het gebruik van expertise optioneel; een gebruiker kan een erg generieke grammatica gebruiken, of diens expertise gebruiken om de zoekrichting te

beïnvloeden.

In combinatie met GP, gebruiken we het concept van tegenvoorbeeld-gestuurde inductieve synthese (TGIS) om kandidaatoplossingen te verfijnen op basis van tegenvoorbeelden, totdat de regelaar gegarandeerd voldoet aan de gewenste specificatie. In dit proefschrift stellen we twee TGIS-gebaseerde methoden voor, die verschillen in het gebruikte verificatieparadigma. De eerste aanpak gebruikt een *indirecte* methode, namelijk certificaatfuncties, terwijl de tweede aanpak gebruik maakt van een *directe* methode, namelijk bereikbaarheidsanalyse.

De eerste aanpak die wordt voorgesteld in dit proefschrift, co-synthetiseert zowel regelaars als certificaatfuncties, waarbij het laatstgenoemde gebruikt wordt om (indirect) de gewenste systeemspecificatie te verifiëren. We introduceren een nieuwe Lyapunov-barrièrefunctie (LBF) die, indien deze bestaat, een bereik-vermijd eigenschap impliceert. De LBF is zo gedefinieerd dat diens condities verifieerbaar zijn door middel van een *satisfiability modulo theories* (SMT) solver; een hulpmiddel dat in staat is om te bepalen of aan een eerste-orde logicaformule wordt voldaan. We gebruiken genetisch programmeren om een paar, bestaande uit een kandidaat regelaar en een kandidaat LBF, te synthetiseren op basis van een eindig aantal staat-ruimtemonsters. Gesynthetiseerde paren die aan de LBF-condities voldoen over deze eindige set van staten worden vervolgens formeel geverifieerd aan de hand van een SMT solver. Indien niet aan de LBF-condities wordt voldaan, wordt er een tegenvoorbeeld geëxtraheerd, dat wordt gebruikt voor het verfijnen van het syntheseproces. Deze eerste methodologie passen we toe op algemene hybride systemen die zijn gemodelleerd als *jump-flow* systemen, onderhevig aan bereik-vermijd specificaties. Additioneel stellen we een gespecialiseerde aanpak voor gladde niet-lineaire systemen met bemonsterde-data regelaars voor, gebaseerd op *controle* Lyapunov-barrièrefuncties. Tot slot demonstreren we hoe de voorgestelde aanpak gebruikt kan worden voor de verificatie van (bijna) optimale reglaars, die zijn ontworpen aan de hand van *reinforcement learning*.

De tweede aanpak is afhankelijk van bereikbaarheidsanalyse van het systeem. Hiertoe gebruiken we recente ontwikkelingen op het gebied van modelcontrole voor signaal temporele logica (STL) en de generatie van tegenvoorbeelden op basis van bereikbaarheidsanalyse. STL redeneert over enkele banen, terwijl bereikbaarheidsanalyse resulteert in bereikbare sets. Om de kloof te overbruggen tussen enkele banen en bereikbare sets, gebruiken we een geldige transformatie van STL naar *reachset* temporele logica (RTL), die direct over bereikbare sets beredeneert. Om de mate waarin een RTL-formule voldaan wordt te kwantificeren, introduceren we de kwantitatieve semantiek van RTL, die onze synthese van een optimalisatiecriterium voorziet. We gebruiken genetisch programmeren om regelaars te optimaliseren, op basis van een eindige set aan gesimuleerde banen. Regelaars die voor deze banen aan de specificatie voldoen, worden vervolgens geverifieerd met betrekking tot de gehele initiële set aan de hand van bereikbaarheidsanalyse. Indien aan de specificatie niet wordt voldaan op basis van de bereikbaarheidsanalyse, wordt een bijbehorende initiële conditie geëxtraheerd die resulteerde in de overtreding. Dit tegenvoorbeeld wordt vervolgens gebruikt om de regelaarsynthese te verfijnen. Deze tweede methodologie wordt toegepast voor niet-lineaire systemen met bemonsterde-data implementatie van de regelaar, onderhevig aan generieke STL-specificaties.

We demonsteren de effectiviteit van beide aanpakken op meerdere (academische) ca-

sestudies. De voorgestelde methoden hebben verschillende gebruiksgevallen waarvoor ze beter geschikt zijn; de certificaat-gebaseerde methode is het best geschikt voor laag-dimensionale systemen met grote initiële sets, terwijl de bereikbaarheid-gebaseerde methode het beste geschikt is voor hoger-dimensionale systemen met kleine initiële systemen, onderhevig aan ingewikkelde specificaties, uitgedrukt in temporele logica. Hoewel de twee voorgestelde methoden ofwel geschikt zijn voor algemene hybride systemen of algemene temporele logicaspecificaties, stellen we toekomstige uitbreidingen voor naar algemene systemen onderworpen aan algemene temporele logica. Beide aanpakken resulteren in correct-door-constructie regelaars in een compacte gesloten vorm, waarbij het gebruik van expertise optioneel is. De mogelijkheid van beide methoden om bemonsterde-data regelaars te synthetiseren maakt het mogelijk dat deze regelaars eenvoudig worden geïmplementeerd in embedded hardware met gelimiteerd geheugen en rekenkracht, wat dient als opstap naar snellere automatisering.

1

Introduction

1.1. Motivation

Since the dawn of mankind, humans have sought to simplify life, ranging from the invention of tools to the extremes of modern automation. As a result, human civilization evolved as our technology improved. At the time of writing, robots have been around for decades and a large percentage of the population has a 'super computer' in their pocket. However, our imagination regarding automation and robotics surpasses the current state of affairs. What is the limiting factor?

The word *technology* is derived from the Greek word "techne", which is often translated as 'craft' and 'art'. In automation, we try to replace this craft and art by a process with minimal human assistance. While modern technology provides the muscles and brain for automation in the form of mechatronics and the computation power, replacing the art of the human is still in itself an art. The design of algorithms, or *controllers*, that actuate a system in a desired way, is still a nontrivial challenge and has been extensively studied within the field of control engineering. In this dissertation, the research goal is to automate the controller design for a broad class of applications, namely *nonlinear/hybrid systems* with *temporal logic specifications*.

1.1.1. Hybrid systems

While our focus will be on nonlinear/hybrid systems, historically the primary focus of classical control has been on linear systems. As a result, linear control theory has matured, resulting in a range of constructive methodologies for controller design, such as the root locus and LQR control [48, 89], and has a wide range of methods dealing with disturbances and model uncertainties, such as H-infinity synthesis, μ-synthesis, and the use of linear matrix inequalities [22, 146]. Unfortunately, nonlinear control theory does not share the same level of maturity as linear control and, with the exception of specific subclasses of systems, lacks the same level of constructive control design methods [83, 144]. Regardless, smooth nonlinear systems is an important class of systems, with applications in e.g. robotics, automotive and aerospace engineering. For a wide class of modern cyber-

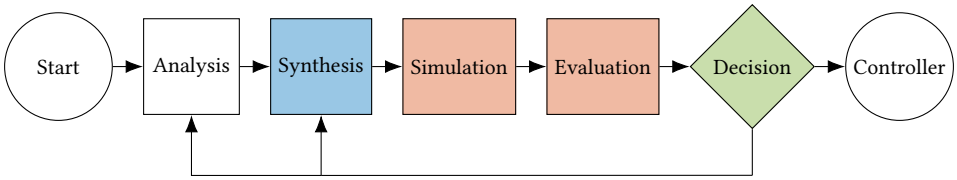


Figure 1.1: Basic cycle of the design process.

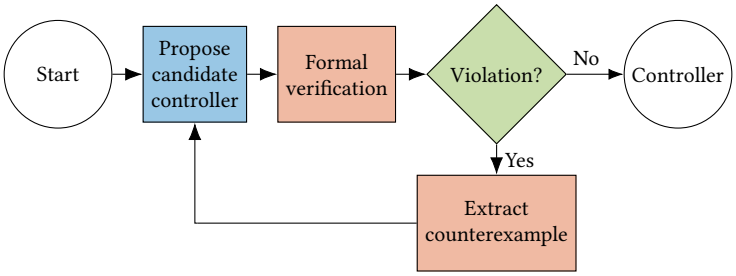
physical systems, smooth nonlinear systems are not sufficient to capture the dynamics: these systems exhibit both *continuous* and *discrete* behavior, and are referred to as hybrid systems [57, 156]. Sources of such behavior include electronic switches, mechanical phenomena, such as impacts and hysteresis, but also the digital implementation of controllers are sources of hybrid behavior, such as sampled data and quantization [57, 156]. Formal synthesis for general hybrid systems lacks constructive controller design methods, making it an intricate process heavily reliant on expert knowledge. To push the automation further, design methods for this complex class of systems is critical.

1.1.2. Temporal specifications

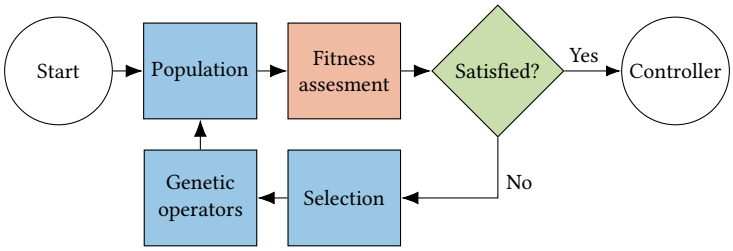
Where the system model describes the system we have at hand, the control specification describes the desired performance. The field of control theory has classically focused on 'complex systems with relatively simple performance criteria', i.e. dynamical systems modelled by differential equations and stability requirements. On the other hand, the field of computer science focuses on 'relatively simple systems with complex specifications', i.e. discrete systems with finite states and intricate system requirements. To formalize correctness of the behavior of computer systems, these complex system requirements have been formulated in temporal logics [14], i.e. logic formulae qualified over time. For example, temporal specifications include statements as '*eventually* system trajectories reach set A', 'the system trajectories are *always* in set B', and 'the system trajectories visit regions C and D *infinitely often*'. In recent years, with the rise of cyber-physical systems, these temporal logic specifications have been introduced to the field of control [16]. Of particular interest are temporal logic variants such as signal temporal logic (STL), which directly reason over continuous-time signals [100]. These temporal logics provide a formal framework for specifying the control specifications.

1.1.3. The design process

Given the complex task of controller synthesis for nonlinear/hybrid systems with temporal logic specifications, the main objective of this work is to shift the controller design by human engineers to an automated process. To this end, let us examine the human design process as a source of inspiration. Let us consider the 'basic cycle of design' model [127], shown in Figure 1.1. This cycle has five phases: analysis, synthesis, simulation, evaluation and decision. In the analysis phase, the desired functionality of the design is evaluated and translated into formal specifications. In the synthesis phase, a tentative design is constructed. This design is believed to be a good solution, but still requires testing. Given the tentative design, in the simulation phase, the behavior of the product is emulated and



(a) CEGIS.



(b) EA.

Figure 1.2: The cycles of CEGIS and EA for controller design. Both cycles follow similar steps as the basic cycle of a design process in Figure 1.1.

its properties derived by means of reasoning and/or models. This model is compared to the desired specifications in the evaluation phase. Based on this comparison, a decision is made on whether the design is satisfactory or not. Typically, the first design can, or needs, to be improved. In this case, the cycle returns to the synthesis step to improve upon the previous design. It is also possible that the previous design criteria or specifications did not properly reflect the intended functionality, and the analysis needs to be refined.

Within the context of control, the analysis phase is the formulation of the controller specifications, e.g. in the form of step-response characteristics or temporal logic specifications. In the synthesis phase the engineer could, e.g., select a set of PID values, formulate a cost function for e.g. LQR control or optimal control, or propose a structure of a nonlinear controller. In the simulation and verification phase the closed-loop behavior is tested or evaluated based on, e.g., simulating system trajectories or verifying stability using a Lyapunov function. Based on these results, the controller can be refined or deemed satisfactory.

In this work we use two methodologies closely resembling the human design cycle, namely Counterexample-guided Inductive Synthesis (CEGIS) [147] and Evolutionary Algorithms (EAs) [43]. Counterexample-guided inductive synthesis is an iterative design method, in which candidate solutions are synthesized based on iteratively added counterexamples. That is, a candidate solution is proposed and subsequently verified. If the desired specification is not met, a counterexample is extracted, which is used to refine the candidate solution. This design cycle is illustrated in Figure 1.2a.

Evolutionary algorithms is a class of optimization methods inspired by the concept of

evolution. Algorithms within this class include Genetic Algorithm (GA) [69], Evolution Strategy (ES) [140], and Genetic Programming (GP) [85]. These methods are population-based algorithms following a similar cycle, shown in Figure 1.2b. Within evolutionary algorithms, a randomly initialized *population* is 'evolved' based on a *fitness function*; a cost function capturing the desired objective. Based on the fitness values of candidate solutions, candidates are *selected*, altered, and/or recombined by means of *genetic operators* to form new candidates, which form a new population. This cycle, or generation, is repeated until a pre-defined stopping criterion is met. The underlying hypothesis is that over a number of these cycles the average fitness increases.

Assuming proper controller specifications are known beforehand, and therefore omitting the analysis phase from the basic design cycle in Figure 1.1, the basic design cycle closely resembles the cycles of CEGIS and EA, illustrated in Figure 1.2. That is, first candidate solutions are *synthesized* based on counterexamples or genetic operators. Subsequently the solutions are *simulated* and *evaluated* through verification or fitness assessment. If none of the candidate solutions satisfy a pre-determined stopping criterion (i.e. the *decision*), solutions are refined and re-designed in a new *synthesis* step. These two methods differ from each other in that within CEGIS the objective is typically qualitative i.e. true/false, whereas within EA the objective is quantitative, i.e. optimization of a cost function.

In this work we combine both of these methods to synthesize correct-by-construction controllers. Whereas evolutionary algorithms such as GA and ES evolve parameters, GP sets itself apart by its capability to evolve entire programs. In this work we use variants of GP and ES to propose candidate controller structures and their parameters, based on a finite set of training data. Subsequently, these candidate controllers are verified by means of a formal verification method. If the desired control specification is not met, a counterexample is extracted for which this specification is violated. This counterexample is then added to the training data and used to refine the candidate controllers. This combination of CEGIS and EA provides a framework resembling the human design process, with the goal to fully automate the controller synthesis for hybrid systems with temporal logic specifications.

1.2. Related work

In recent years, tools have been developed for automatic formal control synthesis for hybrid systems with temporal logic specifications or subproblems thereof. Most of these methods fit into one of three main paradigms: synthesis by means of 1) finite (bi-) simulation abstractions [17, 150], 2) control Lyapunov and/or barrier functions [12, 163] and 3) online optimization-based methods [16].

1.2.1. Abstraction-based methods

The first paradigm finds its roots in the field of computer science, in which temporal logic has been used to describe the correctness of complex system behaviors of intricate computer systems [14]. As it originally dealt with finite systems, (bi-)simulation approaches have been proposed to abstract infinite systems to finite systems [17, 150]. However, as a downside, these approaches (e.g. [55, 63, 99, 124]) suffer from the curse of dimensional-

ity and return controllers in the form of enormous look-up tables [170]. This makes this method especially troublesome for implementation in embedded hardware with limited memory capabilities. To partly overcome the limitations due to the curse of dimensionality, recent work has focused on improving scalability, e.g. by the decomposition into subsystems [79, 101, 105, 113, 153], using multiscale discretization [26, 54, 71], or using discretization-free approaches [169]. Tools implementing these abstraction-based methods include PESSOA [102], SCOTS [131], CoSyMa [107] and ROCS [94]. In this thesis the goal is to avoid state-space discretization and synthesize controllers in the form of compact expressions.

1.2.2. Certificate-based approaches

The second paradigm finds its root in more traditional control theory, as it utilizes certificate functions, i.e., functions that by their existence imply certain system behavior of autonomous systems. Examples include the well-known Lyapunov function [77] and barrier certificates [115], which are used to prove stability and invariance (safety), respectively. Similarly, *control* certificate functions are design tools for non-autonomous systems to modify the system behavior such that the closed-loop system satisfies the desired system properties. Examples of these control certificates are control Lyapunov function (CLF) [12, 35, 135, 136] and control barrier function (CBF) [163], which are design tools for stabilization and safety specifications, respectively. Using these (control) certificate functions or combinations thereof, (a subset of) temporal properties can be inferred indirectly [10, 52, 96, 126, 148, 167]. For general hybrid systems, [64, 65] recently proposed a set of sufficient conditions for certificate functions for temporal logic operators, such as *always*, *eventually* and *until*. To go beyond single temporal operators, the temporal logic formula can be decomposed into a sequence of sub formulae, resulting in a sequence of certificate functions that impose the full specification [19, 20, 36, 65, 165]. However, synthesizing these functions for general hybrid systems is nontrivial.

In the remainder, our focus lies on the synthesis of certificate functions in the context of control design, rather than the analysis of autonomous systems; a review of computational methods for Lyapunov function can be found in [53]. In recent years, we observe two main trends for certificate-based controller synthesis: methods relying on sum-of-squares (SOS) programming and semi-definite programming relaxations, and CEGIS approaches. The SOS approaches, see e.g. [111, 115, 116, 152], require polynomial systems and/or solutions. However, even if a polynomial closed-loop system is asymptotically stable, this does not imply that there exists a polynomial Lyapunov function, as shown in [3]. Additionally, these methods typically require a template solution and can suffer from numerical sensitivity issues.

CEGIS approaches, including [4, 74, 118, 120, 121], synthesize controllers and/or certificate functions by iteratively proposing and verifying candidate solutions, typically by means of a Satisfiability Modulo Theories (SMT) solver; a numerically sound tool capable of verifying whether a first-order logic formula is satisfied or not. The synthesis of these certificates typically does not restrict the class of systems and solutions to polynomials, but does require the user to provide a template solution. In recent work, neural networks have been used within a CEGIS framework for verification and/or formal controller synthesis [1, 27]. Neural networks are known to serve as universal function approximators,

hence they provide a good candidate as Lyapunov function representation.

Other certificate-based approaches include e.g. [59, 145], which rely on the use of templates of solutions in order to find Lyapunov functions, based on relaxations and the subsequent solving of semi-algebraic systems. Several approaches have also combined Lyapunov-like functions in conjunction with artificial intelligence techniques [166], such as Reinforcement learning [18, 93] and neural networks [34]. In these instances, a certificate function is typically used to restrict policies to a safe set of inputs, or are used a posteriori to verify the system behavior.

Overall, these existing methods are limited to a subclass of systems, such as polynomial systems, and/or require expert knowledge in the form of an adequate template solution. In this thesis the goal is to address general systems and minimize the need of expert knowledge.

1.2.3. Optimization-based methods

In the final paradigm, optimization-based methods are employed, typically to optimize a cost function related to the temporal logic specification [16]. In these approaches, the quantitative semantics of STL [40, 45] are utilized, which provides a quantitative score on how robustly the formula is satisfied. The quantitative semantics provides a clear optimization criterion, enabling optimization-based methods for temporal logic, such as model predictive control (MPC) [46, 95, 117, 133, 134], optimal trajectory planning [110], and reinforcement learning [5, 92]. Typically, these approaches only deal with a single initial condition, with the notable exceptions of [46, 117]. In [134], tube MPC is used, in which a tube around a nominal initial condition is found for which the robustness score is guaranteed. Other optimization-based approaches such as [112] are similar to the abstraction methods in that they are automaton-based, but do not require abstraction.

As an alternative, to be able to consider multiple initial conditions, optimization approaches using reachability analysis [6] have been proposed [37, 138, 139, 141]. Reachability analysis constructs an over-approximation of all the states which can be reached, starting in a given initial set. By relying on reachability analysis, controllers are optimized w.r.t. a reachable set, rather than a single trajectory. In [139], MPC is combined with reachability analysis, whereas in [37, 138] reach-avoid problems for nonlinear systems are tackled by synthesizing a sequence of optimal control inputs [37] or linear controllers [138] for a sequence of time intervals. The approach in [138] is extended to piecewise affine systems in [141].

The downside of these optimization methods is that they are only applicable to a single initial condition, they require online optimization, e.g. when using MPC-based approaches, or, in case the optimal solution is computed offline, the controller is typically stored as a look-up table. Our goal is to synthesize a controller that has a relative low computation cost and can be stored efficiently.

1.2.4. Other approaches

While the three aforementioned paradigms cover a wide range of approaches, not all related work belongs to one of these approaches. In this section we cover some of these other alternatives.

In [2] a CEGIS approach is used for the synthesis of digital controllers for linear

continuous-time systems. The authors in [151] use constraint solving is to synthesize a switching law for a hybrid system under a safety specification, based on template solutions. In [97, 98] the synthesis for a fragment of STL is reformulated as a prescribed performance control (PPC) problem, resulting in a continuous state-feedback control law.

Finally, with the enormous surge in interest in artificial intelligence, methodologies exploring the combination of artificial intelligence and formal methods for controller synthesis have arisen, see e.g. the survey in [166]. Approaches of interest outside the aforementioned paradigms include [41], in which neural network controllers are synthesized for nonlinear system with STL specifications and verified by means of reachability analysis for neural networks. Similarly, in [168] a CEGIS-like approach is proposed, which uses reinforcement learning to design neural network controllers for nonlinear discrete-time systems under STL specifications. This controller is designed such that the satisfaction of the STL specification is maximized for an initial set. Rather than using a formal verification method based on reachability analysis, the method relies on the falsifier S-TaLiRo [11] to provide counterexamples, which are used to improve the worst-case performance. Therefore, the property can not always be guaranteed without additional verification.

1.2.5. Evolutionary algorithms

Over the years, multiple evolutionary approaches have been applied to control problems, see e.g. the surveys [47, 91, 142]. Specifically, genetic programming has been proposed for the discovery of Lyapunov functions, see e.g. [62, 103], control Lyapunov functions [154], and controllers, see e.g. [28, 30, 38, 87, 90, 143] to name a few. The survey in [86] provides several anecdotal success stories of the application of genetic programming for control, in which the resulting controllers were 'competitive' with controllers designed by experts. However, in all the aforementioned work, the fitness is based on specific samples and/or simulations and no formal guarantees can be given on the behavior of the system, other than for the specific test cases. Only a limited number of publications consider some form of verification, including [122, 123], in which quadratic Lyapunov functions are used based on the linearized system, and [28], in which the stability of the closed-loop system is guaranteed by using the Kharitonov Theorem, but is therefore only applicable to linear systems.

On the other hand, in the field of computer science, methods have been proposed which combine genetic programming with formal verification methods, such as model checking [68, 73, 75, 76] or SMT solvers [21]. These methods are used to synthesize provably correct programs, which satisfy temporal logic specifications. However, these frameworks have not been applied to the field of control, i.e. the synthesis of controllers for dynamical systems. In this thesis the goal is to use genetic programming with formal verification methods to synthesize correct-by-construction controllers.

1.3. Research goal and contributions

A brief overview of general shortcomings of the methods in Section 1.2 is shown in Table 1.1. Overall, existing work suffers from one or more of the following shortcomings:

1. Resulting controllers are in an impractical form for the implementation in embedded hardware with limited memory, e.g. a sizable look-up table.

Table 1.1: Comparison of the synthesis approaches in literature.

Method	Advantages	Limitations
Abstraction-based	• Complete method	• Curse of dimensionality • Look-up table controller
Certificate-based	• Compact representation	• Requires expert knowledge
Optimization-based	• Optimal control	• Online computation cost *or* Look-up table controller
AI/GP-based	• Little expert knowledge required	• Challenging to provide guarantees

2. Resulting controllers suffer from a considerable online computation cost, such as MPC-based approaches, and are therefore impractical for embedded implementations.
3. The method is only applicable to subclasses of systems, e.g. linear, polynomial, or smooth nonlinear systems.
4. The methodology is only capable of handling a limited class of specifications, such as reach-avoid problems.
5. The methodology relies on expert knowledge, e.g. by requiring solution templates.

The goal of this dissertation is to develop a framework capable of addressing all of these issues. To overcome the first two issues, our approach is to synthesize controllers in the form of compact closed-form expressions, such that we obtain a controller which is to implement and does not suffer the same computational burden compared to methods relying on online optimization. Specifically, the goal is summarized as follows:

Research goal

Develop a framework for automated correct-by-construction synthesis of closed-form controllers for nonlinear/hybrid systems under temporal specifications, where the use of expert knowledge is optional.

In order to synthesize controllers in the form of closed-form expressions without the reliance on expert knowledge, we use genetic programming. Genetic programming sets itself apart by evolving entire programs, rather than optimizing parameters within a predefined structure. In our case, the evolved program is a controller that is built up and modified (evolved) based on elementary building blocks. These building blocks consist of e.g. the state variables and basic functions such as addition and multiplication. Specifically, we propose the use of Grammar-Guided Genetic Programming (GGGP) [104, 162], which allows users to provide a grammar. This enables the user to take a hands-off approach by considering general grammars, ranging from general polynomial expressions to grammars including transcendental functions, or to use expert knowledge and bias the search space to structures for which there is likely to exist a solution. In this sense, there is no requirement of expert knowledge, but using it could yield a speed-up in the synthesis.

In order to design controllers that guarantee temporal logic specifications, we combine genetic programming with formal verification within a Counterexample-guided Inductive Synthesis (CEGIS) framework. That is, candidate controllers are proposed using GP and are verified using formal verification. If the proposed controllers violate the specification, the verification method returns a counterexample, which is used to improve the proposed controllers. In this work we propose the use of two different verification paradigms, resulting in two different synthesis frameworks, namely an indirect and direct method. The first method builds upon one of the corner stones of nonlinear control: Lyapunov functions, or in a broader sense, certificate functions. This results in the following framework:

Indirect method: certificate-based synthesis

Genetic programming-driven co-synthesis of controllers and candidate certificate functions, evaluated based on the conditions of the certificate function.

The second approach relies on the direct evaluation of the closed-loop behavior of the system and controller. This method builds upon one of the fundamental tools of controller evaluation: simulating the behavior of a system. Instead of relying on simulation of singular trajectories, we over-approximate the set of all possible trajectories by means of reachability analysis. This yields the following framework:

Direct method: reachability-based synthesis

Genetic programming-driven controller synthesis, evaluated based on an over-approximation of all the system trajectories.

In general, our main contribution is the proposal of these two approaches, in which we combine existing techniques, including GP, SMT solvers, certificate functions, reachability analysis and CEGIS, to solve the nontrivial problems stated in the research goal, i.e., the automatic synthesis of closed-form controllers for nonlinear/hybrid systems under temporal logic specifications, where the controller structure can be discovered automatically. To enable the combination of these techniques into a single framework, additional contributions are made within this dissertation:

- We introduce (specialized) Lyapunov-like functions for sampled-data and hybrid systems for specific reach-avoid problems, which are automatically verifiable by means of state-of-the-art SMT solvers (Sections 4.3, 4.4, and 4.7.2).
- We define the quantitative semantics of reachset temporal logic (Section 5.3).

Finally, the proposed framework enables our final contribution:

- We use our certificate-based framework to automatically verify near-optimal control policies by means of formal synthesis of Lyapunov-like functions (Section 4.8).

1.4. Outline

In this section we provide the outline of this dissertation. Figure 1.3 illustrates the class of systems and specifications tackled per chapter.

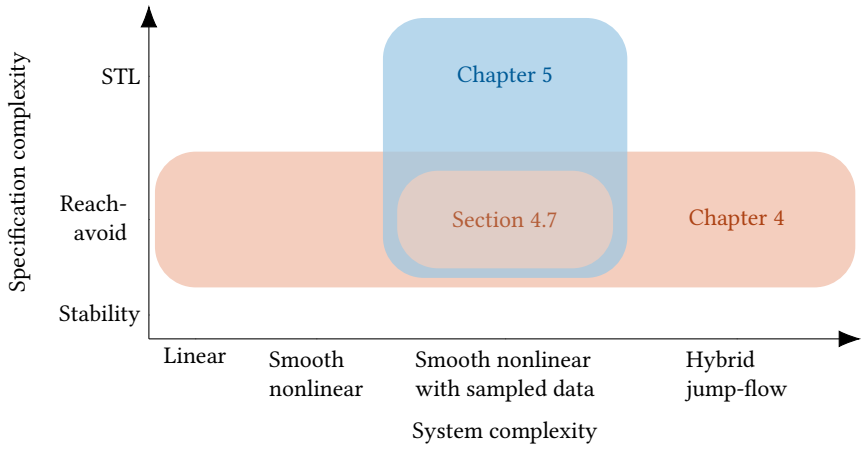


Figure 1.3: Classes of systems and specifications in increasing complexity, and the corresponding chapters that address the corresponding problems.

- **Chapter 2: Preliminaries.** In this chapter we recap the fundamentals underlying this thesis. After introducing the general notation, we introduce the mathematical framework of the class of hybrid systems considered in this work, namely jump-flow systems. Subsequently, an introduction to temporal logic is provided, specifically signal temporal logic and reachset temporal logic, which enables model checking for STL by means of reachability analysis. Finally, we cover satisfiability modulo theories solvers, which form the backbone of verifying the certificate functions in Chapter 4 and satisfaction of STL formulae in Chapter 5.
- **Chapter 3: Grammar-guided genetic programming.** In this chapter we detail our evolutionary algorithm driving the heuristic search for controllers in Chapters 4 and 5, namely a variant of grammar-guided genetic programming.
- **Chapter 4: Certificate-based synthesis.** In this chapter we propose an indirect formal controller synthesis approach, i.e. a framework in which the formal verification relies on the co-synthesized (control) certificate function. We consider hybrid systems modelled as jump-flow systems with reach-avoid problems. Additionally, we develop a specialized approach for nonlinear systems with a sampled-data implementation of the controller. Finally, based on the developed framework, we design certificate functions to verify (near) optimal controllers obtained by means of reinforcement learning.
- **Chapter 5: Reachability-based synthesis.** In this chapter we propose a direct formal controller synthesis approach, i.e. a framework in which the formal verification relies on reachability analysis. We consider smooth nonlinear open-loop systems with a sampled-data implementation of the controller, designed for general specifications expressed as STL formulae.
- **Chapter 6: Discussion.** In this chapter we compare the indirect and direct method

in Chapters 4 and 5. Besides the fact that these two method address a different combination of system class and type of specification (see also Figure 1.3), both methods have their own weaknesses and advantages, resulting in different use cases, which are addressed in this chapter. Additionally, extensions of both methods to a larger class of systems or specifications is also addressed in this chapter.

- **Chapter 7: Conclusions and recommendations.** In this final chapter, we summarize the main contributions and provide an outlook of promising future work.

2

Preliminaries

In this chapter, we introduce notation and preliminary notions regarding the class of systems and classes of specifications relevant to this dissertation, namely hybrid systems in the form of jump-flow systems and temporal logics, in particular signal temporal logic and reachset temporal logic. The chapter is concluded with the preliminaries on satisfiability modulo theories solvers, which play an important role in the verification used within this work.

2.1. Notation

The sets of natural, integer, rational and real numbers are denoted by $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$, respectively, where $\mathbb{N} = \{0, 1, 2 \dots\}$. A non-negative subset is denoted using the subscript $\cdot_{\geq 0}$, e.g., $\mathbb{R}_{\geq 0} = \{x \in \mathbb{R} \mid x \geq 0\}$. Given a set $D \subseteq \mathbb{R}^n$, we denote the boundary, the interior and its power set with ∂D, $int(D)$, and 2^D, respectively. A vector in $\mathbb{R}^n$ comprising of only zeros or ones in denoted as $\mathbf{0}_n$ and $\mathbf{1}_n$ respectively. The image of set A under f is denoted by $f[A]$. Finally, the Euclidean norm is denoted by $\|\cdot\|$.

2.2. Hybrid systems

Throughout this work, hybrid systems are modeled as jump-flow systems, following the framework from [57]. We briefly recall the following definitions:

Definition 2.2.1 (Hybrid time domains [57, Def. 2.3]). *A subset $E \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ is a compact hybrid time domain if $E = \bigcup_{j=0}^{J-1}([t_j, t_{j+1}], j)$ for some finite sequence of times $0 = t_0 \leq t_1 \dots \leq t_J$. It is a hybrid time domain if for all $(T, J) \in E$, $E \cap ([0, T] \times \{0, 1, \dots J\})$ is a compact hybrid time domain.*

Given a hybrid time domain E and a given $j \in \mathbb{N}$, we denote a time interval $T^j := \{t \mid (t, j) \in E\}$.

Definition 2.2.2 (Hybrid arc [57, Def. 2.4]). *A function $\phi : E \to \mathbb{R}^n$ is a hybrid arc if E is a hybrid time domain and if for each $j \in \mathbb{N}$ the function $t \mapsto \phi(t, j)$ is locally absolutely continuous on the interval T^j.*

Now a hybrid jump-flow system is defined as follows:

Definition 2.2.3 (Hybrid system [57, §2.1]). *A hybrid system $\mathcal{H}$ is defined as a tuple (C, F, D, G), where set $C \subset \mathbb{R}^n$ is the flow set, set-valued function $F : C \rightrightarrows \mathbb{R}^n$ the flow map, set $D \subset \mathbb{R}^n$ the jump set, and set-valued function $G : D \rightrightarrows \mathbb{R}^n$ the jump map.*

Given a set-valued function $M : \mathbb{R}^m \rightrightarrows \mathbb{R}^n$, we denote its domain with $\mathrm{dom}M$, defined as $\mathrm{dom}M := \{x \in \mathbb{R}^m \mid M(x) \neq \emptyset\}$. In this work, we assume that the considered hybrid systems satisfy the following conditions, also referred to as the hybrid basic conditions in [57]:

Assumption 2.2.1 (Hybrid basic conditions [57, Ass. 6.5]).

1. *C and D are closed subsets of $\mathbb{R}^n$.*
2. *$F : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is outer semicontinuous and locally bounded relative to C, $C \subset \mathrm{dom}F$, and $F(x)$ is convex for every $x \in C$.*
3. *$G : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is outer semicontinuous and locally bounded relative to D, and $D \subset \mathrm{dom}G$.*

For the exact definition of outer semicontinuity and local boundedness for set-valued mappings we refer to Definition 5.9 and 5.14 in [57]. A special class of systems satisfying these conditions are those whose flow and jump maps are described by continuous functions $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^n$. If a hybrid system $\mathcal{H}$ does not satisfy the hybrid basic conditions, one could construct the Krasovskii regularization (see Definition 4.13 in [57]), which by definition satisfies these conditions. Under the hybrid basic conditions, solutions to the hybrid system are defined as follows:

Definition 2.2.4 (Solution to a hybrid system [57, §6.2.1]). *A hybrid arc $\phi : E \to \mathbb{R}^n$ is a solution to a hybrid system $\mathcal{H}$ if $\phi(0,0) \in C \cup D$ and*

- *$\forall j \in \mathbb{N}$ and almost all $t \in T^j$:*

$$\begin{aligned} \phi(t,j) &\in C, \\ \dot{\phi}(t,j) &\in F(\phi(t,j)). \end{aligned}$$

- *$\forall (t,j) \in \{(t,j) \in E \mid (t, j+1) \in E\}$:*

$$\begin{aligned} \phi(t,j) &\in D, \\ \phi(t,j+1) &\in G(\phi(t,j)). \end{aligned}$$

Figure 2.1 illustrates an example of the flow and jump sets C and D, and a solution $\phi(t,j)$. In this work, we distinguish the following classes of hybrid arcs ϕ:

Definition 2.2.5 (Types of hybrid arcs [57, Def. 2.5]). *A hybrid arc $\phi : E \to \mathbb{R}^n$ is called*

- complete *if its domain E is unbounded;*

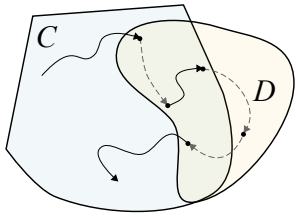


Figure 2.1: Example of a flow set C, jump set D and a solution $\phi(t, j)$.

- Zeno *if it is complete and* $\sup\{t \in \mathbb{R}_{\geq 0} \mid \exists j \in \mathbb{N} : (t, j) \in E\} < \infty$*, i.e. there is an infinite number of jumps within a finite time interval;*
- maximal *if there exists no solution* ψ *to* $\mathcal{H}$ *such that* $\mathrm{dom}\phi \subset \mathrm{dom}\psi$ *and* $\phi(t, j) = \psi(t, j)$ *for all* $(t, j) \in \mathrm{dom}\phi$*.*

Finally, we denote $\mathcal{S}_{\mathcal{H}}(I)$ as the set of all maximal solutions $\phi : E \to \mathbb{R}^n$ to $\mathcal{H}$ with $\phi(0, 0) \in I$:

$$\mathcal{S}_{\mathcal{H}}(I) = \{\phi \mid \phi(0, 0) \in I \text{ and } \phi(t, j) \text{ is a solution to } \mathcal{H}\}. \tag{2.1}$$

Throughout this dissertation, we consider two classes of nonlinear systems which can be considered as special cases of jump-flow systems. These two classes are **continuous** systems and continuous open-loop systems with **sampled-data** controllers. The subclass of continuous-time systems, as the name suggests, only has continuous data, i.e. it is described by hybrid systems $\mathcal{H} = (C, F, \emptyset, \emptyset)$. Since this subclass has no jump set and map, we define it with the reduced data as follows:

Definition 2.2.6 (Continuous-time system)**.** *A continuous-time system* Σ *is defined as a tuple* (C, F)*, where* $C \subset \mathbb{R}^n$ *is the flow set and the set-valued function* $F : C \rightrightarrows \mathbb{R}^n$ *is the flow map.*

Since solutions $\phi : E \to \mathbb{R}^n$ have a constant jump argument, i.e. $E = T^0 \times \{0\}$, we denote these solutions with $\xi : T^0 \to \mathbb{R}^n$, where $\xi(t) = \phi(t, 0)$.

Next, let us consider the subclass of continuous open-loop systems with a sampled-data implementation of the controller. Consider a continuous open-loop system described by the data $(\mathcal{X}, F_{\mathrm{ol}})$, where $\mathcal{X} \subseteq \mathbb{R}^n$ and $F_{\mathrm{ol}} : \mathbb{R}^n \times \mathbb{R}^m \rightrightarrows \mathbb{R}^n$, such that for almost all $t \in T^0$

$$\dot{\xi}(t) \in F_{\mathrm{ol}}(\xi(t), u(t)), \tag{2.2}$$
$$\xi(t) \in \mathcal{X}, \tag{2.3}$$

where $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ denotes the control input. Consider a sampled-data implementation of a state-feedback controller $\kappa : \mathcal{X} \to \mathcal{U}$, i.e., the control input is updated periodically after η seconds and is held constant in between updates. Using the jump-flow framework, this system can be modelled as $\mathcal{H} = (C, F, D, G)$ with

$$C = \mathcal{X}^2 \times [0, \eta], \qquad F(s) = (F_{\mathrm{ol}}(s_{\mathrm{x}}, \kappa(s_{\mathrm{q}})), \mathbf{0}_n, 1),$$
$$D = \mathcal{X}^2 \times \{\eta\}, \qquad G(s) = (s_{\mathrm{x}}, s_{\mathrm{x}}, 0),$$

where the state vector $s = (s_{\mathrm{x}}, s_{\mathrm{q}}, s_{\mathrm{t}}) \in \mathcal{X}^n \times [0, \eta]$ consists of the continuous states $s_{\mathrm{x}} \in \mathcal{X}$, the sampled states $s_{\mathrm{q}} \in \mathcal{X}$ and a timer state $s_{\mathrm{t}} \in [0, \eta]$. Similar to the partition of the state vector, let us partition solutions ϕ as $\phi(t, j) = (\xi(t, j), \xi_q(t, j), \tau(t, j))$, where ξ denotes the continuous states, $\xi_q(t, j) \in \mathbb{R}^n$ the sampled states, and $\tau(t, j) \in [0, \eta]$ the timer state. Since the jump instances and the state evolution of the sampled and timer states are easily derived from ξ and t[1], we typically omit these in our analysis of this class of systems. That is, the class of continuous open-loop systems with sampled-data controller, with solutions $\xi : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, is defined as follows:

Definition 2.2.7 (Continuous open-loop system with sampled-data controller)**.**
A continuous open-loop system with sampled-data controller Σ^{sd} is defined as a tuple $(\mathcal{X}, F_{\mathrm{ol}}, \kappa, \eta)$, where set $\mathcal{X} \subset \mathbb{R}^n$ is the state space, set-valued function $F_{\mathrm{ol}} : \mathcal{X} \times \mathcal{U} \rightrightarrows \mathbb{R}^n$ the open-loop flow map, the function $\kappa : \mathcal{X} \to \mathcal{U}$ the controller, and scalar $\eta > 0$ the sampling time.

2.3. Temporal logic

Stemming from the field of computer science, temporal logic (TL) has been used to describe the correctness of the behavior of complex software and hardware systems [14]. In recent years, temporal logic has been applied to the field of control [16]. In this work we are particularly interested in signal temporal logic (STL), which reasons over continuous-time signals [100]. STL is defined through the following recursive grammar:

$$\varphi := \text{true} \mid h(s) \geq 0 \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_{[a,b]} \varphi_2, \tag{2.4}$$

where $\varphi, \varphi_1, \varphi_2$ are STL formulae, and $h(s) \geq 0$ is a predicate over a signal $s : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and a function $h : \mathbb{R}^n \to \mathbb{R}$. The Boolean operators $\neg$ and $\wedge$ denote negation and conjunction, respectively, and $\mathcal{U}_{[a,b]}$ denotes the *until* operator. Using the given STL grammar, one can also define other standard (temporal) operators, such as

- disjunction: $\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$,
- next: $\bigcirc_a \varphi := \text{true}\, \mathcal{U}_{[a,a]} \varphi$,
- eventually: $\Diamond_{[a,b]} \varphi := \text{true}\, \mathcal{U}_{[a,b]} \varphi$,
- always: $\Box_{[a,b]} \varphi := \neg\Diamond_{[a,b]} \neg\varphi$.

Given a set $Y \subset \mathbb{R}^n$ which can be expressed as

$$Y := \left\{ x \in \mathbb{R}^n \;\middle|\; \bigvee_i \bigwedge_j h_{ij}(x) \sim 0 \right\}, \quad \sim \in \{\geq, >\},$$

we denote the logic function indicating set membership by $\varphi_Y = \bigvee_i \bigwedge_j h_{ij}(x) \sim 0$. The satisfaction relation $(s, t) \models \varphi$ indicates that the signal s starting at t satisfies φ. The STL

[1] That is, $j = \left\lfloor \frac{t}{\eta} \right\rfloor$, $\xi_{\mathrm{q}}(t, j) = \xi(j\eta, j)$ and $\tau(t, j) = t - j\eta$.

semantics is defined recursively as:

$$\begin{aligned}
(s,t) \models & h(s) \geq 0 && \iff h(s(t)) \geq 0,\\
(s,t) \models & \neg\varphi && \iff (s,t) \not\models \varphi,\\
(s,t) \models & \varphi_1 \wedge \varphi_2 && \iff (s,t) \models \varphi_1 \text{ and } (s,t) \models \varphi_2,\\
(s,t) \models & \varphi_1 \mathcal{U}_{[a,b]} \varphi_2 && \iff \exists t' \in [t+a, t+b],\ (s,t') \models \varphi_2\\
& && \quad \text{and } \forall t'' \in [t,t'),\ (s,t'') \models \varphi_1.
\end{aligned}$$

Note that our semantics of the until operator $\mathcal{U}$ conforms to the definition in [125], which deviates from the definition in e.g. [100]. In the latter, φ_1 and φ_2 have to hold simultaneously at $t = t'$. Our choice for our adapted definition is motivated by our use of the results in [125] (see Section 2.3.3). Using these semantics, we can also derive the semantics for the derived operators:

$$\begin{aligned}
(s,t) \models & \varphi_1 \vee \varphi_2 && \iff (s,t) \models \varphi_1 \text{ or } (s,t) \models \varphi_2,\\
(s,t) \models & \bigcirc_a \varphi && \iff (s,t+a) \models \varphi,\\
(s,t) \models & \Diamond_{[a,b]} \varphi && \iff \exists t' \in [t+a, t+b],\ (s,t') \models \varphi,\\
(s,t) \models & \Box_{[a,b]} \varphi && \iff \forall t' \in [t+a, t+b],\ (s,t') \models \varphi.
\end{aligned}$$

2.3.1. QUANTITATIVE SEMANTICS

STL is equipped with a *quantitative semantics* that provides a robustness score of how well a signal s starting at time t satisfies or violates the STL specification [40, 45]. The quantitative semantics are given by a function $\rho(s, \varphi, t)$ recursively defined as:

$$\begin{aligned}
\rho(s, \text{true}, t) &= +\infty,\\
\rho(s, h(s) \geq 0, t) &= h(s(t)),\\
\rho(s, \neg\varphi, t) &= -\rho(s, \varphi, t),\\
\rho(s, \varphi_1 \wedge \varphi_2, t) &= \min(\rho(s, \varphi_1, t), \rho(s, \varphi_2, t)),\\
\rho(s, \varphi_1 \mathcal{U}_{[a,b]} \varphi_2, t) &= \max_{t' \in [t+a, t+b]} \left(\min \left(\rho(s, \varphi_2, t'), \min_{t'' \in [t, t')} \rho(s, \varphi_1, t'') \right) \right).
\end{aligned}$$

For the derived operators we get:

$$\begin{aligned}
\rho(s, \varphi_1 \vee \varphi_2, t) &= \max(\rho(s, \varphi_1, t), \rho(s, \varphi_2, t)),\\
\rho(s, \bigcirc_a \varphi, t) &= \rho(s, \varphi, t+a),\\
\rho(s, \Diamond_{[a,b]} \varphi, t) &= \max_{t' \in [t+a, t+b]} \rho(s, \varphi, t'),\\
\rho(s, \Box_{[a,b]} \varphi, t) &= \min_{t' \in [t+a, t+b]} \rho(s, \varphi, t').
\end{aligned}$$

The quantitative semantics is sound and complete [39, 45], that is:

$$\begin{aligned}
\rho(s,\varphi,t) > 0 \Rightarrow (s,t) \models \varphi \text{and} (s,t) \models \varphi \Rightarrow \rho(s,\phi,t) \geq 0,\\
\rho(s,\varphi,t) < 0 \Rightarrow (s,t) \not\models \varphi \text{and} (s,t) \not\models \phi \Rightarrow \rho(s,\phi,t) \leq 0.
\end{aligned}$$

Additionally, if $\rho(s, \varphi, t)$ is negative, lower values imply that φ is more strongly violated. Conversely, if $\rho(s, \varphi, t)$ is positive, higher values imply that φ is satisfied more robustly.

2.3.2. Connection to jump-flow systems

In Section 2.2, hybrid systems were modelled as jump-flow systems, where solutions are defined w.r.t. hybrid time domains, whereas signal temporal logic is defined for signals over time. To bridge this gap, we could redefine STL for signals over the hybrid time domain. In this case, we redefine the until operator $\mathcal{U}_{\mathcal{I}}$, where $\mathcal{I} \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ denotes a hybrid time interval. The satisfaction relationship $(\phi, t, j) \models \varphi$, for $\phi : E \to \mathbb{R}^n$, indicates then that the hybrid arc at time t and jump j satisfies φ. Moreover, the semantics is redefined as

$$
\begin{aligned}
(\phi,t,j) \models & h(\phi) \geq 0 && \iff h(\phi(t,j)) \geq 0,\\
(\phi,t,j) \models & \neg\varphi && \iff (\phi,t,j) \not\models \varphi,\\
(\phi,t,j) \models & \varphi_1 \wedge \varphi_2 && \iff (\phi,t,j) \models \varphi_1 \text{ and } (\phi,t,j) \models \varphi_2,\\
(\phi,t,j) \models & \varphi_1 \mathcal{U}_{\mathcal{I}} \varphi_2 && \iff \exists (t',j') \in ((t,j)+\mathcal{I}) \cap E,\ (\phi,t',j') \models \varphi_2 \text{ and}\\
& && \quad \forall (t'',j'') \in ([t,t') \times [j,j')) \cap E,\ (\phi,t'',j'') \models \varphi_1.
\end{aligned}
$$

Alternatively, given a hybrid arc $\phi : E \to \mathbb{R}^n$, we can omit the explicit jump argument, resulting in a *set-valued* solution $\xi_\phi : \mathbb{R}_{\geq 0} \rightrightarrows \mathbb{R}^n$, satisfying

$$
\forall t' \in \{t \mid (t,j) \in E\} : \xi_\phi(t') = \{\phi(t',j) \mid (t',j) \in E\}. \tag{2.5}
$$

Redefining STL for set-valued functions, the grammar remains identical as in Section 2.3 and the semantics is redefined with

$$
(\xi_\phi, t) \models h(\xi_\phi) \geq 0 \iff \forall s \in \xi_\phi(t) : h(s) \geq 0, \tag{2.6}
$$

where the remainder of the semantics follows verbatim to the semantics in Section 2.3. In this work we are not interested in STL specifications w.r.t. the jump argument, and therefore we will adopt the latter definition of STL for jump-flow systems.

2.3.3. Reachset temporal logic

The STL formulae previously defined reason over a singular trajectory. However, in this dissertation, we are not only interested in the performance of the system for a singular initial condition, but rather for a pre-defined set of initial conditions. That is, given a system $\mathcal{H}$ and a set of initial conditions I, we are interested in the behavior of all trajectories in $\mathcal{S}_{\mathcal{H}}(I)$. In this thesis, we only consider STL specifications with respect to only continuous time, and therefore we consider trajectories in which we omit the explicit jump argument. Therefore, abusing notation, we use $\xi \in \mathcal{S}_{\mathcal{H}}(I)$ to denote $\xi \in \{\xi_\phi \mid \phi \in \mathcal{S}_{\mathcal{H}}(I)\}$, where $\xi, \xi_\phi : \mathbb{R}_{\geq 0} \rightrightarrows \mathbb{R}^n$ and ξ_ϕ is defined as in (2.5). Now let us define a reachable set:

Definition 2.3.1 (Reachable set). *Given a system $\mathcal{H}$ and initial set I, a mapping $R_e : \mathbb{R}_{\geq 0} \to 2^{\mathbb{R}^n}$ is an exact reachable set if and only if:*

$$
\forall t \in \mathbb{R}_{\geq 0} : \{x \mid \xi \in \mathcal{S}_{\mathcal{H}}(I) \text{ and } x \in \xi(t)\} = R_e(t). \tag{2.7}
$$

A mapping $R : \mathbb{R}_{\geq 0} \to 2^{\mathbb{R}^n}$ is a reachable set if and only if $\forall t \in \mathbb{R}_{\geq 0} : R_e(t) \subset R(t)$.

That is, a reachable set satisfies $\forall t \in \mathbb{R}_{\geq 0}, \forall \xi \in \mathcal{S}_{\mathcal{H}}(I) : \xi(t) \subseteq R(t)$. Note that typically the exact reachable set cannot be computed [6]. Tools for reachability analysis include Flow* [29], CORA [8], dReach [84] and SpaceEx [49]. In this work, we use the reachability analysis tool CORA, which returns a sequence of sets

$$\mathcal{R} = \mathcal{R}_{\{t_0\}}\mathcal{R}_{(t_0,t_1)}\mathcal{R}_{\{t_1\}}\mathcal{R}_{(t_1,t_2)}\dots\mathcal{R}_{\{t_m\}}, \tag{2.8}$$

that form a reachable set given by

$$R(t) = \begin{cases} \mathcal{R}_{\{t_i\}} & \text{if } t = t_i, \\ \mathcal{R}_{(t_i,t_{i+1})} & \text{if } t \in (t_i, t_{i+1}). \end{cases} \tag{2.9}$$

The STL semantics over singular trajectories does not directly translate to the evaluation over reachable sets. To be able to reason directly over a reachable set, [125] introduced reachset temporal logic (RTL). The RTL fragment relevant for this work is given by:

$$\begin{aligned} \psi &:= \text{true} \mid h(x) \geq 0 \mid \neg\psi \mid \psi_1 \wedge \psi_2, \\ \Psi &:= \mathcal{A}\psi \mid \Psi_1 \vee \Psi_2 \mid \Psi_1 \wedge \Psi_2 \mid \bigcirc_a \Psi, \end{aligned}$$

where ψ, ψ_1, ψ_2 are propositional formulae over a state $x \in \mathbb{R}^n$, $h : \mathbb{R}^n \to \mathbb{R}$, Ψ, Ψ_1, Ψ_2 are formulae over a reachable set $R : \mathbb{R}^n \to 2^{\mathbb{R}^n}$, and $\mathcal{A}$ denotes the *all* operator. The semantics is defined as follows:

$$\begin{aligned} x &\models h(x) \geq 0 && \iff h(x) \geq 0, \\ x &\models \neg\psi && \iff x \not\models \psi, \\ x &\models \psi_1 \wedge \psi_2 && \iff x \models \psi_1 \text{ and } x \models \psi_2, \\ (R,t) &\models \mathcal{A}\psi && \iff \forall x \in R(t) : x \models \psi, \\ (R,t) &\models \Psi_1 \vee \Psi_2 && \iff (R,t) \models \Psi_1 \text{ or } (R,t) \models \Psi_2, \\ (R,t) &\models \Psi_1 \wedge \Psi_2 && \iff (R,t) \models \Psi_1 \text{ and } (R,t) \models \Psi_2, \\ (R,t) &\models \bigcirc_a \Psi && \iff (R, t+a) \models \Psi. \end{aligned}$$

To enable a transformation from STL to RTL, we pose the following assumption on the STL formula:

Assumption 2.3.1. *The STL formula φ is c-divisible, i.e., all interval bounds of the temporal operators of φ are divisible by c.*

Given an STL formula φ satisfying Assumption 2.3.1, the results in [125, Lemma 2 & Lemma 4] provide a sound transformation Υ to transform STL to RTL[2]:

Theorem 2.3.1 (Sound transformation [125, Theorem 1])**.** *Given a system $\mathcal{H}$ and initial set I, let φ be an STL formula satisfying Assumption 2.3.1 for some value c, and $R(t)$ be the reachable set of Σ in the form of (2.9) with $t_{i+1} - t_i = c$. The transformation Υ from [125], bringing the STL formula φ into an RTL formula $\Psi = \Upsilon(\varphi)$, is sound, i.e.:*

$$\forall \xi \in \mathcal{S}_{\mathcal{H}}(I) : (\xi, t) \models \varphi \impliedby (R,t) \models \Psi. \tag{2.10}$$

[2]The full definition of the transformation Υ is quite involved, yet not essential for the understanding of the ideas presented in this thesis. Therefore, we refer to [125] for the full definition of the transformation Υ.

The transformation Υ from [125] from STL yields RTL formulae of the form

$$\Psi = \bigwedge_{i\in \mathsf{I}} \bigvee_{j\in \mathsf{J}_i} \bigcirc_{j\frac{c}{2}} \bigvee_{k\in \mathsf{K}_{ij}} \mathcal{A}\psi_{ijk}, \tag{2.11}$$

where $\mathsf{I}, \mathsf{J}_i, \mathsf{K}_{ij}$ are finite index sets and ψ_{ijk} are non-temporal subformulae. As can be seen, j closely relates to a time step $c/2$, whereas i and k relate to the number of conjunctions and disjunctions. The reachable set in (2.9) is formed by the reachable sequence $\mathcal{R}$, which partitions time into an alternating sequence of points and open intervals. Similarly, the transformation from STL to RTL transforms the reasoning over an infinite set of time instances to reasoning over an alternating sequence of points and intervals. In this partition, the value $c/2$ can be seen as the time step between the points and a time interval. Due to this partitioning, the transformation Υ is a sound transformation, but in general not complete, i.e., the converse of (2.10) does generally not hold. Therefore, the transformation Υ is subjected to some conservatism[3], which can be reduced by taking smaller values of c.

2.4. SATISFIABILITY MODULO THEORIES SOLVERS

In this thesis we prove properties of systems through Lyapunov-like functions or reachability analysis. However, checking whether a function is a Lyapunov-like function, or whether a reachable set satisfies some (nonlinear conditions), can be nontrivial problems in themselves. These problems boil down to determining whether first-order logic formulae over the reals are satisfied or not. To this end, we use Satisfiability Modulo Theories (SMT) solvers [15], which are tools capable of reasoning over first-order logic formulae, based on a set of background theories. Let us consider the following standard form of a first-order logic formula:

$$\varphi = \forall x \in X : \left(\bigwedge\nolimits_{i=1}^{k} \left(\bigvee\nolimits_{j=1}^{l_i} f_{ij}(x) \leq 0 \right) \right), \tag{2.12}$$

where $X \subseteq \mathbb{R}^n$ and $f_{ij} : \mathbb{R}^n \to \mathbb{R}$. If f_{ij} is polynomial, this problem is decidable, and we can use SMT solvers such as Z3 [32] to prove or disprove such statements. However, for the most general statement of the problem, first-order logic formulae of the form (2.12) are undecidable [50]. To address these general cases, we use a δ-complete decision procedure [50], which determines whether a first-order logic formula is unsatisfiable (unsat) or if the δ-weakening is satisfiable (δ-sat). The δ-weakening can be seen as a perturbed version of the original inequality, which renders the decision process decidable.

A first-order logic formula (2.12) can be verified using a δ-complete decision procedure by using it to prove that $\neg\varphi$ is unsatisfiable. The formula $\neg\varphi$ is equivalent to a logic formula φ' of the form[4]

$$\varphi' := (\exists z \in Z) \left(\bigwedge\nolimits_{i=1}^{k'} \left(\bigvee\nolimits_{j=1}^{l'_i} f'_{ij}(z) = 0 \right) \right), \tag{2.13}$$

[3] If the considered STL fragment is restricted to sampled-time STL [125], the transformation is sound and complete, and therefore no conservatism is introduced.

[4] First, the negated form $\neg\phi$ is rewritten as $\exists x \in X : P(x)$, where $P(x)$ denotes a propositional formula in conjunctive normal form. Secondly, the strict inequalities in $P(x)$ are replaced by equalities by introducing auxiliary variables, as shown in [51, Lemma 2.1].

where $Z \subset \mathbb{R}^{n+m}$, $m = \Sigma_{i=1}^{k'} l_i'$ is the number of introduced auxiliary states, and $f_{ij}' : \mathbb{R}^{n+m} \to \mathbb{R}$. Given this formula, the δ-weakening is given by

$$\varphi^\delta := (\exists z \in Z) \left(\bigwedge_{i=1}^{k'} \left(\bigvee_{j=1}^{l_i'} |f_{ij}'(z)| \leq \delta \right) \right), \tag{2.14}$$

where $\delta \in \mathbb{Q}_{\geq 0}$ is a positive rational number specified by the user. If the δ-complete decision procedure returns unsat for φ' in (2.13), which is equivalent to $\neg\varphi$, we obtain a proof of the satisfiability of the original formula φ in (2.12). Note that unsat and δ-sat are not mutually exclusive. Intuitively, it is possible that a formula is not satisfied, while it is satisfied under a small perturbation. This is illustrated in the following example:

Example 2.4.1. *Let us consider $\forall x \in [-1, 1] : -x^2 \leq 0$, which is obviously true. First, taking the negation, we have*

$$\neg\forall x \in [-1, 1] : -x^2 \leq 0 \equiv \exists x \in [-1, 1] : -x^2 > 0. \tag{2.15}$$

As shown in [51, Lemma 2.1], the right-hand side is equivalent to the following formula, which is in the form in (2.13):

$$\exists x \in [-1, 1], \exists y \in (0, m] : -x^2 - y = 0, \tag{2.16}$$

where $m \in \mathbb{Q}_{\geq 0}$ is any value greater than the maximum of $-x^2$ over $[-1, 1]$, i.e. $m > 0$. This logic formula in (2.16) is unsatisfied, as expected. The δ-weakening is given by

$$\exists x \in [-1, 1], \exists y \in (0, m] : |-x^2 - y| \leq \delta, \tag{2.17}$$

which is satisfied for any arbitrary small value of $\delta \in \mathbb{Q}_{\geq 0}$, hence the answer to the δ-complete decision process is both unsat and δ-sat. Now let us consider

$$\forall x \in [-1, 1] : -x^2 \leq \varepsilon. \tag{2.18}$$

for some $\varepsilon > 0$. Rewriting its negation in the form in (2.13), we get

$$\exists x \in [-1, 1], \exists y \in (0, m'] : -x^2 - \varepsilon - y = 0, \tag{2.19}$$

where $m' \in \mathbb{Q}_{\geq 0}$ is any value greater than the maximum of $-x^2 - \varepsilon$ over $[-1, 1]$, i.e. $m' > 0$. Again, this inequality in (2.19) is unsatisfied. The δ-weakening is given by

$$\exists x \in [-1, 1], \exists y \in (0, m'] : |-x^2 - \varepsilon - y| \leq \delta. \tag{2.20}$$

If δ is chosen such that $\delta \leq \varepsilon$, the δ-weakening is not satisfied, hence the answer to the δ-complete decision process is unambiguously unsat.

In this work we use the SMT solver dReal [51], which implements this δ-complete decision procedure. In case a formula is both unsat and δ-sat, the solver can return either answer. In our synthesis and verification approaches, we circumvent this ambiguity by synthesizing solutions that satisfy the relevant inequalities robustly, such that an overlap between these two cases does not occur. Specifically, this is addressed in Remark 4.5.1 and Section 5.5.3.

3

Grammar-guided genetic programming

In this chapter we outline the main grammar-guided genetic programming algorithm used throughout this thesis. Each step of the algorithm is addressed, including the encoding of individuals, the use of grammars, the selection and genetic operators.

Parts of this chapter have been published in [161] and [157].

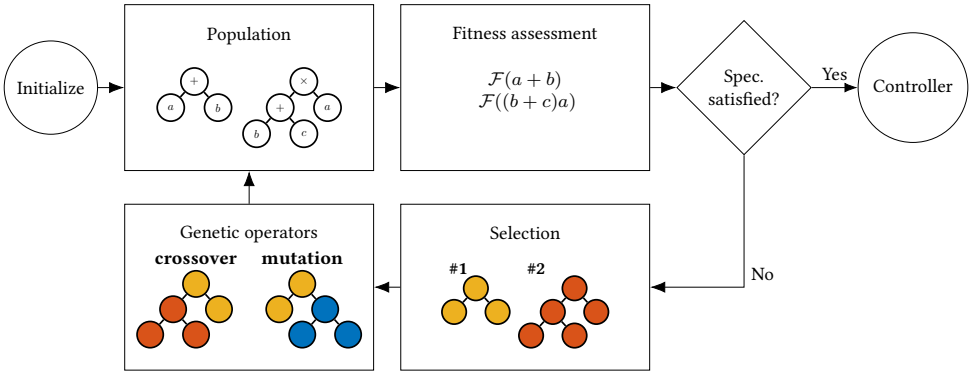


Figure 3.1: Outline of genetic programming.

3.1. Introduction

In this dissertation we design controllers using Genetic Programming (GP) [85], which sets itself apart by evolving entire programs, rather than optimizing parameters within a pre-defined structure, as is typically the case for Genetic Algorithm (GA) or Evolution Strategy (ES). In our case, the evolved program is a controller that is built up and modified (evolved) based on elementary building blocks. These building blocks consist of e.g. the state variables and basic functions such as addition and multiplication.

Evolutionary Algorithms (EAs), including genetic programming, seem to satisfy the *No Free Lunch theorem* [164]. Informally, it entails that the average performance over the entire space of possible problems is equal for all non-revisiting[1] black-box algorithms [43]. However, it is possible to circumvent the No Free Lunch theorem by incorporating problem-specific knowledge. To this end, we use the variant Grammar-Guided Genetic Programming (GGGP) [104, 162], in which we add structure and expert-knowledge to solutions by means of grammars.

This chapter is organised as follows. First, we outline the general outline of genetic programming in Section 3.2. Secondly, we introduce the use of a grammar in Section 3.3. In Section 3.4 we provide general algorithm details. Finally, the chapter is concluded with a discussion on the advantages and limitations of the chosen algorithm in Section 3.5

3.2. Algorithm outline

Within genetic programming, candidate solutions, also referred to as *individuals*, have two types of representation, namely the *phenotype* and *genotype*. The phenotype is the actual solution, in our case the controller in the form of a closed-form expression. The genotype is an encoding of the phenotype in a form that allows for easy manipulation, typically an expression tree. This manipulation is done using so-called *genetic operators*, that e.g. change the genotype of an individual, or recombines multiple genotypes. Given an individual, a metric on how well the objective is achieved is captured in a *fitness function*. In

[1] Non-revisiting means that each candidate solution is only visited once. This can be achieved in EA through the use of an archive.

our context, the fitness function captures how well the controller yields a closed-loop system that satisfies a user-defined control specification. For example, a fitness function for the satisfaction of a temporal logic formula can be defined using the corresponding quantitative semantics (see Section 2.3.1), which gives a score for how well a signal satisfies the formula.

The algorithm is initialized with a randomly generated *population* of individuals. Subsequently, each individual is scored using the fitness function. Depending on their fitness value, individuals can be *selected* to be recombined or modified using *genetic operators*, such as *crossover* and *mutation*. In the former, two subtrees of individuals are interchanged, whereas in the latter, a random subtree is replaced by a new random subtree. Each genetic operator has a user-defined rate, which determines the probability of the operator being applied to the selected individuals. The process of selection and modification through genetic operators is repeated until a new population is created, with the hypothesis that the average fitness of the population increases over cycles, which are referred to as *generations*. This cycle is repeated until a satisfactory individual is found or a maximum number of generations is met. For example, when co-synthesizing Lyapunov functions and controllers, a satisfactory individual consists of a pair that satisfies the Lyapunov function conditions, which can be checked by an SMT solver. Similarly, when synthesizing controllers for STL/RTL specifications, a satisfactory individual results in a reachable set that satisfies the specification. Since the algorithm is not guaranteed find such a solution in a fixed number of generations (see Section 3.5), the user-defined maximum number of generations is to prevent an infinite search.

To summarize, the algorithm goes through the following steps, also illustrated in Figure 3.1:

1. A random population of candidate solutions is generated.
2. For each individual the fitness is computed.
3. A new population is generated by repeatedly selecting individuals and modifying them using genetic operators.
4. Steps 2 to 3 are repeated until a satisfactory solution is obtained, or a maximum number of generations is met.

3.3. GRAMMAR

As stated before, we use grammar-guided genetic programming [104, 162], which imposes that the genotypes of all individuals adhere to a certain grammar. That is, the population is initialized by creating random individuals adhering to the grammar and the used genetic operators are defined such that the resulting individuals also adhere to the grammar. This enables the user to constrain the search space and potentially incorporate expert-knowledge. The used grammar is in Backus-Naur form (BNF) [13], defined by the tuple $(\mathcal{N}, \mathcal{S}, \mathcal{P})$, where $\mathcal{N}$ denotes a set of nonterminals, $\mathcal{S} \in \mathcal{N}$ is a starting tree, and $\mathcal{P}$ are the production rules.

To illustrate the use of grammars, we consider two cases. In Chapter 5, we use a grammar to design time-varying controllers, whereas in Chapter 4 we co-design state-feedback controllers κ and Lyapunov-like functions V, i.e. we use a grammar to design

Nonterminals $\mathcal{N}$ and starting tree $\mathcal{S}$

$\mathcal{N} = \{\langle \text{pol} \rangle, \langle \text{mon} \rangle, \langle \text{const} \rangle\},$
$\mathcal{S} = \langle \text{pol} \rangle\, x_1 + \langle \text{pol} \rangle\, x_2$

Production rules $\mathcal{P}$

$\langle \text{pol} \rangle ::= \langle \text{const} \rangle \mid \langle \text{const} \rangle \times \langle \text{mon} \rangle \mid \langle \text{pol} \rangle + \langle \text{pol} \rangle$
$\langle \text{mon} \rangle ::= t \mid t \times \langle \text{mon} \rangle$
$\langle \text{const} \rangle ::= \text{RandomReal} \in [-10, 10]$

(a)

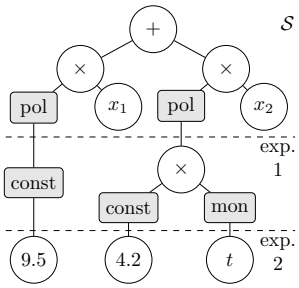
Nonterminals $\mathcal{N}$ and starting tree $\mathcal{S}$

$\mathcal{N} = \{\langle \text{pol} \rangle, \langle \text{mon} \rangle, \langle \text{lin} \rangle, \langle \text{var} \rangle, \langle \text{const} \rangle\},$
$\mathcal{S} = \text{Tuple}(\langle \text{pol} \rangle, \langle \text{lin} \rangle)$

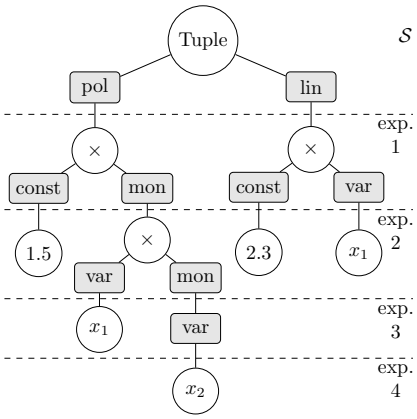
Production rules $\mathcal{P}$

$\langle \text{pol} \rangle ::= \langle \text{const} \rangle \mid \langle \text{const} \rangle \times \langle \text{mon} \rangle \mid \langle \text{pol} \rangle + \langle \text{pol} \rangle$
$\langle \text{mon} \rangle ::= \langle \text{var} \rangle \mid \langle \text{var} \rangle \times \langle \text{mon} \rangle$
$\langle \text{lin} \rangle ::= \langle \text{const} \rangle \times \langle \text{var} \rangle \mid \langle \text{lin} \rangle + \langle \text{lin} \rangle$
$\langle \text{var} \rangle ::= x_1 \mid x_2$
$\langle \text{const} \rangle ::= \text{RandomReal} \in [-10, 10]$

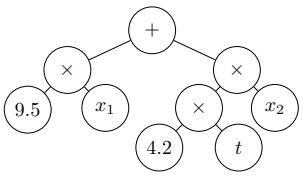
(b)



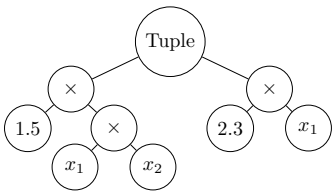
(c)



(d)



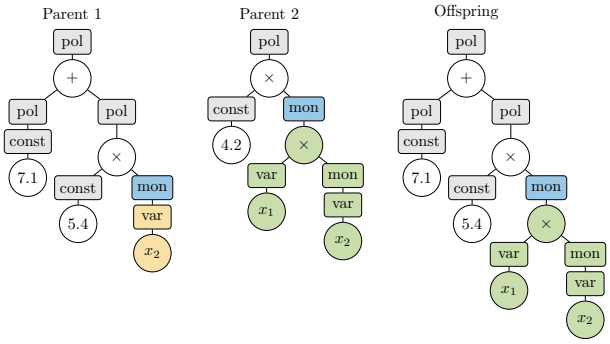
(e)



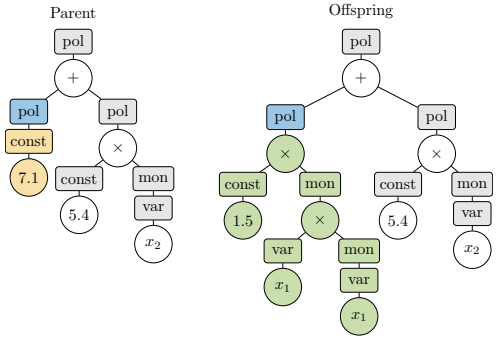
(f)

Figure 3.2: Examples of grammars for (a) time-varying linear controllers and (b) tuple of polynomial Lyapunov-like function V and linear controller κ. Corresponding fully expanded genotypes are shown in (c) and (d). To obtain the phenotype, first the nonterminal nodes are removed, resulting in the expression trees in (e) and (f). The corresponding phenotypes are $9.5x_1 + 4.2tx_2$ and $(1.5x_1x_2, 2.3x_1)$ for (c) and (d), respectively.

tuples (V, κ). An example of the former is shown in Figure 3.2a. In this grammar, the nonterminals correspond to polynomials $\langle \text{pol} \rangle$, monomials $\langle \text{mon} \rangle$ over the variable time t, and constants $\langle \text{const} \rangle$. The starting tree $\mathcal{S}$ restricts the class of controllers to time-varying state-feedback laws, linear in the state $x \in \mathbb{R}^2$. An example of a grammar to design the tuple (V, κ) is shown in Figure 3.2b. Similarly, the nonterminals $\langle \text{pol} \rangle$, $\langle \text{mon} \rangle$,



(a) Crossover.



(b) Mutation.

Figure 3.3: Illustrative examples of the grammar-aware crossover and mutation operators, adhering to the production rules in 3.2b.

$\langle \text{const} \rangle$ denote polynomials, monomials and constants. Additionally, the nonterminals $\langle \text{lin} \rangle$ and $\langle \text{var} \rangle$ denote linear expressions and variables, respectively. Given the starting tree $\mathcal{S}$, the tuple (V, κ) is restricted to polynomial Lyapunov-like functions and linear state controllers.

Given the grammar, a genotype is constructed as follows: beginning with the starting tree, for all leaf nodes containing a nonterminal, a subtree is randomly selected from the corresponding production rules and put under the leaf node. This procedure is repeated until all leaf nodes are free of nonterminals. To prevent infinite depth trees due to recursive production rules, all recursive rules are omitted from the production rules after a fixed number of expansions of the nonterminals. Given the grammars in Figures 3.2a and 3.2b, examples of corresponding fully expanded genotypes are shown in Figures 3.2c and 3.2d. To obtain the phenotype from a genotype, first all nonterminal nodes are replaced with their underlying subtrees. This is illustrated for the genotypes in Figures 3.2c and 3.2d in Figures 3.2e and 3.2f. Given this form, a phenotype is obtained by rewriting the resulting expression tree as an analytic expression. The phenotypes corresponding to the genotypes in Figures 3.2c and 3.2d are given by $9.5x_2 + 4.2tx_2$ and $(1.5x_1x_2, 2.3x_1)$, respectively.

In this work, the considered genetic operators are defined such that the resulting trees still adhere to the same grammar as before. We consider crossover and mutation. In the crossover operator, given two individuals, for each individual a random subtree with the same nonterminal root is selected and these are interchanged. In the mutation operator, a random subtree is selected and replaced with a newly grown subtree with the same nonterminal root. Illustrative examples of both operators are shown in Figure 3.3.

3.4. Algorithm details

In this section we discuss key elements within the GP algorithm and the choices we have made in this thesis. We limit our scope to the methods employed in this dissertation. For reviews on challenges and approaches within genetic programming we refer to, e.g., [31, 109].

3.4.1. Selection

The purpose of selection is to select a candidate solution with a proportionate likelihood to its fitness value. In order to form a full new population, this selection is repeated multiple times. Several selection methods have been proposed, with the most well known being proportional selection, ranking selection, and tournament selection [43, 58, 70]. In this work we use tournament selection, in which a fixed number of individuals are randomly chosen from the population, and the individual with the highest fitness is returned as the selected individual. The benefits of this approach are the simple implementation and its low computation cost.

3.4.2. Multi-objective optimization

Throughout this dissertation, we sometimes have multiple fitness criteria, resulting in a multi-objective optimization (MOO). These types of problems arise naturally in genetic programming; besides optimizing a certain cost function natural to the problem, it might also be desired to minimize the complexity or the number of parameters of an automatically synthesized expression. In our context, this can be beneficial, as it can result in controllers that are e.g. easier to interpret or require less memory in its implementation. MOO is a widely studied field in EA, and multiple approaches have been proposed, which can be classified as classical approaches and contemporary approaches [31]. In the first class, a trade-off between all objectives is (implicitly) made a priori by formulating a single fitness function, whereas in the second class all objectives are optimized simultaneously, resulting in a set of optimal solutions. An overview of approaches can be found in [31, 42, 44, 106]. The approaches relevant for this work are briefly highlighted below.

Single fitness function In Chapter 4, the fitness function is based on the satisfaction of multiple conditions on a Lyapunov-like function. Each condition is captured in an independent fitness measure, indicating its (level of) satisfaction; the maximum fitness value implies the satisfaction of the condition, whereas lower values imply that the condition is not met. All conditions need to be eventually satisfied, i.e. attain the maximum fitness value, thus no trade-off has to be made. Therefore, all measures are combined into a single fitness function. To aid convergence, we adopt a weighted aggregation methodology

[72]. That is, the total fitness is dependent on a weighted sum of all sub measures, where the weights vary over time. Our specific approach is described in further detail in Section 4.5.2.

Pareto-aware optimization In most cases, multiple objectives cannot be resolved into one fitness function without resulting in an implicit trade-off between the different objectives. An example arising naturally in our context is the trade-off between controller performance and complexity of the expression. In this case we utilize a contemporary approach based on Pareto optimality, in which a set of optimal controllers is returned, rather than a singular controller. A solution is Pareto optimal (or non-dominated) if there exists no Pareto-dominating solution, which is defined as follows:

Definition 3.4.1 (Pareto dominance)**.** *Given a multi-objective minimization problem and two objective vectors $x, x' \in \mathbb{R}^n$, the vector x is said to Pareto dominate x' if*

1. $\forall i \in \{1, \ldots, n\} : x_i \leq x'_i$,
2. $\exists j \in \{1, \ldots, n\} : x_j < x'_j$.

An overview of Pareto-aware methods for evolutionary algorithms can be found in, e.g., [31, 44, 106, 108]. In this work we use the non-dominated sorting algorithm NSGA-II [33], a Pareto optimal-aware sorting algorithm, which ranks candidate controllers based on the Pareto optimality of all criteria. This rank is then used as the fitness value within the selection. While we resort to NSGA-II, this method can be interchanged by any other Pareto-aware approach.

Secondary fitness measures In Pareto-aware ranking or ranking-based selection such as tournament selection, secondary fitness measures are used to break the tie in case multiple individuals have the same fitness. In the case of NSGA-II, candidates are primarily ranked based on their Pareto optimality, and secondarily on the crowding distance [33], which is an indication of the uniqueness of the solution. In Section 4.5.2 we discuss the secondary fitness measures used in the certificate-based synthesis.

3.4.3. Parameter optimization

In classical genetic programming, the constants within a solution are derived from the recombination of a set of pre-defined constants. However, this constant generation has proven to be inefficient [85, 132] and therefore alternative approaches have been proposed [31]. A common approach is the hybridization of genetic programming with a local search technique [61], in which the constants are optimized for every genetic programming generation, using the local search technique. In this work we adopt such a hybridization method and use Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [67] as the local search method. CMA-ES is a differentiation-free optimization method, regarded to be robust with respect to discontinuous fitness functions [66]. More specifically, we use the variant sep-CMA-ES [128], which has the advantage that it is linear in space and time complexity. In this method it is assumed that the parameters are independent. While this assumption generally does not hold for our applications, the advantage of the complexity outweighs the loss in performance.

3.5. Discussion

Convergence properties of evolutionary algorithms, including genetic programming and evolution strategies, have been studied, see e.g., [78, 129, 130]. In [78], it has been proven that candidate solutions within the population asymptotically converge to a global optimum within the solution space spanned by the grammar. Intuitively, the proofs go along the line that, due to infinite exploration thanks to random search, if time goes to infinity, the entire search space is explored. However, these asymptotic convergence results are irrelevant to the scope of this work, since we are only interested in finding a solution within a finite number of generations. This is strengthened by an 'all or nothing' requirement to the solutions: if the controller does not satisfy the specification, it is essentially useless. When considering GP with a finite number of generations, it is not a complete method, i.e. there is no guarantee a solution is found within a finite number of generations, even if it exists within the search space. Regardless, the benefits of GP, namely the automatic exploration of solution structures and the provision of closed-form solutions, outweigh the lack of completeness.

4

Certificate-based synthesis

In this chapter a framework is proposed for automatic formal controller synthesis for general hybrid systems with a subset of safety and reachability specifications. The framework uses genetic programming to automatically co-synthesize controllers and candidate Lyapunov-like functions. These candidate Lyapunov-like functions are used to formally verify the control specification, and are verified themselves using an SMT solver. The advantages of this approach are that very few restrictions are made to the class of systems and solutions, the synthesized controllers are expressed as compact expressions, and no explicit solution structure has to be specified beforehand. We demonstrate the effectiveness of the proposed framework in several case studies, including non-polynomial systems, sampled-data systems, systems with bounded uncertainties, switched systems, and systems with jumps.

Parts of this chapter have been published in [159–161]. Specifically, Sections 4.2 to 4.6 are verbatim from [161], Section 4.7 from [159] and Section 4.8 from [160].

4.1. Introduction

In this chapter we utilize the paradigm of certificate functions, without the need to restrict the class of functions present in the dynamics and solutions. Inspired by the idea of counterexample-guided inductive synthesis (CEGIS) (see e.g. [121], [74]), we propose a framework that relies on a combination of Genetic Programming (GP) and Satisfiability Modulo Theories (SMT) solvers [15] to automatically synthesize and verify controllers for hybrid systems. This is done by co-designing a controller and Lyapunov barrier-like function. The subclass of temporal logic specifications we deal with are simple safety and reachability specifications. Using genetic programming, the structure of a solution is evolved, and therefore is not required to be specified beforehand. This is particularly useful when no solution exists in a certain solution parameterization, e.g. a second-order polynomial, as the algorithm explores other structures automatically. Finally, the resulting controllers are compact analytic expressions, as opposed to the abstraction-based methods.

Several studies have proposed CEGIS based on SMT solvers for the synthesis of (control) certificate functions. In [74] Lyapunov and barrier functions are synthesized for the verification of continuous-time switched systems. In [118] and [121], SMT solvers have been used for counterexample-guided synthesis of control Lyapunov barrier-like functions for switched nonlinear systems. Our main contributions are the extension to general hybrid systems, as well as being able to evolve the solutions structure.

This chapter is organized as follows. First, the class of systems (i.e. hybrid systems) and the subclass of specifications (i.e. reachability and safety) are formally defined in the problem definition in Section 4.2. Secondly, in Section 4.3, we introduce a Lyapunov barrier function which infers the desired system specifications, followed by various relaxations in Section 4.4. The automatic synthesis approach is described in Section 4.5, followed by case studies, including continuous-time systems, sampled-data systems, systems subjected to bounded disturbances, switching controllers and full jump-flow systems. In Section 4.7 we introduce a specialized approach for continuous-time sampled-data systems. In Section 4.8 we use the proposed framework to verify near-optimal controllers which were designed using reinforcement learning. Finally, the chapter is concluded in Section 4.9.

4.2. Problem definition

Let us consider a state space $\mathbb{R}^n$, input space $\mathbb{R}^m$ and output space $\mathbb{R}^l$. Given a flow set C, jump set D, and open-loop flow map $F_{\text{ol}} : \mathbb{R}^n \times \mathbb{R}^m \rightrightarrows \mathbb{R}^n$, open-loop jump map $G_{\text{ol}} : \mathbb{R}^n \times \mathbb{R}^m \rightrightarrows \mathbb{R}^n$ and output map $h : \mathbb{R}^n \to \mathbb{R}^l$, in this chapter we propose a methodology to design a static output-feedback controller $\kappa : \mathbb{R}^l \to \mathbb{R}^m$, resulting in a closed-loop hybrid system $\mathcal{H}_{\text{cl}} = (C, F, D, G)$ with $F(s) = F_{\text{ol}}(s, \kappa \circ h(s))$ and $G(s) = G_{\text{ol}}(s, \kappa \circ h(s))$. In this work, controllers are designed for specifications in terms of safety w.r.t. a safe set and reachability w.r.t. a goal set for solutions starting in an initial set. We consider compact safe sets $S \subset C \cup D$, compact initial sets $I \subset S$ and compact goal sets $O \subset S$, which can be represented as

$$Y = \left\{ s \in C \cup D \;\middle|\; \bigwedge\nolimits_{i=1}^{i_Y} b_{Y,i}(s) \leq 0 \right\} \tag{4.1}$$

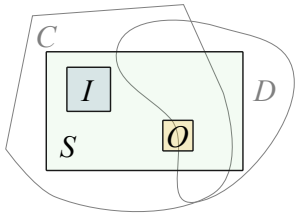


Figure 4.1: Example of a flow set C, jump set D, safe set S, initial set I and goal set O.

for $Y \in \{S, I, O\}$, with $i_Y > 0$ and $b_{Y,i} : \mathbb{R}^n \to \mathbb{R}$ for all $i \in \{1, \dots i_Y\}$. The main reason for choosing bounded sets is for numerical and practical reasons within the automatic synthesis and verification. An example of a safe, initial and goal set projected on a 2D space is shown in Figure 4.1.

For a solution $\phi : E \to \mathbb{R}^n$, let us define the hybrid time intervals $E_{\leq(T,J)} := E \cap ([0,T] \times [0,J])$ and $E_{\geq(T,J)} := E \backslash ([0,T) \times [0,J))$. Now given the safe, initial and goal sets, and solutions $\phi : E \to \mathbb{R}^n$, consider the following desired closed-loop specifications:

$\texttt{CS}_1$ *Reach while stay (RWS)*: all maximal solutions ϕ to $\mathcal{H}_{\text{cl}}$ starting from the initial set I eventually reach the goal set O, while staying within the safe set S:

$$\begin{aligned} \forall \phi \in \mathcal{S}_{\mathcal{H}_{\text{cl}}}(I), \exists (T,J) \in E, \forall (t,j) \in E_{\leq(T,J)} : \\ \phi(t,j) \in S \wedge \phi(T,J) \in O. \end{aligned} \tag{4.2}$$

$\texttt{CS}_2$ *Reach and stay while stay (RSWS)*: all maximal solutions ϕ to $\mathcal{H}_{\text{cl}}$ starting from the initial set I eventually reach and stay in the goal set O, while always staying within the safe set S:

$$\begin{aligned} \forall \phi \in \mathcal{S}_{\mathcal{H}_{\text{cl}}}(I), \exists (T,J) \in E, \forall (t,j) \in E, \\ \forall (a,b) \in E_{\geq(T,J)} : \phi(t,j) \in S \wedge \phi(a,b) \in O. \end{aligned} \tag{4.3}$$

These specifications $\texttt{CS}_1$ and $\texttt{CS}_2$ can be rewritten in STL, defined in Section 2.3, as:

$$\forall \phi \in \mathcal{S}_{\mathcal{H}_{\text{cl}}}, (\xi_\phi, t) \models \varphi_{\texttt{CS}_i}, \quad i = 1, 2, \tag{4.4}$$

with set-valued function $\xi_\phi(t') = \{\phi(t', j) \mid (t', j) \in E\}$, and

$$\begin{aligned} \varphi_{\texttt{CS}_1} &= \varphi_S \mathcal{U}_{[0,\infty)} \varphi_O, \\ \varphi_{\texttt{CS}_2} &= \Box_{[0,\infty)} \varphi_S \wedge \Diamond_{[0,\infty)} \Box_{[0,\infty)} \varphi_O. \end{aligned}$$

Note that satisfying specification $\texttt{CS}_1$ or $\texttt{CS}_2$ does not preclude that complete solutions of system $\mathcal{H}_{\text{cl}}$ exhibit Zeno behavior. Corollaries 4.3.2 and 4.4.2 will address this issue. Moreover, note that specification $\texttt{CS}_2$ does not impose that solutions should stay in O after the first time instant it enters O, but rather that for each solution there exists a time instant $(T, J) \in E$ after which it stays in O. With the definition of the system and specifications, we are ready to define the following problem:

Problem 4.2.1

Given a specification $\texttt{CS}_1$ or $\texttt{CS}_2$ w.r.t. compact sets (S, I, O) and the open-loop system $(C, F_{\text{ol}}, D, G_{\text{ol}}, h)$, synthesize an analytic controller $\kappa : \mathbb{R}^l \to \mathbb{R}^m$ such that the closed-loop system satisfies the specification.

Next to synthesizing the controller for the flow/jump map, in some applications it is desired to design the flow set and jump set as part of the hybrid controller, for example in the synthesis of a supervisory controller that determines which controller mode should be active. Consider open-loop flow and jump sets $C_{\text{ol}},\ D_{\text{ol}}$ dependent on the controller $\kappa : \mathbb{R}^l \to \mathbb{R}^m$ such that $C = C_{\text{ol}}(\kappa \circ h(x)),\ D = D_{\text{ol}}(\kappa \circ h(x))$. This yields the following variation of Problem 4.2.1:

4

Problem 4.2.2

Given a specification $\texttt{CS}_1$ or $\texttt{CS}_2$ w.r.t. compact sets (S, I, O) and the open-loop system $(C_{\text{ol}}, F_{\text{ol}}, D_{\text{ol}}, G_{\text{ol}}, h)$, synthesize an analytic controller $\kappa : \mathbb{R}^l \to \mathbb{R}^m$ such that the closed-loop system satisfies the specification.

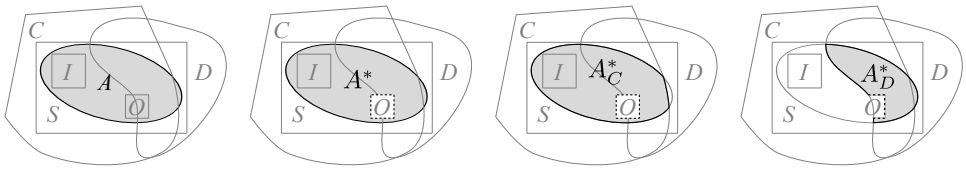
4.3. Lyapunov barrier functions

In this chapter we verify specification $\texttt{CS}_1$ or $\texttt{CS}_2$ by means of a Lyapunov barrier function (LBF), introduced in this section, which is co-evolved with the controller. In this section, we present an LBF in Definition 4.3.1 and present relaxations thereof in Section 4.4. The proofs of the technical results are presented in Appendix A. Definition 4.3.1 is similar to Lyapunov and/or barrier functions for hybrid systems as proposed in [64, 65, 115], to which we consider slight modifications for the purpose of automatic synthesis. In particular, the LBF conditions are posed as nonlinear inequalities over the reals, which are in general not decidable. Therefore, the synthesis and verification rely on δ-decidability instead, in which a perturbed version of the inequalities are used, see Section 2.4. As a consequence, the LBF conditions are proposed with this constraint in mind. With a similar reasoning, we assume that the goal set O has a nonempty interior. As remarked earlier, with the purpose of using SMT solvers to verify the conditions, we assume that the sets (S, I, O) are compact. Consider the following assumption:

Assumption 4.3.1 (Specification sets assumption). *The compact sets (S, I, O) can be expressed in the form (4.1), $S \subset C \cup D$, $O, I \subseteq int(S)$ and $int(O) \neq \emptyset$.*

Remark 4.3.1 (Existence of solutions). *Under the hybrid basic conditions on $\mathcal{H}_{\text{cl}}$, it follows from Proposition 6.10 in [57] that for all $s \in I \subseteq int(S) \subset C \cup D$, there exists a nontrivial solution ϕ to $\mathcal{H}_{\text{cl}}$ with $\phi(0, 0) = s$.*

Definition 4.3.1 (Lyapunov barrier function). *A function $V \in \mathcal{C}^1(S, \mathbb{R})$ is a Lyapunov barrier function w.r.t. the compact sets (S, I, O) and system $\mathcal{H}_{\text{cl}}$, if there exist $\gamma_{\text{c}}, \gamma_{\text{d}} > 0$*



(a) Sublevel set A. (b) Set $A^* := A \backslash O$. (c) Set $A^*_C := A^* \cap C$. (d) Set $A^*_D := A^* \cap D$.

Figure 4.2: Example of the sublevel set A and sets A^*, A^*_C and A^*_D.

such that

$$\forall s \in I : \; V(s) \leq 0, \tag{4.5a}$$
$$\forall s \in \partial S : \; V(s) > 0, \tag{4.5b}$$
$$\forall s \in A^*_D : \; G(s) \subseteq S, \tag{4.5c}$$
$$\forall s \in A^*_C, \forall f \in F(s) : \langle \nabla V(s), f \rangle \leq -\gamma_{\mathrm{c}}, \tag{4.5d}$$
$$\forall s \in A^*_D, \forall g \in G(s) : V(g) - V(s) \leq -\gamma_{\mathrm{d}}, \tag{4.5e}$$

where $A^* := A \backslash O$, *for* $Y \in \{C, D\}$, $A^*_Y := A^* \cap Y$ *and*

$$A := \{s \in S \mid V(s) \leq 0\}. \tag{4.6}$$

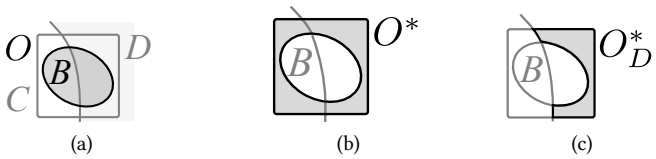
The sublevel set A and its subsets are illustrated in Figure 4.2. Set A is in some sense similar to both a basin of attraction of O and a forward invariant set (up until the goal set is reached), and it contains the initial set I, as by condition (4.5a). The basin of attraction-like nature stems from (4.5d) and (4.5e), which impose that during flow and jumps the value of the LBF decreases. The forward invariant-like nature of A stems from conditions (4.5b) and (4.5c), which impose that during flow and jumps, solutions cannot leave the safe set S and due to the decrease need to remain within A. Finally, it can be proven that these properties are sufficient to imply that trajectories eventually have to enter O while staying in S, as is formalized in Theorem 4.3.1 and its proof.

Remark 4.3.2 (Parameter choice)**.** *Without loss of generality, we can select* γ_{c} *and* γ_{d} *to be equal, as we can always select the minimum of the two, i.e.* $\gamma = \min(\gamma'_{\mathrm{c}}, \gamma'_{\mathrm{d}})$*. Subsequently, the choice of this* γ *is arbitrary, because if a solution* V^* *exists for* γ^**, there always exists a linear transformation of* V^* *such that the inequalities in* (4.8) *are satisfied for any* γ*.*

An LBF can be used to verify specification $\mathtt{CS}_1$, as shown in the following theorem.

Theorem 4.3.1 (Reach while stay)**.** *Given the closed-loop system* $\mathcal{H}_{\mathrm{cl}}$*, if there exists an LBF* V *w.r.t. compact sets* (S, I, O) *satisfying Assumption 4.3.1, then the closed-loop system satisfies* (4.2)*.*

The LBF implies that states within the sublevel set A enter the goal set O in finite time. However, it does not imply that trajectories entering O stay there, nor stay in the safe set. The following corollary to Theorem 4.3.1 gives sufficient conditions such that specification $\mathtt{CS}_2$ is enforced.



(a) (b) (c)

Figure 4.3: Given the sets from Figure 4.2a, an example of (a) the sublevel set B, (b) set $O^* := O \backslash int(B)$, and (c) set $O^*_D := O^* \cap D$.

Corollary 4.3.1 (Reach and stay while stay). *Given a closed-loop system $\mathcal{H}_{\text{cl}}$ and LBF V w.r.t. compact sets (S, I, O) satisfying Assumption 4.3.1, if $\exists \beta \in \mathbb{R}$ such that V additionally satisfies*

$$\forall s \in O^*_D : G(s) \subseteq S, \tag{4.7a}$$
$$\forall s \in O^*_C, \forall f \in F(s) : \langle \nabla V(s), f \rangle \leq -\gamma_{\text{c}}, \tag{4.7b}$$
$$\forall s \in O^*_D, \forall g \in G(s) : V(g) - V(s) \leq -\gamma_{\text{d}}, \tag{4.7c}$$
$$\forall s \in \partial O : \; V(s) > \beta, \tag{4.7d}$$
$$\forall s \in B \cap D : \; G(s) \subseteq B, \tag{4.7e}$$

where $B := \{s \in O \mid V(s) \leq \beta\}$, $O^* = O \backslash int(B)$ and for $Y \in \{C, D\}$, $O^*_Y = O^* \cap Y$, then the closed-loop system $\mathcal{H}_{\text{cl}}$ satisfies (4.3).

The sublevel set B and some subsets are illustrated in Figure 4.3. Set B is a forward invariant set inside the interior of O and all maximum solutions starting in I enter this set within finite time. Intuitively, (4.7a)-(4.7c) extend the set on which conditions (4.5c)-(4.5e) hold to include $O \backslash int(B)$. Condition (4.7d) and (4.7a) render B forward invariant, similarly to the role of conditions (4.5b) and (4.5c) w.r.t S. Together, they imply that solutions enter a forward invariant subset of O.

Specification $\texttt{CS}_2$ reasons over maximal solutions, but it does not exclude the possibility of Zeno behavior, as shown in the following example:

Example 4.3.1 (Zeno behavior). *Consider a hybrid system with $G(s, u) = 0$ and $D = \{0\}$. This system admits complete solutions that are Zeno, i.e. infinite jumps within a finite time interval, as each jump goes into the jump set. Now if the goal set is defined such that $D \subseteq O$, the existence of an LBF which satisfies the additional conditions (4.7) is not contradicted by G, D and O. Therefore, an LBF satisfying Corollary 4.3.1 is not sufficient to exclude the admittance of Zeno solutions.*

The next corollary to Theorem 4.3.1 establishes a sufficient condition on V such that the maximal solutions are non-Zeno. We provide no proof for this result, as it is analogous to the proof of Corollary 4.4.2 in the next Section.

Corollary 4.3.2 (Zeno-free maximal solutions). *Given a closed-loop system $\mathcal{H}_{\text{cl}}$, LBF V w.r.t. compact sets (S, I, O) satisfying Assumption 4.3.1, and there exists a $\beta \in \mathbb{R}$ such that V satisfies (4.9), if $B \cap D = \emptyset$, all solutions $\phi \in \mathcal{S}_{\mathcal{H}_{\text{cl}}}(I)$ are non-Zeno.*

4.4. RELAXATIONS

In this section, without limiting the class of considered systems, the system states are explicitly divided into three types, allowing for specialized and relaxed Lyapunov-like conditions. Given a solution $\phi(t,j)$, we distinguish continuous states $\phi_{\mathrm{x}}(t,j)$, discrete states $\phi_{\mathrm{q}}(t,j)$ and timer states $\phi_{\mathrm{t}}(t,j)$. The continuous states can change during both flow and jumps, whereas the discrete states can only change during jumps. The timer states increase at a constant rate during flow and each timer state $\phi_{\mathrm{t},i}(t,j)$ is reset after η_i seconds. Now, $\phi(t,j)$ is partitioned as

$$\begin{aligned}\phi(t,j) &= (\phi_{\mathrm{x}}(t,j), \phi_{\mathrm{q}}(t,j), \phi_{\mathrm{t}}(t,j)),\\ \phi_{\mathrm{x}}(t,j) &\in \mathcal{X} \subseteq \mathbb{R}^{n_{\mathrm{x}}},\ \phi_{\mathrm{q}}(t,j) \in \mathcal{Q} \subseteq \mathbb{R}^{n_{\mathrm{q}}},\\ \phi_{\mathrm{t}}(t,j) &\in \mathcal{T} := \Pi_{i=1}^{n_{\mathrm{t}}}[0,\eta_i],\ \eta_i > 0.\end{aligned}$$

Here $C \cup D \subseteq \mathcal{X} \times \mathcal{Q} \times \mathcal{T} \subseteq \mathbb{R}^n$ and $n = n_{\mathrm{x}} + n_{\mathrm{q}} + n_{\mathrm{t}}$. Similarly, a point $s \in \mathbb{R}^n$ is partitioned as $s = (s_{\mathrm{x}}, s_{\mathrm{q}}, s_{\mathrm{t}})$. The timer reset motivates the distinction in two types of jumps: a timer jump if $\phi(t,j) \in D_{\mathrm{t}}$ is induced by the timer resets, and a system jump if $\phi(t,j) \in D_{\mathrm{s}}$ is induced by the system states $(\phi_{\mathrm{x}}, \phi_{\mathrm{q}})$. During system jumps, the timer states remain constant, whereas during timer jumps, the timer state that triggered the jump is reset to zero. This yields the following system structure:

$$F_{\mathrm{ol}}(s,u) = \left(F_{\mathrm{ol,x}}(s,u), \mathbf{0}_{n_{\mathrm{q}}}, \mathbf{1}_{n_{\mathrm{t}}}\right),$$

$$G_{\mathrm{ol}}(s,u) = \begin{cases} G_{\mathrm{ol,s}}(s,u), & \text{if } s \in D_{\mathrm{s}} \backslash D_{\mathrm{t}},\\ G_{\mathrm{ol,t}}(s,u), & \text{if } s \in D_{\mathrm{t}} \backslash D_{\mathrm{s}},\\ G_{\mathrm{ol,s}}(s,u) \cup G_{\mathrm{ol,t}}(s,u), & \text{if } s \in D_{\mathrm{s}} \cap D_{\mathrm{t}}, \end{cases}$$

$$G_{\mathrm{ol,s}}(s,u) = \left(G^{\mathrm{xq}}_{\mathrm{ol,s}}(s,u), s_{\mathrm{t}}\right), \quad G_{\mathrm{ol,t}}(s,u) = \left(G^{\mathrm{xq}}_{\mathrm{ol,t}}(s,u), \mathrm{reset}(s_{\mathrm{t}})\right),$$

$$D_{\mathrm{s}} \subseteq \mathcal{X} \times \mathcal{Q} \times \mathcal{T}, \quad D_{\mathrm{t}} \subseteq \bigcup\nolimits_{i=1}^{n_{\mathrm{t}}} D_{\mathrm{t},i},$$

$$D_{\mathrm{t},i} \subseteq \mathcal{X} \times \mathcal{Q} \times \Pi_{k=1}^{i-1}[0,\eta_k] \times \{\eta_i\} \times \Pi_{k=i+1}^{n_{\mathrm{t}}}[0,\eta_k]$$

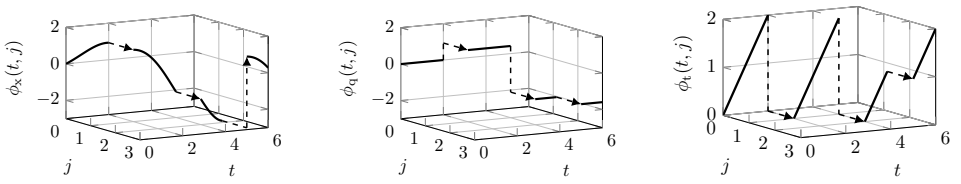
$$\mathrm{reset}(c) = (\mathrm{reset}_1(c_1), \ldots, \mathrm{reset}_{n_{\mathrm{t}}}(c_{n_{\mathrm{t}}})),$$

$$\mathrm{reset}_i(c_i) = \begin{cases} 0 & \text{if } c_i = \eta_i,\\ c_i & \text{otherwise,} \end{cases}$$

where $F_{\mathrm{ol,x}} : \mathbb{R}^n \times \mathbb{R}^m \rightrightarrows \mathbb{R}^{n_{\mathrm{x}}}$ denotes the flow map for the continuous states and $G^{\mathrm{xq}}_{\mathrm{ol,s}}, G^{\mathrm{xq}}_{\mathrm{ol,t}} : \mathbb{R}^n \times \mathbb{R}^m \rightrightarrows \mathbb{R}^{n_{\mathrm{x}}+n_{\mathrm{q}}}$ are the jump maps for the continuous and discrete states, triggered by the system states and timer state. The jump set of the entire system is given by $D = D_{\mathrm{s}} \cup D_{\mathrm{t}}$. The distinction between continuous, discrete and timer states and their respective behavior is illustrated in Figure 4.4. In the remainder we use the notation $G_{\mathrm{s}}(s) = G_{\mathrm{ol,s}}(s, \kappa \circ h(s))$ and $G_{\mathrm{t}}(s) = G_{\mathrm{ol,t}}(s, \kappa \circ h(s))$ for the jump maps of the closed-loop system.

Examples of states that could be modeled as discrete states include logic states, discrete states, and sampled states for sampled-data systems. The timer state can be used to model the sample update of sampled-data systems.

Remark 4.4.1 (Absence of state types)**.** *We allow the possibility for n_{x}, n_{q}, n_{t} to be zero, i.e. the absence of continuous, discrete or timer states. Subsequently, with abuse of notation,*



(a) Continuous state $\phi_x(t,j)$. (b) Discrete state $\phi_q(t,j)$. (c) Timer state $\phi_t(t,j)$.

Figure 4.4: Example of the evolution of the different types of states for a sampled-data system with the sampled state as discrete state. The system is subjected to $D_s = \{s \in \mathbb{R}^3 \mid s_x \leq -3\}$, $D_t = \mathbb{R}^2 \times \{2\}$, $G_s(s) = (1, s_q, s_t)$, $G_t(s) = (s_x, s_x, 0)$, resulting in timer state-induced jumps $j \in \{1, 2\}$ and system state-induced jump $j = 3$.

4

we define for the corresponding 'non-existing' space $\mathbb{R}^0$ such that $A \times \mathbb{R}^0 := A$. Note that the object $\mathbb{R}^0$ is not equal to the empty set, as $A \times \emptyset = \emptyset$.

For the three types of states, we assume that the safe set, initial set and goal set satisfy the following assumption, which helps to further relax the conditions on the candidate LBF.

Assumption 4.4.1 (Specification sets assumption revised)**.** *Given compact sets $S_x \subseteq \mathcal{X}$, $I_x \subset int(S_x)$, $O_x \subset int(S_x)$, $S_q \subseteq \mathcal{Q}$, and $O_q \subseteq S_q$, the compact safe, initial and goal sets (S, I, O) can be expressed as in the form in (4.1) and are defined such that:*

1. $S := S_x \times S_q \times \mathcal{T} \subseteq C \cup D$.
2. $I \subseteq I_x \times S_q \times \mathcal{T} \subset S$.
3. $O := O_x \times O_q \times \mathcal{T} \subset S$ *and* $int(O_x) \neq \emptyset$.

Here S_x, I_x and O_x are the safe, initial and goal set of the continuous states and S_q and O_q the safe and goal set of the discrete states. Note that by definition the entire timer state space is considered to be in the safe and goal set.

Remark 4.4.2 (Existence of solutions, revisited)**.** *Analogous to Remark 4.3.1, for all $s \in I \subseteq (int(S_x) \times S_q \times \mathcal{T}) \subset C \cup D$, there exists a nontrivial solution ϕ to $\mathcal{H}_{cl}$ with $\phi(0,0) = s$.*

The explicit division between continuous, discrete and timer states allows for relaxations on the conditions on the candidate LBF. The proofs are presented in the Appendix A.

Proposition 4.4.1 (Sufficient conditions for RWS)**.** *Given the closed-loop system $\mathcal{H}_{cl}$ and compact sets (S, I, O) satisfying Assumption 4.4.1, if there exists a candidate LBF V that satisfies (4.5a), (4.5c), (4.5d) and*

$$\forall s \in \partial S_x \times S_q \times \mathcal{T} : \; V(s) > 0, \tag{4.8a}$$

$$\forall s \in A^*_{D_s}, \forall g_s \in G_s(s) : V(g_s) - V(s) \leq -\gamma_d, \tag{4.8b}$$

$$\forall s \in A^*_{D_t}, \forall g_t \in G_t(s) : V(g_t) - V(s) \leq 0, \tag{4.8c}$$

where for $Y \in \{D_s, D_t\}$, $A^*_Y := A^* \cap Y$, then the closed-loop system satisfies (4.2).

Compared to the original LBF, it is sufficient if $V(s) > 0$ holds only at the boundaries of the safe set of the continuous states, i.e. $\partial S_{\mathrm{x}} \times S_{\mathrm{q}} \times \mathcal{T}$, as during flow the discrete states and timer states cannot escape the safe set. Furthermore, due to persistent flowing and systems jumps, there is no need for the decrease during timer jumps in (4.8c).

Similarly, Corollary 4.3.1 and 4.3.2 can be relaxed:

Corollary 4.4.1 (Sufficient conditions for RSWS). *Given a closed-loop system $\mathcal{H}_{\mathrm{cl}}$, compact sets (S, I, O) that satisfy Assumption 4.4.1, and a candidate LBF V satisfying all conditions in Proposition 4.4.1, if $\exists \beta \in \mathbb{R}$ such that V additionally satisfies (4.7a), (4.7b), (4.7e) and*

$$\forall s \in O^*_{D_{\mathrm{s}}}, \forall g_{\mathrm{s}} \in G_{\mathrm{s}}(s) : V(g_{\mathrm{s}}) - V(s) \leq -\gamma_{\mathrm{d}}, \tag{4.9a}$$

$$\forall s \in O^*_{D_{\mathrm{t}}}, \forall g_{\mathrm{t}} \in G_{\mathrm{t}}(s) : V(g_{\mathrm{t}}) - V(s) \leq 0, \tag{4.9b}$$

$$\forall s \in \partial O_{\mathrm{x}} \times O_{\mathrm{q}} \times \mathcal{T} : \; V(s) > \beta, \tag{4.9c}$$

where for $Y \in \{D_{\mathrm{s}}, D_{\mathrm{t}}\}$, $O^*_Y = O^* \cap Y$, then the closed-loop system $\mathcal{H}_{\mathrm{cl}}$ satisfies (4.3).

Corollary 4.4.2 (Zeno-free maximal solutions). *Given a closed-loop system $\mathcal{H}_{\mathrm{cl}}$, a candidate LBF V satisfying all conditions in Corollary 4.4.1 w.r.t. compact sets (S, I, O) satisfying Assumption 4.4.1, if $B \cap D_{\mathrm{s}} = \emptyset$, all solutions $\phi \in \mathcal{S}_{\mathcal{H}_{\mathrm{cl}}}(I)$ are non-Zeno.*

Similar to the Lyapunov relaxations for hybrid inclusions in [57, §3.3], we can relax the LBF conditions further, if we have persistent jumping or persistent flowing. In this chapter we only consider the latter.

Assumption 4.4.2 (Restricted jumps). *All jumps cannot be followed by additional jumps, i.e. $\forall s \in S \cap D : G(s) \notin D$.*

Maximal solutions to systems that satisfy this assumption are intrinsically subjected to persistent flowing and therefore no decrease along V for every jump is required:

Corollary 4.4.3 (Sufficient LBF conditions: Persistent flow). *Given a closed-loop system $\mathcal{H}_{\mathrm{cl}}$ which satisfies Assumption 4.4.2, Theorem 4.3.1, Proposition 4.4.1 and Corollaries 4.3.1 and 4.4.1 hold with respect to $\gamma_{\mathrm{d}} = 0$.*

4.5. AUTOMATIC SYNTHESIS

In the previous sections we derived conditions on a candidate LBF to infer that the closed-loop system satisfies specification $\mathtt{CS}_1$ or $\mathtt{CS}_2$. Based on these fundamentals, we propose a framework to co-synthesize a controller κ and relaxed LBF candidate V, such that the closed-loop systems provably satisfies these specifications. The synthesis method uses genetic programming (see Chapter 3) to propose candidate controllers and LBF functions, i.e. the tuple (V, κ), which are subsequently formally verified using an SMT solver. If a candidate solution is disproved to be a solution, the SMT solver provides a counterexample which is then used to refine the candidate solutions. The use of the SMT solver is described in Section 4.5.1. The fitness within GP is detailed in Section 4.5.2. Finally, the overall algorithm outline is given in Section 4.5.3.

4.5.1. SMT solver-based verification

The verification of the desired controller specifications is reduced to verifying nonlinear inequalities on a candidate LBF, as shown in Section 4.3 and 4.4. These inequalities are verified by means of the SMT solver dReal [51], as described in Section 2.4. Recall that dReal either returns unsat or δ-sat, which are not mutually exclusive. If there is an overlap, dReal can return either case. This issue is addressed in Remark 4.5.1. In case a formula is δ-sat, dReal provides a domain in which the formula is δ-sat. From this domain we can sample states that are counterexamples where the inequality is (close to be) violated.

4.5.2. Fitness

The evolutionary search is driven by the fitness function. In this section we elaborate on how the fitness function is constructed. Based on the inequalities in Proposition 4.4.1 and Corollary 4.4.1, we employ both testing and verification techniques to assign a fitness value to a candidate solution. Given an inequality over a set, the testing is done on a finite subset of the original infinite set. This test provides us with a quality measure of candidate solutions, and thus provides a search direction for the genetic evolution. The verification method uses the SMT solver to determine a Boolean answer to whether the inequality is satisfied over the entire set.

The conditions on the LBF in Proposition 4.4.1 and Corollary 4.4.1 can be expressed as a propositional formula φ in the standard form:

$$\varphi := \forall x \in X : \left(\bigwedge\nolimits_{i=1}^{k} \left(\bigvee\nolimits_{j=1}^{l_i} f_{ij}(x) \leq 0 \right) \right), \tag{4.10}$$

where $f_{ij} : \mathbb{R}^n \to \mathbb{R}$. The standard form of the conditions in Proposition 4.4.1 and Corollary 4.4.1 can be found in Appendix B. Similar to the quantitative semantics of STL (see Section 2.3), given a formula φ in the form in (4.10), we formulate for a point $x \in X$ a satisfaction measure $\rho_\varphi : \mathbb{R}^n \to \mathbb{R}$ as:

$$\rho_\varphi(x) = \max_{i \in \{1,\dots,k\}} \left(\min_{j \in \{1,\dots,l_i\}} f_{ij}(x) \right). \tag{4.11}$$

Note that here $f_{ij}(x)$ is negative if the inequality in (4.10) is satisfied. As a result, if $\rho_\varphi(x)$ is negative, φ is true and $\rho_\varphi(x)$ is positive otherwise. Now, based on the measure ρ_φ, we construct an error metric:

$$e_\varphi(x) := \max(\rho_\varphi(x), 0), \tag{4.12}$$

which for a given point x is equal to zero if φ is true and positive if not. Based on the error metric (4.12), we construct a sample-based fitness over a finite set of samples $\hat{X} = \{x_1, \dots, x_p\} \subset X$ as:

$$\mathcal{F}_{\text{samp},\varphi} := (1 + \|[e_\varphi(x_1), \dots, e_\varphi(x_p)]\|)^{-1}. \tag{4.13}$$

By definition $\mathcal{F}_{\text{samp},\varphi} \in [0, 1]$ and is equal to 1 if for all $x \in \hat{X}$ the propositional logic formula φ is true.

Besides sample-based testing, the logic formula is formally verified by means of the SMT solver. Given the output of the SMT solver, the SMT-based fitness is defined as

$$\mathcal{F}_{\text{SMT},\varphi} = \begin{cases} 1, & \text{if } \neg\varphi \text{ is unsat,} \\ 0, & \text{if } \neg\varphi \text{ is } \delta\text{-sat.} \end{cases} \tag{4.14}$$

Finally, the full fitness of a pair (V, κ) satisfying the conditions in Proposition 4.4.1 is defined as a weighted sum of the sample-based and SMT-based fitness for each condition. The weighting is motivated by the intuition that prior to checking the conditions of the derivative and the jumps (inequalities (4.5c), (4.5d), (4.8b), and (4.8c)), V must first have the 'correct shape', i.e. satisfy the conditions with respect to the initial set and safe set (inequalities (4.5a) and (4.8a)). Therefore, the conditions are sequentially weighted with

$$w_i = \lfloor w_{i-1} \mathcal{F}_{\text{samp},\varphi_{i-1}} \rfloor, \quad i \in \{2, \dots 6\},$$

and $w_1 = 1$, where for each φ_i the corresponding inequality is shown in Appendix B in Table B.1. The final overall fitness is then defined as:

$$\mathcal{F} := \frac{1}{12} \sum_{i=1}^{6} w_i \left(\mathcal{F}_{\text{samp},\varphi_i} + \mathcal{F}_{\text{SMT},\varphi_i} \right). \tag{4.15}$$

Note that $\mathcal{F} \in [0, 1]$ and only if $\mathcal{F} = 1$, all conditions are formally proven by means of the SMT solver, hence the candidate function V is an LBF. In case it is desired to verify conditions from Corollary 4.4.1, the fitness function is extended in a similar way.

Remark 4.5.1 (Robustness w.r.t. δ-sat)**.** *As stated before in Section 2.4, δ-sat and unsat are not always mutually exclusive. If both are true, dReal can return either case. To circumvent this overlap, candidate solutions are synthesized such that they are robust w.r.t. the δ perturbation. This is done by strengthening the inequalities used in the sample-based fitness relatively to the δ perturbation. That is, for some $\epsilon \geq \delta$ and a formula expressed as (4.10), the sample-based fitness is redefined using the following strengthened formula:*

$$\varphi' := \forall x \in X, \left(\bigwedge_{i=1}^{k} \left(\bigvee_{j=1}^{l_i} f_{i,j}(x) + \epsilon \leq 0 \right) \right).$$

Secondary fitness measures

In case two or multiple individuals have the same fitness value, secondary fitness measures are used to rank individuals. The first secondary fitness value is based on the number of parameters and the second secondary fitness is based on the norm of all the parameter values. The latter promotes less complex but equivalent individuals, and the former aims to prevent parameters to blow up without improving the fitness.

4.5.3. Algorithm outline

Given a system $\mathcal{H}_{\text{cl}}$, compact sets (S, I, O) and a grammar, the algorithm undergoes the following steps:

1. A random population of (V, κ) tuples is generated adhering to the provided grammar.
2. The parameters of each individual are optimized w.r.t. the sample-based fitness using CMA-ES.
3. For all individuals with full sample-based fitness, an SMT solver is used. If there is a violation, counterexamples are generated by the SMT solver, which are added to the set employed in the sample-based fitness.

4. The overall fitness in (4.15) is computed for all individuals.
5. A new population is generated by:
   (a) Copying the best individuals of the current generation.
   (b) Selecting individuals using tournament selection and modifying them using genetic operators.
6. Steps 2 to 5 are repeated until the maximum fitness value (i.e. 1) is obtained, or a maximum number of generations is met.

4.6. Case studies

In this section we demonstrate the effectiveness of the proposed approach on several benchmark systems. Here we consider continuous-time systems, sampled-data systems, uncertain systems, switching controllers, and fully hybrid systems. All benchmarks were performed using an Intel Xeon CPU E5-1660 v3 3.00GHz using 14 parallel CPU cores. The GGGP and CMA-ES algorithms were both implemented in Mathematica 11.1.

Within the synthesis, the choice of the grammar is essential. In this work, we use a grammar covering polynomials and/or use case-specific insights to bias the grammar. Here the use of polynomials is motivated by the Weiestrass approximation theorem, stating that any continuous function on a closed interval can be approximated arbitrarily close by a polynomial. Regardless, there still might not exist a polynomial LBF [3] or it might yield a very high-order polynomial, such that the use of transcendental functions like sine functions or exponentials might be more beneficial.

4.6.1. Continuous open-loop systems

First of all, we consider fully continuous-time open-loop systems, i.e. with $D_\mathrm{s} = \emptyset$. We consider five systems, adopted from [119] and [158] and references therein, defined by the open-loop continuous dynamics $f_{\mathrm{ol,x}} : \mathbb{R}^{n_\mathrm{x}} \times \mathcal{U} \to \mathbb{R}^{n_\mathrm{x}}$, with $\mathcal{U} \subset \mathbb{R}^m$, shown in Table 4.1. These systems are: a linear system, 2nd- and 3rd-order polynomial systems, a pendulum system, and a pendulum-on-cart system.

We consider saturated control inputs, i.e. controllers of the form

$$\kappa(x) = \mathrm{sat}_{(\underline{u},\overline{u})} \circ \kappa'(x),$$
$$\mathrm{sat}_{(\underline{u},\overline{u})}(x) = \max(\underline{u}, \min(\overline{u}, x)),$$

where $\kappa' : \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}^m$ is an analytic controller to be synthesized by the proposed framework. Furthermore, we consider the system with continuous full state-feedback and with sampled-data input. In the former case the system dynamics is given by $\mathcal{H}_\mathrm{ct}$ with data

$$C = \mathbb{R}^{n_\mathrm{x}},\ F(s) = f_{\mathrm{ol,x}}(s_\mathrm{x}, \kappa \circ h(s)),$$
$$D = \emptyset,\ G(s) = \emptyset,\ h(s) = s_\mathrm{x}.$$

Given a sampling time $\eta > 0$, the effect of sampled data can be modeled by adding the sampled states as additional discrete states, resulting in the system $\mathcal{H}_\mathrm{sd}$ with

$$C = \mathbb{R}^{2n_\mathrm{x}} \times [0, \eta],\ F(s) = \begin{pmatrix} f_{\mathrm{ol,x}}(s_\mathrm{x}, \kappa \circ h(s)), & \mathbf{0}_{n_\mathrm{x}}, & 1 \end{pmatrix},$$
$$D = D_\mathrm{t} = \mathbb{R}^{2n_\mathrm{x}} \times \{\eta\},\ G(s) = (s_\mathrm{x}, s_\mathrm{x}, 0),\ h(s) = s_\mathrm{q}.$$

Table 4.1: Continuous-time systems with input $u \in \mathcal{U} = [\underline{u}, \overline{u}]$. 1: linear system. 2: 2nd-order polynomial system. 3: 3rd-order polynomial system. 4: Pendulum system. 5: Pendulum-on-cart system.

System	$f_{\mathrm{ol,x}}(x,u)$	$(S_\mathrm{x}, I_\mathrm{x}, O_\mathrm{x})$	$(\underline{u}, \overline{u})$
1	$\begin{pmatrix} x_2 \\ -x_1 + u \end{pmatrix}$	$([-1,\ 1]^2,$ $[-0.5, 0.5]^2,$ $[-0.1, 0.1]^2)$	(-1,1)
2	$\begin{pmatrix} x_2 - x_1^3 \\ u \end{pmatrix}$	$([-1,\ 1]^2,$ $[-0.5, 0.5]^2,$ $[-0.05,\ 0.05]^2)$	(-1,1)
3	$\begin{pmatrix} -10x_1 + 10x_2 + u \\ 28x_1 - x_2 - x_1x_3 \\ x_1x_2 - 2.6667x_3 \end{pmatrix}$	$([-5,\ 5]^3,$ $[-1.2, 1.2]^3,$ $[-0.3, 0.3]^3)$	(-100,100)
4	$\begin{pmatrix} x_2 \\ \frac{mlg}{J}\sin(x_1) - \left(\frac{b}{J} + \frac{K^2}{JR_a}\right)x_2 + \frac{K}{JR_a}u \end{pmatrix}$ $m = 5.50 \cdot 10^{-2}$ kg, $l = 4.20 \cdot 10^{-2}$ m, $J = 1.91 \cdot 10^{-4}$ kg m^2, $g = 9.81$ m/s^2, $K = 5.36 \cdot 10^{-2}$ Nm/A, $R_a = 9.50\Omega$. $b = 3.0 \cdot 10^{-6}$Nms	$([-2\pi,\ 2\pi]$ $\times [-100,\ 100]\,,$ $[-\pi,\ \pi]$ $\times [-10,\ 10],$ $[-1.0,\ -0.5]$ $\times [-1.0,\ 1.0])$	(-10,10)
5	$\begin{pmatrix} x_2 \\ \frac{g}{l}\sin(x_1) - \frac{b}{ml^2}x_2 + \frac{1}{ml}\cos(x_1)u \end{pmatrix}$ $g = 9.8$ m/s^2, $b = 2$ Nms $l = 0.5$ m, $m = 0.5$ kg.	$([-2\pi, 2\pi]$ $\times [-10, 10],$ $[-0.5, 0.5]^2,$ $[-0.25, 0.25]^2$	(-6,6)

Note that here $h(s)$ is dependent on the discrete states s_q. Given these models $\mathcal{H}_\mathrm{ct}$ and $\mathcal{H}_\mathrm{sd}$, we synthesize controllers κ' and LBFs V for specification $\mathtt{CS}_1$ with (S, I, O) as $(S_\mathrm{x}, I_\mathrm{x}, O_\mathrm{x})$ for $\mathcal{H}_\mathrm{ct}$ and as $(S_\mathrm{x}^2 \times \mathcal{T}, \{(s_\mathrm{x}, s_\mathrm{q}, s_\mathrm{t}) \in I_\mathrm{x}^2 \times \{0\} \mid s_\mathrm{q} = s_\mathrm{x}\}, O_\mathrm{x}^2 \times \mathcal{T})$ for $\mathcal{H}_\mathrm{sd}$, where $(S_\mathrm{x}, I_\mathrm{x}, O_\mathrm{x})$ are defined for each system in Table 4.1, and η as shown in Table 4.2.

As a baseline of the proposed framework, we synthesize controllers and LBFs based on parameterized candidate solutions with fixed structures. Since the structure is fixed, no genetic operators are applied. This is a special case of the full framework, where the grammar specifies a single full candidate template. For these parameterized solutions, we consider for models $\mathcal{H}_\mathrm{ct}$ templates of the form:

$$\begin{aligned} V(s) &= x^T A_1 x + c, \\ \kappa'(z) &= Kx, \\ x &= s_\mathrm{x} - x_O, \end{aligned}$$

and for model $\mathcal{H}_\mathrm{sd}$:

$$\begin{aligned} V(s) &= x^T A_1 x + (\eta - s_\mathrm{t})(x-z)^T A_2 (x-z) + c, \\ \kappa'(s) &= Kz, \\ x &= s_\mathrm{x} - x_O, \\ z &= s_\mathrm{q} - x_O, \end{aligned}$$

Table 4.2: Results across 10 runs for continuous-time systems with continuous controllers, using a fixed template. μ: mean, σ: standard deviation.

System	number of generations				time [s]			
	min	max	μ	σ	min	max	μ	σ
1	1	1	1.0	0.00	3.44	3.87	3.61	0.14
2	1	3	2.1	0.57	3.44	11.30	7.85	2.21
3	2	4	2.7	0.82	8.10	23.00	13.45	5.54
4	4	9	7.0	1.76	15.90	47.29	33.63	10.95
5	2	5	2.9	0.99	7.60	23.29	12.32	5.00

Table 4.3: Results across 10 runs for continuous-time systems with sampled-data controllers, using a fixed template. μ: mean, σ: standard deviation. [1] SMT time-out, [2] No convergence.

System	η	number of generations				time [s]			
		min	max	μ	σ	min	max	μ	σ
1	0.01	1	7	2.7	1.83	14.49	126.28	49.82	36.00
2	0.01	2	6	4.1	1.37	30.33	179.11	107.35	53.13
3	0.001	$-^1$	-	-	-	-	-	-	-
4	0.001	$-^2$	-	-	-	-	-	-	-
5	0.01	3	16	8.6	3.66	36.80	576.35	178.98	153.87

where A_1, A_2 are upper-triangular matrices, c a constant, and x_O the center of O_{x}. We consider 14 individuals and start with 100 test samples and a maximum of 300 counterexamples, where a first-in-first-out principle is used. We use per iteration 30 CMA-ES generations and we set the maximum number of iterations to 200. The results are shown in Tables 4.2 and 4.3. Here we observe that for model $\mathcal{H}_{\mathrm{sd}}$ of system 3 the computation time of the SMT solver surpassed the user-imposed time-out limit of 20 seconds for all individuals in a generation. In this case no counterexamples are generated, nor an answer is provided whether an individual is a solution, hence the algorithm is terminated. For model $\mathcal{H}_{\mathrm{sd}}$ of system 4 and the given template, we observe that no solutions are found within the maximum number of iterations. Note that this is no guarantee that no solution exists within this solution structure.

Let us consider the solutions for model $\mathcal{H}_{\mathrm{ct}}$ of system 5. Using a line search over β and checking the inequalities in Corollary 4.3.1 using an SMT solver, we found that for 9 out of 10 solutions we could find a β such that Corollary 4.3.1 holds, i.e. the closed-loop system also satisfies $\mathtt{CS}_2$. An example of a solution that also satisfies $\mathtt{CS}_2$ is given by

$$V(s) = -14.4983 + 23.06s_1^2 + 11.6469s_1s_2 + 17.9399s_2^2,$$
$$\kappa(s) = -11.0776s_1 - 9.32858s_2,$$

with $\beta = -14.0381$. The sets S, I, O, A, and B are shown in Figure 4.5. Since Corollary 4.3.1 holds, A is a forward invariant set which is found automatically using the proposed framework. Moreover, note that given the found solution, we cannot trivially increase

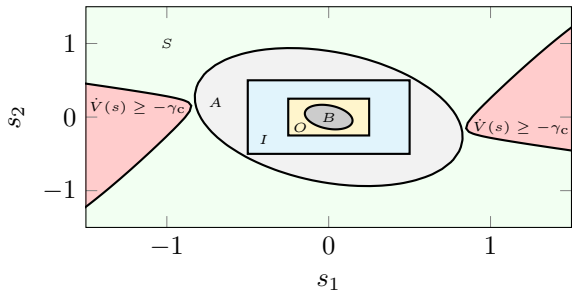
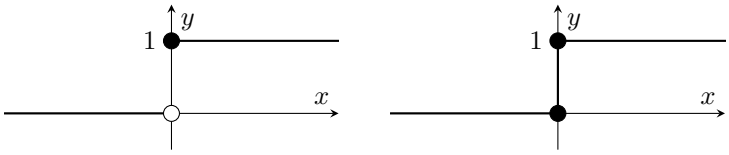


Figure 4.5: Specification sets (S, I, O) and the sublevel sets A and B of a found result for system 5 with continuous-time controller. The red areas indicate where the derivative $\dot{V}(s) = \langle V(s), F(s) \rangle$ is above γ_c.



(a) $y = \sigma(x)$. (b) $y = \hat{\sigma}(x)$.

Figure 4.6: Switching function $\sigma(x)$ and its outer semicontinuous variant $\hat{\sigma}(x)$.

the size of this forward invariant set A, e.g. by shifting V, as we can observe that for some neighboring states of A we have $\langle \nabla V(s), f(s) \rangle > -\gamma_c$, which would then violate condition (4.5d).

4.6.2. Bounded uncertainties

Let us consider a continuous-time system described by $\dot{x}(t) = f(x, d)$, where $d \in \Delta$ is a bounded disturbance and Δ is compact. This system can be modeled in the framework by writing the dynamics as the following set-valued function:

$$F(s) = \{ f(s, d) \in \mathbb{R}^n \mid d \in \Delta \}.$$

Let us reconsider model $\mathcal{H}_{\text{ct}}$ of system 5 (pendulum-on-cart) from Table 4.1 and adapt $F(s)$ to

$$F(s) = \begin{pmatrix} s_2 \\ \left\{ \frac{g}{l} \sin(s_1) - \frac{b s_2}{m l^2} + \frac{1}{ml} \cos(s_1) \kappa(s) + d \mid d \in \Delta \right\} \end{pmatrix},$$

with $\Delta = [-0.5, 0.5]$. Using the same solution template as before, for 10 runs, synthesis took on average 3.3 generations and 24.22 seconds.

4.6.3. Switching controllers

Using the proposed framework, it is possible to consider switching controllers. Let us consider the DC-DC boost converter system from [55], modeled as a switched system

Table 4.4: Production rules $\mathcal{P}$.

$\mathcal{N}$	Rules
$\langle \text{pol} \rangle$	$::= \langle \text{pol} \rangle + \langle \text{pol} \rangle \mid \langle \text{const} \rangle \times \langle \text{mon} \rangle$
$\langle \text{mon} \rangle$	$::= \langle \text{var} \rangle \mid \langle \text{var} \rangle \times \langle \text{mon} \rangle$
$\langle \text{var} \rangle$	$::= s_1 \mid s_2$
$\langle \text{const} \rangle$	$::=$ Random Real $\in [-10, 10]$

with $q \in \{0, 1\}$:

$$f_q(s) = A_q s + b, \quad b = (v_s, 0)^T$$
$$A_0 = \begin{pmatrix} -\frac{r_l}{x_l} & 0 \\ 0 & -\frac{1}{x_c}\frac{1}{r_0+r_c} \end{pmatrix},$$
$$A_1 = \begin{pmatrix} -\frac{1}{x_l}\left(r_l + \frac{r_0 r_c}{r_0+r_c}\right) & -\frac{1}{\alpha x_l}\frac{r_0}{r_0+r_c} \\ \frac{\alpha}{x_c}\frac{r_0}{r_0+r_c} & -\frac{1}{x_c}\frac{1}{r_0+r_c} \end{pmatrix}.$$

The parameters of the model are as taken in [55]. In order to control the system, one has to design the switching signal q. We approach this by designing a state-dependent control law with the following structure:

$$q(s) = \sigma(\kappa(s)),$$
$$\sigma(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases}$$

Rewriting this system as a hybrid system (as in definition 2.2.3), we have $\mathcal{H} = (C, F, \emptyset, \emptyset)$ with:

$$F(s) = \{A'(s)q + b'(s) \mid q \in \hat{\sigma}(\kappa(s))\},$$
$$A'(s) = \begin{pmatrix} -\frac{s_1}{x_l}\frac{r_0 r_c}{r_0+r_c} - \frac{s_2}{x_l}\frac{r_0}{r_0+r_c} \\ \frac{s_1}{x_c}\frac{r_0}{r_0+r_c} \end{pmatrix}, b'(s) = \begin{pmatrix} -\frac{s_1 r_l}{x_l} + v_s \\ -\frac{s_2}{x_c}\frac{1}{r_0+r_c} \end{pmatrix},$$

where $\hat{\sigma}$ is an outer semicontinuous over-approximation of σ defined as

$$\hat{\sigma}(x) = \begin{cases} 1 & \text{if } x > 0, \\ [0, 1] & \text{if } x = 0, \\ 0 & \text{if } x < 0. \end{cases} \tag{4.16}$$

Note that this system satisfies the hybrid basic conditions in assumption 2.2.1. The difference between σ and $\hat{\sigma}$ is illustrated in Figure 4.6. Note that this hybrid model includes the original dynamics, i.e. if $\kappa(s) > 0$, $F(s) = f_1(s)$, if $\kappa(s) < 0$, $F(s) = f_2(s)$, and if $\kappa(s) = 0$, $\{f_0(s), f_1(s)\} \subset F(s)$.

We synthesize a controller κ for specification $\texttt{CS}_1$ with the safe, initial and goal set as in [121], i.e. $S = [0.65, 1.65] \times [4.95, 5.95]$, $I = [0.85, 0.95] \times [5.15, 5.25]$, $O = [1.25, 1.45] \times [5.55, 5.75]$. Given that the initial and goal sets are relatively close to the safe

set, a second-order polynomial is likely not to suffice, and therefore we bias our solutions by including a pre-specified barrier function of the form:

$$B(c,s) = \frac{c_1}{1.66 - s_1} + \frac{c_2}{5.96 - s_2} + \frac{c_3}{s_1 - 0.64} + \frac{c_4}{s_2 - 4.94}.$$

Using this barrier function, we employ the start tree of the candidate LBF $\mathcal{S}_V$ given by the sum of a constant $\langle \text{const} \rangle$, polynomial $\langle \text{pol} \rangle$ and the barrier function $B(c,s)$:

$$\begin{aligned} \mathcal{S}_V &= \langle \text{const} \rangle + \langle \text{pol} \rangle + \langle \text{const} \rangle \, B(c,s), \\ c &= \left(\langle \text{const} \rangle, \quad \langle \text{const} \rangle, \quad \langle \text{const} \rangle, \quad \langle \text{const} \rangle \right). \end{aligned}$$

Furthermore, taking inspiration from synthesis of switching controllers based on a CLFB (see e.g. [121], [159]), the controller is based on the candidate LBF V, such that

$$\begin{aligned} q = \hat{\sigma}(\kappa(s)) = 1 \text{ if } \langle \nabla V(s), f_0(s) \rangle > \langle \nabla V(s), f_1(s) \rangle, \\ q = \hat{\sigma}(\kappa(s)) = 0 \text{ if } \langle \nabla V(s), f_0(s) \rangle < \langle \nabla V(s), f_1(s) \rangle. \end{aligned}$$

In other words, a mode q is selected so that it minimizes $\langle \nabla V(s), f_q(s) \rangle$. This is achieved by the following controller:

$$\kappa(s) = \langle \nabla V(s), f_0(s) \rangle - \langle \nabla V(s), f_1(s) \rangle). \tag{4.17}$$

Based on this prior knowledge, we use the start tree $\text{Tuple}(\mathcal{S}_V, \kappa(s))$ and the production rules in Table 4.4. We used 8 individuals, a maximum tree depth of 10, a mutation chance of 0.8, crossover chance of 0.3, 30 generations in CMA-ES, 100 test samples and a maximum of 300 counterexamples. In 10 different runs with a maximum of 200 generations, we found in 3 runs a solution in the 104th, 115th, and 151st generation with on average 20 seconds per generation. An example of a found solution is given by:

$$\begin{aligned} V(s) =& 1.66125 B(c,s) - 2.96592 s_1^3 - 5.36934 s_1^2 s_2^2 \\ & - 26.175 s_1^2 - 5.55243 s_1 s_2^2 + 26.763 s_1 s_2 \\ & - 17.0781 s_1 - 0.0612397 s_2^3 - 10.2641 s_2^2 \\ & + 48.5132 s_2 + 32.6963, \\ c =& \left(28.2706, \quad 16.4118, \quad 2.64323, \quad 3.967 \right), \end{aligned}$$

and the corresponding controller given by (4.17). Given this solution, the set A and the controller regions are shown in Figure 4.7. While the controller is synthesized for states that start in I, specification $\texttt{CS}_1$ holds for all states starting in A.

4.6.4. Discovering controller structures

In this section we illustrate how our method can be used to automatically find an appropriate controller structure. Here we consider the nonholonomic integrator with $\mathcal{H} = (\mathbb{R}^3, F_{\text{ol}}(s, \kappa(s)), \emptyset, \emptyset)$ and

$$F_{\text{ol}}(s,u) = \left(u_1, \quad u_2, \quad s_1 u_2 - s_2 u_1 \right),$$

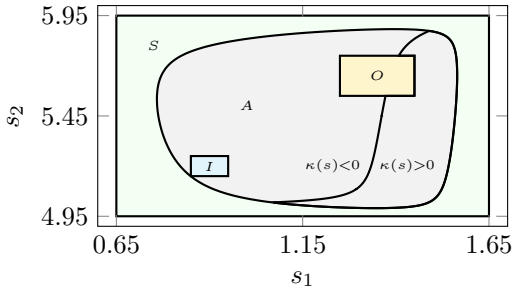


Figure 4.7: Specification set (S, I, O) and the level sets for a found LBF for the DC-DC boost converter system

which does not satisfy Brockett's necessary condition [24, 156]. Therefore, while this system is controllable, there exists no continuous-time state-feedback law to asymptotically stabilize the system. However, note that this does not automatically imply that there does not exist a continuous state-feedback law which satisfies the specifications $\texttt{CS}_1$ and $\texttt{CS}_2$ for a given (S, I, O). Moreover, we consider a saturated input $u_i = \mathrm{sat}_{(-1,1)} \circ \kappa_i(s)$ for $i \in \{1, 2\}$ and a safe set $S = [-5, 5]^3$, initial set $I = [-3, 3]^2 \times [-0.1, 0.1]$ and goal set $O = [-0.5, 0.5]^3$. That is, it is desired to steer the system to a neighborhood around the origin, where initially x_3 is close to zero.

For simplicity, we consider a parameterized quadratic LBF and for the controller a grammar containing multiple controller classes, namely linear, polynomial, and discontinuous controllers. The start symbol is given by $\mathcal{S} = \mathrm{Tuple}\left(\langle \mathrm{V} \rangle, \left(\langle \kappa_\mathrm{i} \rangle, \langle \kappa_\mathrm{i} \rangle\right)\right)$, with

$$\langle \mathrm{V} \rangle ::= \langle \mathrm{const} \rangle + \langle \mathrm{const} \rangle\, s_1^2 + \langle \mathrm{const} \rangle\, s_2^2 + \langle \mathrm{const} \rangle\, s_3^2,$$

and the (other) production rules are given in Table 4.5. In the grammar, $\langle \mathrm{disc} \rangle$ is the nonterminal for discontinuous expressions and sign denotes the outer semicontinuous sign function, defined as $\mathrm{sign}(x) := 2\hat{\sigma}(x) - 1$, where $\hat{\sigma}$ is defined in (4.16). Finally, the discontinuities are limited to $\mathrm{sign}(s_3)$, to limit the search space and because it is a repeating element in the controllers found in [156]. We used 28 individuals, a maximum tree depth of 4, a mutation chance of 0.8, crossover chance of 0.3, a maximum of 200 generations, 30 generations in CMA-ES, 100 test samples and a maximum of 300 counterexamples.

Out of 10 independent runs, the algorithm found in 7 runs a solution within 200 generations. On average, these 7 runs took 19.76 minutes and 110 generations. Of these 7, 6 controllers contained a discrete element in both inputs, 1 controller was fully polynomial, and no linear controllers were found. The polynomial controller is given by

$$\begin{aligned} V(s) &= -5.3754 + 0.3457 s_1^2 + 0.2184 s_2^2 + 21.6876 s_3^2, \\ \kappa(s) &= \begin{pmatrix} -0.523878 s_1 + 1.47349 s_2 s_3 \\ -0.169653 s_2 - 5.76889 s_1 s_3 + 1.16537 s_3^2 \end{pmatrix}. \end{aligned}$$

Therefore, despite the system not meeting Brockett's necessary condition, for this specification, the algorithm was able to automatically find a sufficient continuous control law, whereas no linear controller was found.

Table 4.5: Production rules $\mathcal{P}$

$\mathcal{N}$	Rules
$\langle \kappa_{\mathrm{i}} \rangle$	$::= \langle \mathrm{lin} \rangle \mid \langle \mathrm{pol} \rangle \mid \langle \mathrm{pol} \rangle + \langle \mathrm{const} \rangle \langle \mathrm{disc} \rangle$
$\langle \mathrm{lin} \rangle$	$::= \langle \mathrm{const} \rangle\, s_1 + \langle \mathrm{const} \rangle\, s_2 + \langle \mathrm{const} \rangle\, s_3$
$\langle \mathrm{pol} \rangle$	$::= \langle \mathrm{pol} \rangle + \langle \mathrm{pol} \rangle \mid \langle \mathrm{const} \rangle \times \langle \mathrm{mon} \rangle$
$\langle \mathrm{disc} \rangle$	$::= \mathrm{sign}(s_3) \mid \langle \mathrm{pol} \rangle\, \mathrm{sign}(s_3) \mid$
$\langle \mathrm{mon} \rangle$	$::= \langle \mathrm{var} \rangle \mid \langle \mathrm{var} \rangle \times \langle \mathrm{mon} \rangle$
$\langle \mathrm{var} \rangle$	$::= s_1 \mid \ldots \mid s_3$
$\langle \mathrm{const} \rangle$	$::= \text{Random Real} \in [-10, 10]$

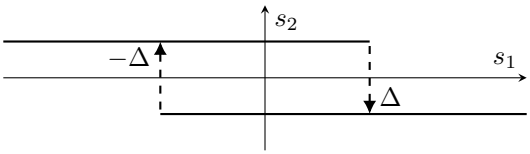


Figure 4.8: Hysteresis.

4.6.5. Jump-flow systems

Let us consider a system with $D_{\mathrm{s}} \neq \emptyset$, namely a hysteresis system adopted from [23], graphically illustrated in Figure 4.8. This system can be modeled as a hybrid automaton, as shown in [23]. Using the jump-flow formalism, the system states $s = (s_{\mathrm{x}}, s_{\mathrm{q}}) \in \mathbb{R} \times \{-1, 1\}$ consist of a single continuous state $s_{\mathrm{x}} \in \mathbb{R}$ and a discrete state, which models the state of the hysteresis $s_{\mathrm{q}} \in \{-1, 1\}$. The data of this system is given by:

$$
\begin{aligned}
F_{\mathrm{ol}}(s, u) &= \big(s_{\mathrm{q}} + u, 0\big)\,, \\
G_{\mathrm{ol}}(s, u) &= \begin{cases} (s_{\mathrm{x}}, -1) & \text{if } s \in D_1, \\ (s_{\mathrm{x}}, 1) & \text{if } s \in D_2, \end{cases} \\
H_1 = \{s_{\mathrm{x}} \in \mathbb{R} \mid s_{\mathrm{x}} \geq \Delta\}, &\; H_2 = \{s_{\mathrm{x}} \in \mathbb{R} \mid s_{\mathrm{x}} \leq -\Delta\}, \\
C &= [-\Delta, \Delta] \times \{-1, 1\} \cup H_1 \times \{-1\} \cup H_2 \times \{1\}, \\
D_1 = H_1 \times \{1\},\; D_2 &= H_2 \times \{-1\},\; D_{\mathrm{s}} = D_1 \cup D_2, \\
h(s) &= s_{\mathrm{x}}.
\end{aligned}
$$

Setting $\Delta = 1$, we consider the safe, initial and goal set as $(S, I, O) = ([-5, 5] \times \{-1, 1\}, [-2, 2] \times \{-1, 1\}, [-1, 1] \times \{-0.5, 0.5\})$. Using the solution template

$$
\begin{aligned}
V(s) &= sA_1 s + c, \\
\kappa(s) &= cs_1,
\end{aligned}
$$

where A_1 is an upper-triangular matrix and c a constant, and using the same settings as before, we synthesized solutions across 10 runs in 2.4 generations and 5 seconds. An

example of a solution is given by

$$\begin{aligned} V(s) = &- 228.165 + 25.0271 s_1^2 \\ &+ 0.189837 s_1 s_2 + 84.7784 s_2^2, \\ \kappa(s) = &- 11.7482 s_1. \end{aligned}$$

4.6.6. Design of flow and jump maps

Finally, we demonstrate that the approach can also be used to design the flow and jump sets C and D. We revisit the DC-DC boost converter from Section 4.6.3. Instead of designing a switching signal, we augment the state space with a logic state and design a map $\kappa : \mathbb{R}^2 \to \mathbb{R}$ that partitions the state space. The closed-loop system is given by the hybrid data (C, F, D, G):

$$\begin{aligned} F(s) = \begin{pmatrix} A(s_{\mathrm{x}})s_{\mathrm{q}} + b(s_{\mathrm{x}}) \\ 0 \end{pmatrix}, \; G(s) = \begin{pmatrix} s_{\mathrm{x}} \\ 1 - s_{\mathrm{q}} \end{pmatrix}, \\ C = \{(x,0) \in S \mid \kappa(x) \leq \varepsilon\} \cap \{(x,1) \in S \mid \kappa(x) \geq 0\}, \\ D = \{(x,0) \in S \mid \kappa(x) \geq \varepsilon\} \cap \{(x,1) \in S \mid \kappa(x) \leq 0\}, \end{aligned}$$

where $x \in \mathbb{R}^{n_{\mathrm{x}}}$ and $\varepsilon > 0$. Note that infinite switching between the modes is prevented by design by a hysteresis parameterized by $\varepsilon > 0$. Moreover, as $C \cap D \neq \emptyset$, solutions are not unique, but regardless, the synthesis guarantees that all maximal solutions satisfy the specification. We use again the same expert insight as in Section 4.6.3 and set $\kappa(x)$ to be equal to the controller structure in (4.17). We find that for $\varepsilon = 0.001$, the previously found solution in Section 4.6.3 is again an LBF.

4.6.7. Discussion

We proceed now to discuss the results of the case studies and compare them to results in the literature. The average computation time in the results of continuous-time systems in Table 4.2 suggests that the computational time increases as the systems become more nonlinear. However, general conclusions on the computation time are speculative. That is, besides that the method is not guaranteed to find a solution within a finite number of generations, the computation time can highly vary depending on factors including the system dynamics, the system order, provided expert-knowledge in the form of a grammar, and the genetic programming parameters.

Comparing the method with SCOTS and ROCS for the inverted pendulum system (system 5 in Table 4.1), we obtained a controller in the form of a simple expression, whereas according to [94], with a state grid size of 0.001 the abstraction used in SCOTS took more than 12 hours and did not return a result, and ROCS generated a controller in 400 seconds with a controller consisting of 26340 partitions. Using the proposed methodology, synthesis took on average 178.98 seconds and the controller was given by a single analytic expression. Another advantage of the proposed method over these abstraction-based approaches is that it does not directly depend on discretization of the state and input space, resulting in potentially better scalability, as the proposed method requires less memory. Finally, the specification is guaranteed for continuous-time trajectories, rather than for

discrete-time trajectories instances, which is the case for these abstraction-based methods.

Comparing our method with the counterexample-guided synthesis methods presented in [118, 121], the method presented in this chapter is overall slower, but is able to discover the solution structures itself, whereas e.g. for the DC-DC boost converter the authors [121] had to iteratively add barrier functions by hand before a solution could be found. Moreover, in our benchmarks we provided sampled-data controllers with a larger sampling time than the minimum dwell-times presented in [118, 121], and the presented framework is able to cope with hybrid systems described with differential and difference inclusions, whereas the methods in [118, 121] are restricted to nonlinear switched systems.

The proposed method is not complete, i.e. solutions may not be found even if they exist. This may stem from a not sufficiently expressive choice of grammar, or due to the used optimizers (GGGP and CMA-ES), which do not guarantee finding a solution within a fixed number of generations. Therefore, the method is best used in combination with expert-knowledge, incorporated in the grammar, which biases the search to viable candidate solutions. Nonetheless, the required expert-knowledge is less than the one required when employing e.g. sum of squares programming or counterexample-guided synthesis approaches, where the user has to provide a solution structure.

The computation time of the framework can be improved in several instances. First of all, GGGP and CMA-ES are implemented in Mathematica. Implementation in lower-level languages could speed up the computation time. Additionally, more efficient implementations exploiting paralellization, e.g. by using GPU-based computation and more advanced GP variants should improve speed and scalability.

Finally, using the proposed framework, we did not find sampled-data controllers for all systems in Table 4.1 (systems 3 and 4). In the next section we present a specialized approach for sampled-data nonlinear systems which is able synthesize controllers for these systems.

4.7. Specialized synthesis for sampled-data systems

The framework introduced in Section 4.5 is for very general hybrid systems. In this section we explore a specialized approach for sampled-data systems. Using this specialized approach, controllers are synthesized for cases in which in Section 4.6.1 no sampled-data controller was found. Additionally, in this section we utilize a control Lyapunov barrier function (CLBF), rather than an LBF. Where an LBF is used to verify reach-avoid problems for an autonomous system, a CLBF can be used to derive a control input such that the specification is guaranteed. Specifically, we consider a set of controller modes, either pre-defined or co-evolved with the CLBF, and use a periodic switching law based on the CLBF.

This approach presented in this section is a follow-up to [158], in which also a combination of GP and SMT solvers is used. The main contributions of this section w.r.t. [158] are: 1) synthesis w.r.t. a pre-defined periodic sampling time, rather than arbitrary switching with a (more conservative) minimum dwell-time and 2) the use of a different and less conservative CLBF.

4.7.1. Problem definition

Let us consider nonlinear continuous-time systems described by

$$\dot{\xi}(t) = f(\xi(t), u(t)), \tag{4.18}$$

where the variables $\xi(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ and $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ denote the state and input respectively. We assume that the dynamics satisfy the following assumption:

Assumption 4.7.1. *The derivative* $\frac{\partial f(x,u)}{\partial x}$ *exists for all* $x \in \mathcal{X}$.

In this section we design sampled-data state-feedback controllers $\kappa : \mathbb{R}^n \to \mathcal{U}$, such that $u(t) = \kappa(\xi(t_k)), \forall t \in [t_k, t_k+\eta)$, where $\eta > 0$ denotes a constant sampling time. This yields a closed-loop system Σ^{sd} described by the data $(\mathcal{X}, f, \kappa, \eta)$ (see Definition 2.2.7). We address the same problem as described in Section 4.2, here simplified for the subclass of continuous-time systems with sampled-data controllers. That is, given a compact safe set $S \subseteq \mathcal{X}$, compact initial set $I \subset S$ and compact goal set $O \subset S$, we consider the following specifications:

- $\texttt{CS}_1$ *Reach while stay (RWS)*: all solutions ξ to Σ^{sd} starting from the initial set I eventually reach the goal set O, while staying within the safe set S:
$$\forall \xi \in \mathcal{S}_{\Sigma^{\text{sd}}}(I), \exists T, \forall t \in [0, T] : \xi(t) \in S \wedge \xi(T) \in O. \tag{4.19}$$
- $\texttt{CS}_2$ *Reach and stay while stay (RSWS)*: all solutions ξ to Σ_{cl} starting from the initial set I eventually reach and stay in the goal set O, while always staying within the safe set S:
$$\forall \xi \in \mathcal{S}_{\Sigma^{\text{sd}}}(I), \exists T, \forall t \geq 0, \forall \tau \geq T : \xi(t) \in S \wedge \xi(\tau) \in O. \tag{4.20}$$

We address the following problem:

Problem 4.7.1

Given the compact sets (S, I, O) and system (4.18), synthesize a controller $\kappa : \mathbb{R}^n \to \mathcal{U}$ such that the closed-loop system $\Sigma^{\text{sd}} = (\mathcal{X}, f, \kappa, \eta)$ satisfies specification $\texttt{CS}_1$ or $\texttt{CS}_2$.

4.7.2. Control strategy

In this section we discuss the used control strategy and establish how it solves problem 4.7.1 by means of Theorem 4.7.1 and Corollary 4.7.1, which can be seen as specialized variants of Theorem 4.3.1 and Corollary 4.3.1, respectively. The main difference here is that we consider a CLBF, rather than an LBF. The proofs of the theorem and corollaries in this section can again be found in Appendix A.

Consider a set of controller modes with index set $Q \subset \mathbb{Z}_{\geq 0}$:

$$\mathcal{G} = \{g_q : \mathcal{X} \to \mathcal{U} \mid q \in Q\}. \tag{4.21}$$

Given the system (4.18), an initial state $x = \xi(t_k)$, let us denote the (over-approximated) reachable set for $t \in [t_k, t_k + \eta]$ under a controller mode q as $R_q(x)$ s.t. given a q, $\forall t \in [t_k, t_k + \eta] : \xi(t) \in R_q(\xi(t_k))$. In this section we consider a switching controller based on a CLBF, which is defined as follows:

Definition 4.7.1 (Control Lyapunov Barrier Function). *A function $V \in \mathcal{C}^1(S, \mathbb{R})$ is a Control Lyapunov Barrier Function w.r.t. the compact sets (S, I, O), $S \subseteq \mathcal{X}$, $I, O \subseteq int(S)$, system (4.18), and controller modes (4.21) if there exists a scalar $\gamma > 0$ such that*

$$\forall x \in I : V(x) \leq 0, \tag{4.22a}$$

$$\forall x \in \partial S : V(x) > 0, \tag{4.22b}$$

$$\forall x \in A \backslash O, \exists q \in Q, \forall z \in R_q(x) : \dot{V}_q(x, z) \leq -\gamma, \tag{4.22c}$$

where $A := \{x \in S \mid V(x) \leq 0\}$ and $\dot{V}_q(x, z) = \langle \nabla V(z), f(z, g_q(x)) \rangle$.

Remark 4.7.1. *Similar to Remark 4.3.2, the choice of γ is arbitrary, because if a solution V^* exists for γ^*, there always exists a linear transformation of V^* such that the inequalities in (4.22) are satisfied for any γ.*

4

Given a CLBF V, we consider periodically switching controllers of the form such that for all $t \in [t_k, t_k + \eta)$

$$\begin{cases} u(t) & = \kappa(\xi(t_k)), \\ \kappa(x) & = g_{q_k}(x), \\ q_k & = \arg\min\limits_{q \in Q} \max\limits_{z \in R_q(x)} \dot{V}_q(x, z). \end{cases} \tag{4.23}$$

This controller strategy based on the CLBF enforces specification $\texttt{CS}_1$, as shown in the following theorem.

Theorem 4.7.1 (Reach while stay). *Given a system (4.18), CLBF V w.r.t. compact sets (S, I, O) and controller (4.23), then (4.19) holds.*

Similarly as the result in Theorem 4.3.1 in Section 4.3, these conditions are not sufficient for forward invariance of (a subset of) the goal set, hence trajectories might leave the goal set after entering it. The following corollary establishes sufficient conditions for specification $\texttt{CS}_2$.

Corollary 4.7.1 (Reach and stay while stay). *Given a system (4.18), CLBF V w.r.t. compact sets (S, I, O), and a controller (4.23), if $\exists \beta \in \mathbb{R}$ such that*

$$\forall x \in \partial O : \; V(x) > \beta, \tag{4.24a}$$

$$\forall x \in O \backslash int(B), \exists q \in Q, \forall z \in R_q(x) : \dot{V}_q(x, z) \leq -\gamma, \tag{4.24b}$$

where $B := \{x \in S \mid V(x) \leq \beta\}$, then (4.20) holds.

4.7.3. ONE-STEP AHEAD REACHABLE SET

In this chapter the reachable set is constructed by using Euler's forward method and bounding the local truncation error (LTE). This yields the following analytic expression

$$r_q(x, \tau, e) = x + \tau f(x, g_q(x)) + \frac{1}{2}\tau^2 e,$$

such that the over-approximated reachable set is given by

$$R_q(s) = \bigcup_{(\tau,e)\in E} r_q(s,\tau,e) \tag{4.25}$$

with $E := [0,\eta] \times \Pi_{i=1}^{n}[-\varepsilon_i, \varepsilon_i]$ and

$$\varepsilon_i = \max_{(x,u)\in\mathcal{X}\times\mathcal{U}} \left| \frac{\partial f_i(x,u)}{\partial x} f_i(x,u) \right|. \tag{4.26}$$

While this construction can be quite conservative, it allows for relatively simple analytic expressions.

4

4.7.4. Automatic synthesis

To solve problem 4.7.1, we synthesize the pair $(V, \mathcal{G})$ similarly as the pair (V, κ) as described in Section 4.5. The fitness function, based on the inequalities in Theorem 4.7.1 and Corollary 4.7.1, is constructed as described in Section 4.5.2. To this end, these inequalities are rewritten to the standard form in 4.10, and are shown in Appendix B.

Additional operations

To aid in finding the correct shift of $V(x)$ such that (4.22a) is satisfied, the following biasing is performed before each fitness evaluation within CMA-ES:

$$V'(x) = V(x) - \max\Big(\max_{x\in I_{\text{samp}}} (V(x)), 0\Big), \tag{4.27}$$

where I_{samp} denotes a subsampled set of I. To guide the search further, we impose the additional condition

$$\forall x \in S \backslash O : V(x) \geq V(x_c), \tag{4.28}$$

where x_c denotes the center of the goal set.

4.7.5. Implementation

The switching law in (4.23) is computationally intensive to check online. By offline designing $\alpha_q : \mathbb{R}^n \to \mathbb{R}$ for all $q \in Q$ such that

$$\begin{aligned} &\forall x \in D, \forall q \in Q : \max_{z\in R_q(x)} \dot{V}_q(x,z) > -\gamma \implies \\ &\min_{p\in Q}(\dot{V}_p(x,x) + \alpha_p(x)) < \dot{V}_q(x,x) + \alpha_q(x), \end{aligned} \tag{4.29}$$

allows us to replace the switching law with:

$$q_k(t_k) = \arg\min_{q\in Q}(\dot{V}_q(\xi(t_k), \xi(t_k)) + \alpha_q(\xi(t_k))). \tag{4.30}$$

Intuitively, when at a point x a mode q' is not viable under the reachable set $R_q(x)$, the nominal system $\dot{V}_{q'}(x,x)$ plus buffer $\alpha_{q'}(x)$ should not minimize the set $\bigcup_{q\in Q} \dot{V}_q(x,x) + \alpha_q(x)$, such that it is not selected by the switching law. The functions $\alpha_q(x)$ can be designed and verified offline using again an SMT solver.

Table 4.6: Controller modes $\mathcal{G}$ and bounded ε corresponding to the systems in Table 4.1.

System	$\mathcal{G}$	ε
1	$\{-1, 0, 1\}$	$(2, 1)$
2	$\{-1, 0, 1\}$	$(7, 0)$
3	$\{-100, -50, -5, 0, 5, 50, 100\}$	$(3800, 6800, 1900)$
4	$\{-10, -5, 0, 5, 10\}$	$(600, 12700)$
5	$\{-6, -2, 0, 2, 6\}$	$(200, 3200)$

Theorem 4.7.2 (Reach while stay). *Given a CLBF, if* $\forall q \in Q$, $\alpha_q(x)$ *satisfies* (4.29) *for* $D = A \backslash O$, *switching law* (4.30) *yields that* (4.19) *holds.*

Corollary 4.7.2 (Reach and stay while stay). *Given a CLBF satisfying* (4.24), *if* $\forall q$, $\alpha_q(x)$ *satisfies* (4.29) *for* $D = A \backslash int(B)$, *using switching law* (4.30) *yields that* (4.20) *holds.*

The proof of Corollary 4.7.2 is analogous to the proof of Theorem 4.7.2 and Corollary 4.7.1 and therefore not presented in Appendix A

4.7.6. CASE STUDIES

Let us revisit the systems in Table 4.1. For these case studies, we fix the control mode vector field $\mathcal{G}$ and synthesized controllers for the reach-while-stay specification $\texttt{CS}_1$, as specified for each system in Table 4.6. Across all these case studies, we use a population of 16 individuals, a maximum of 50 generations, and a maximum of 30 generations within CMA-ES. The mutation and crossover rates are both chosen to be 0.5. The number of test samples and maximum number of additional counterexamples are set to 100 and 300 respectively. For the counterexamples, a first-in-first-out principle is used. The (arbitrary) γ of the CLBF was set to $\gamma = 0.1$ and the precision parameter of dReal set to $\delta = 0.001$. The values of ε_i are obtained using bisection and the SMT solver and are also reported in Table 4.6. The GGGP algorithm and CMA-ES are implemented in Mathematica, running on an Intel Xeon CPU E5-1660 v3 3.00GHz using 8 CPU cores.

The used grammar is defined by $\mathcal{S}_V = \langle \text{const} \rangle + \langle \text{expr} \rangle$, $\mathcal{N}$ and $\mathcal{P}$ as shown in Table 4.7, and $\mathcal{P}^*$ is obtained by removing all recursive rules from $\mathcal{P}$. While this grammar restricts to polynomial CLBFs, the proposed approach can also be used for non-polynomial CLBFs. Finally, the maximum recursive rule depth is set to 7.

To show repeatability, the synthesis is again repeated 10 times for each benchmark. Statistics on the number of generations and the total synthesis time are shown in Table 4.8. With the exception of the third-order polynomial system, in all 10 runs a solution was found for each benchmark. For the third-order system only a single run did not find a solution within 50 generations. Given the found solutions, we use again bisection and the SMT solver to find a β such that the conditions in Corollary 4.7.1 hold. We find a β such that the conditions in Corollary 4.7.1 hold for: 4 solutions of the linear system, 1 of the 2nd-order system, 8 of the 3rd-order system, 0 of the pendulum system and 6 of the pendulum on cart system. Hence for these solutions the stronger specification $\texttt{CS}_2$ is guaranteed.

Table 4.7: Production rules $\mathcal{P}$.

$\mathcal{N}$	Rules
$\langle \text{expr} \rangle$	$::= \langle \text{expr} \rangle + \langle \text{expr} \rangle \mid \langle \text{pol} \rangle$
$\langle \text{pol} \rangle$	$::= \langle \text{pol} \rangle + \langle \text{pol} \rangle \mid \langle \text{const} \rangle \times \langle \text{mon} \rangle$
$\langle \text{mon} \rangle$	$::= \langle \text{var} \rangle \mid \langle \text{var} \rangle \times \langle \text{var} \rangle$
$\langle \text{var} \rangle$	$::= x_1 - x_{c,1} \mid \ldots \mid x_n - x_{c,n}$
$\langle \text{const} \rangle$	$::=$ Random Real $\in [-10, 10]$
$\langle \text{G} \rangle$	$::= \{\langle \text{lin} \rangle\} \mid \ldots \mid \{\langle \text{lin} \rangle, \langle \text{lin} \rangle, \langle \text{lin} \rangle\},$
$\langle \text{lin} \rangle$	$::= \langle \text{const} \rangle (x_1 - x_{c,1}) + \cdots + \langle \text{const} \rangle (x_n - x_{c,n})$ $\mid \langle \text{const} \rangle \langle \text{var} \rangle \mid \langle \text{const} \rangle$

Table 4.8: Results across 10 runs for continuous-time systems with sampled-data controllers, using a fixed template. μ: mean, σ: standard deviation.

System	η	number of generations				time [s]			
		min	max	μ	σ	min	max	μ	σ
1	0.01	3	5	3.8	0.79	11.55	21.34	15.67	0.79
2	0.01	7	11	9.1	1.52	32.26	67.92	47.81	12.32
3	0.001	6	50	16.7	16.03	86.39	524.96	205.08	138.02
4	0.001	3	12	7.6	3.03	32.97	185.96	106.72	54.84
5	0.001	5	16	8.6	3.47	48.02	155.17	85.2	35.18

One of the found solutions for the pendulum system is

$$V(x) = -4015.83 + 10.8526x_1' + 199.048x_1'^2 + 0.311673x_2 + 18.8116x_1'x_2 + 2.23916x_2^2,$$

where $x_1' = (0.75 + x_1)$. We manually design $\alpha_q(x)$ for all $q \in Q$ to be:

$$\alpha_1(x) = 100,\ \alpha_2(x), \alpha_3(x), \alpha_4(x) = 0,\ \alpha_5(x) = 500,$$

for which (4.29) holds. Figure 4.9 shows the phase plot of the closed-loop system for $\xi(0) \in \{(-\pi, 10), (-2, -5), (1.5, 0), (\pi, 10)\}$. It can be seen that indeed all trajectories satisfy $\texttt{CS}_1$.

For this solution, we could not find a β such that Corollary 4.7.1 holds. Nevertheless, by increasing the goal set to O to $[-1, -0.5] \times [-1.5, 1.5]$, it can be verified that for $\beta = -4012.3$ the conditions in Corollary 4.7.1 hold.

The used sampling times η are significantly larger than the minimal dwell-times reported in [158] and [120]. For example, for the pendulum on cart system benchmark we used $\eta = 0.001$ seconds, whereas [120] reports a theoretical minimum dwell-time of $2 \cdot 10^{-6}$ seconds.

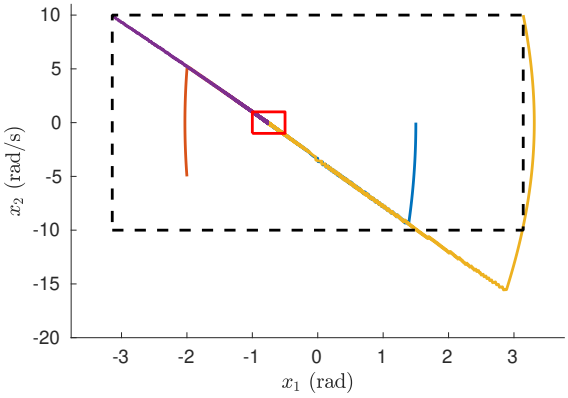


Figure 4.9: Phase diagram of different initial conditions for the pendulum system using a found CLBF. Dashed: initial set, red: goal set.

Table 4.9: Results for 10 runs for the Pendulum on a cart system without pre-defining $\mathcal{G}$.

	Min	Max	μ	σ
number of generations	4	14	7.6	2.84
t [s]	135.59	536.12	267.82	124.48

Evolving $\mathcal{G}$

Let us reconsider system 5 (i.e., pendulum on a cart) from Table 4.1 and specification $\texttt{CS}_1$, but without pre-specifying $\mathcal{G}$. We saturate the input with

$$u(t_k) = \max(-6, \min(6, g_{q_k}(\xi(t_k)))).$$

A separate gene for the controller modes $\mathcal{G}$ is used with start symbol $\mathcal{S}_G = \langle \mathrm{G} \rangle$ and the product rules in Table 4.4. The results for 10 runs are shown in Table 4.9. Comparing Table 4.8 with 4.9 we observe a comparable number of generations required to find a solution, although a longer computation time per generation is observed. However, the benefit is that no discretization of the input space is required. One of the found solutions is given by

$$\begin{aligned} V &= -22.2281 + 52.1542x_1^2 + 13.0965x_1x_2 + 17.3873x_2^2, \\ \mathcal{G} &= \{-11.0824x_1 - 13.2558x_2\}. \end{aligned}$$

Note that $\mathcal{G}$ consists of only a single mode, hence no switching law is required when implementing this controller. Finally, for $\beta = -19.5313$, V satisfies (4.9), hence using this controller also guarantees $\texttt{CS}_2$.

4.7.7. Discussion

Comparing the hybrid framework to the specialized synthesis for sampled-data systems, we observe that we are not able to find sampled-data controllers for all systems in Table

4.1 (systems 3 and 4 failed) using the hybrid framework, as opposed to the specialized case. In the case of system 3, this is due to time-out issues with the SMT solver as a result of the increased complexity w.r.t. the increased system order. For system 4, the hybrid approach is too conservative. Nevertheless, the specialized case requires the additional assumption 4.7.1 on the system dynamics, which is not required in the hybrid framework. Moreover, in this specialized framework, the user needs to bound the Lagrangian remainder beforehand, whereas in the hybrid framework in Section 4.5, this is not required.

Comparing our approach to [120], in which a switching controller is proposed based on an automatically synthesized CLBF, our method does not require a template solution, does not require a pre-specified set of controller modes, and our used sampling time is larger than the (conservative) lower bounds on the minimum dwell-times of the switching controller in [120].

The sampling times in our framework can be made less conservative by using less conservative over-approximations of the reachable set, e.g. by using higher order Taylor series approximations or using local bounds rather than for the entire domain. Finally, the functions $\alpha_q(x)$ that simplify the switching condition are currently synthesized by hand. In future work, the aim is to automate this synthesis as well, for example by again using the combination of GP with SMT solvers.

4.8. Verification of near-optimal controllers

In the previous sections, we synthesized controllers by co-evolving them together with a (C)LBF. In this section we use the proposed framework to verify near-optimal controllers obtained through near-optimal control synthesis. The controllers are obtained through reinforcement learning (RL) [149], and are verified using the certificate function synthesis presented before. Specifically, we apply model-based RL control design which returns a near-optimal controller described by an analytic expression [88]. The proposed approach is demonstrated on the synthesis of an optimal controller for an Anti-lock Braking System (ABS), which actively controls the wheel dynamics during severe braking. Its purpose is to maximize braking performance while avoiding excessive wheel slip or wheel lock and thus maintaining the vehicle's ability to steer. RL is used to design a controller to minimize the braking distance and through the use of the Lyapunov Barrier function we formally verify convergence to standstill and bounds on the braking distance.

4.8.1. Problem definition

Let us again consider a nonlinear system of the form

$$\dot{\xi}(t) = f(\xi(t), u(t)), \tag{4.31}$$

where $\xi(t) \in \mathcal{X} \subset \mathbb{R}^n$ and $u(t) \in \mathcal{U} \subset \mathbb{R}^m$ denote the state and input respectively. In this section we consider an optimal control design method for discrete-time systems, requiring system (4.31) to be discretized. To ensure aspects like reachability and safety are not lost for the original system, we formally verify the resulting control law w.r.t. the original continuous-time model.

OPTIMAL CONTROL DESIGN

The discretized system (4.31) is described by the state transition function

$$x_{k+1} = f'(x_k, u_k), \tag{4.32}$$

with $x_k, x_{k+1} \in \mathcal{X}$ and $u_k \in \mathcal{U}$. This function is assumed to be available, but it does not have to be stated by explicit equations; it can be, for instance, a generative model given by a numerical simulation of complex differential equations. The control goal is specified through a *reward function* which assigns a scalar reward $r_{k+1} \in \mathbb{R}$ to each state transition from x_k to x_{k+1}:

$$r_{k+1} = \rho(x_k, u_k, x_{k+1})\,. \tag{4.33}$$

This function is defined by the user and typically calculates the reward based on the difference between the current state and a given constant reference state x_r that should be attained.

The goal is to find an (approximately) optimal control policy $\pi : \mathcal{X} \to \mathcal{U}$ such that in each state it selects a control action so that the cumulative discounted reward over time, called the return, is maximized:

$$R^\pi = E\Big\{\sum_{k=0}^{\infty} \gamma^k \rho\big(x_k, \pi(x_k), x_{k+1}\big)\Big\}\,. \tag{4.34}$$

Here $\gamma \in (0, 1)$ is a discount factor and the initial state x_0 is drawn uniformly from the state-space domain $\mathcal{X}$ or its subset. Hence the considered control problem is:

Problem 4.8.1

Design a control policy $\pi : \mathcal{X} \to \mathcal{U}$ such that the return is maximized.

FORMAL VERIFICATION

Given the closed-loop system Σ_{cl} given by the data $(\mathcal{X}, f_{\text{cl}})$ (see Definition 2.2.6), where $f_{\text{cl}}(x) = f(x, \pi(x))$, the next goal is to formally verify whether it satisfies specification $\texttt{CS}_1$, here reformulated for continuous-time systems:

$\texttt{CS}_1$ *Reach while stay (RWS)*: all solutions ξ to Σ_{cl} starting from the initial set I eventually reach the goal set O, while staying within the safe set S:

$$\forall \xi \in \mathcal{S}_{\Sigma_{\text{cl}}}(I), \exists T, \forall t \in [0, T] : \xi(t) \in S \wedge \xi(T) \in O. \tag{4.35}$$

We addresses the following problem:

Problem 4.8.2

Given the compact sets (S, I, O) and closed-loop system Σ_{cl}, verify that specification $\texttt{CS}_1$ is satisfied.

This verification is done by means of automatic synthesis of a Lyapunov Barrier function.

4.8.2. Methodology

Optimal controller design

The return (4.34) is approximated by the value function $\mathcal{V}^\pi : X \to \mathbb{R}$ defined as:

$$\mathcal{V}^\pi(x) = E\Big\{\sum_{k=0}^{\infty} \gamma^k \rho\big(x_k, \pi(x_k), x_{k+1}\big) \Big| \, x_0 = x\Big\}. \tag{4.36}$$

An approximation of the optimal value function, denoted by $\hat{\mathcal{V}}^*(x)$, can be computed by solving the Bellman optimality equation

$$\hat{\mathcal{V}}^*(x) = \max_{u \in \mathcal{U}} \Big[\rho\big(x, \pi(x), f'(x,u)\big) + \gamma \hat{V}^*\big(f'(x,u)\big)\Big]. \tag{4.37}$$

4

To simplify the notation, in the sequel, we drop the hat and the star superscript, i.e. $\mathcal{V}(x)$ is used to denote the approximately optimal value function.

To compute $\mathcal{V}(x)$, we use the fuzzy V-iteration algorithm [25]. Given the process model (4.32) and the reward function (4.33), define the set $C = \{c_1, \ldots, c_N\}$ of points on a regular grid in the state space. Further define a vector of triangular membership functions $\phi = [\phi_1(x), \ldots, \phi_N(x)]^\top$ so that each $\phi_i(x)$ is centered at c_i, i.e., $\phi_i(c_i) = 1$ and $\phi_j(c_i) = 0, \forall j \neq i$. The membership functions are normalized so that $\sum_{j=1}^{N} \phi_j(x) = 1, \forall x \in \mathcal{X}$. Finally, define a finite set of discrete control input values $U = \{u^1, u^2, \ldots, u^M\} \subset \mathcal{U}$.

The value function is approximated by the following basis-function expansion

$$\mathcal{V}(x) = \theta^\top \phi(x),$$

where $\theta = [\theta_1, \ldots, \theta_N]^\top \in \mathbb{R}^N$ is a parameter vector found through the following value iteration:

$$\theta_i \leftarrow \max_{u \in U} \left[\rho(c_i, u, f'(c_i,u)) + \gamma \theta^\top \phi\left(f'(c_i,u)\right)\right] \tag{4.38}$$

for $i = 1, 2, \ldots, N$. This iteration is guaranteed to converge [25] and terminates when the following condition is satisfied:

$$||\theta - \theta^-||_\infty \leq \epsilon, \tag{4.39}$$

with θ^- the parameter vector calculated in the previous iteration and ϵ a user-defined convergence threshold. Fuzzy value iteration is very effective for second and third-order systems; computing the optimal value function is a matter of seconds. However, the computational and memory requirements grow exponentially and the method becomes impractical for systems above order four.

There are two principal ways to derive the control policy from the value function [88]. The first one is based on an online maximization of the Bellman optimality equation's right-hand side (hill-climbing policy), while the second one applies the Bellman equation offline and uses basis functions to interpolate online (interpolated policy). Here we apply the latter method. For all states c_i, $i = 1, 2, \ldots, N$, the optimal control action p_i is computed offline as follows:

$$p_i = \arg\max_{u \in U} \left[\rho(c_i, u, f'(c_i,u)) + \gamma \theta^\top \phi\left(f'(c_i,u)\right)\right] \tag{4.40}$$

and the control actions are collected in a vector: $p = [p_1, \ldots, p_N]^\top \in U^N$. In an arbitrary state x, the corresponding control action is then obtained by interpolation:

$$u = p^\top \phi(x), \tag{4.41}$$

where $\phi(x)$ are the same basis functions as defined for $\mathcal{V}(x)$. An obvious advantage of this method is its computational simplicity: most computations are done offline (vector p is actually obtained for free as a byproduct of the fuzzy value iteration algorithm) and the online interpolation is computationally cheap. Another advantage is that (4.41) directly produces continuous control actions. However, the control signal is not necessarily smooth and the interpolation can also result in a steady-state error. Therefore, we use a simplified version of the symbolic approximation method proposed in [88], which is computationally effective and also yields smooth controls. We build an analytic approximation of the policy in the following way. For a typical optimal control problem, the policy surface can be split into saturated parts where the control signal attains the minimal or maximal possible value, and a rather steep transition between the two parts. The transition is generally nonlinear, but often can be well enough approximated by a linear function. The overall policy is then described by:

$$u \quad = \quad \mathrm{sat}\big(Kx\big), \tag{4.42}$$

where K is obtained by using linear regression on samples of the steep transition augmented with samples on the boundaries between the transition and the saturated hyper planes. The function $\mathrm{sat}(\cdot)$ defined as follows:

$$\mathrm{sat}(z) = \max\left(U_{\min}, \min\left(U_{\max}, z\right)\right).$$

For general systems for which such an approximation does not suffice; the aforementioned symbolic approximation in [88] can be used within the framework presented in this chapter.

VERIFICATION THROUGH LYAPUNOV BARRIER FUNCTIONS

The safety and reachability specification $\mathtt{CS}_1$ is again verified indirectly by means of a LBF (see Definition 4.3.1). For the considered continuous-time system, the LBF conditions reduce to:

$$\forall x \in I : V(x) \leq 0, \tag{4.43a}$$
$$\forall x \in \partial S : V(x) > 0, \tag{4.43b}$$
$$\forall x \in A \backslash O : \langle \nabla V(x), f(x, \pi(x)) \rangle \leq -\gamma, \tag{4.43c}$$

where $S \subseteq \mathcal{X}$, $I, O \subseteq int(S)$, $\gamma > 0$ and $A := \{x \in S \mid V(x) \leq 0\}$. As we have seen before, it follows from Theorem 4.3.1 that existence of a LBF V implies that the closed-loop system satisfies specification $\mathtt{CS}_1$. This LBF is synthesized as described in Section 4.5.

4.8.3. CASE STUDY: ANTI-LOCK BRAKING SYSTEM

The proposed methodology is demonstrated on an anti-lock braking system. The control synthesis for an ABS system poses challenges due to the highly nonlinear and uncertain

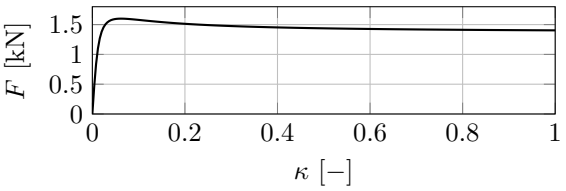


Figure 4.10: Longitudinal force vs. tire slip for a wet asphalt with a water level of 3 mm.

dynamic behavior of the wheel slip phenomenon. A longitudinal model of a corner vehicle is given by:

$$\begin{cases} \dot{v}(t) & = -\frac{1}{m}F(\kappa), \\ \dot{\omega}(t) & = \frac{r_t}{J}F(\kappa) - \frac{\sigma(\omega(t))}{J}u(t), \\ \dot{s}(t) & = v(t), \end{cases} \tag{4.44}$$

where $v(t)$ denotes the vehicle velocity, $\omega(t)$ the wheel angular velocity, $s(t)$ the braking distance, r_t the tire effective rolling radius, $u(t)$ the braking torque, m the corner vehicle mass and J the wheel moment of inertia. Moreover, $F(\kappa)$ is the longitudinal force due to the wheel slip κ:

$$F(\kappa) = mgd\sin(c\tan^{-1}\big(b(1-e)\kappa + e\tan^{-1}(b\kappa)\big), \tag{4.45}$$

with b, c, d and e road surface-specific constants and

$$\kappa = 1 - \frac{\omega(t)r_t}{v(t)}$$

the tire slip. Finally, $\sigma : \mathbb{R} \to \mathbb{R}$ is a continuous approximation of the signum function defined as

$$\sigma(x) = \tanh(100x). \tag{4.46}$$

In this case study we use the parameters $J = 1.2\ \mathrm{kg}\cdot\mathrm{m}^2$, $r_t = 0.305\ \mathrm{m}$, $m = 407.75\ \mathrm{kg}$ and $g = 9.81\ \mathrm{m/s}^2$. We consider the slip force parameters $b = 55.56$, $c = 1.35$, $d = 0.4$ and $e = 0.52$, which correspond to wet asphalt for a water level of 3 mm [56]. Figure 4.10 shows the resulting force for different wheel slip values.

The choice of safe set, goal set and initial set are motivated as follows. According to the EU regulation N13 [155], for wet asphalt the maximum (initial) longitudinal velocity is 90 km/h (=25 m/s). The ABS is initialized of a slip angle of approximately 0 (i.e. $x_2 = x_1/r_t$) and is active until the longitudinal velocity meets the threshold of 5 km/h (= 5/3.6 m/s). Since the radius r_t can deviate slightly, we inflate the initial angular velocity to be bounded by $x_1/(r_t + \delta_r) \leq x_2 \leq x_1/(r_t - \delta_r)$. Finally, we impose an absolute maximum braking distance of 100 meters. This motivates the following choices for the safe set, initial set and

Table 4.10: Value iteration parameters.

Parameter	Symbol	Value	Units
State domain	$\mathcal{X}$	$[0, 10] \times [0, 33]$	m/s ×rad/s
Num. of membership func.	N	$961 = 31 \times 31$	–
Discount factor	γ	0.9999	–
Convergence threshold	ϵ	0.001	–
Sampling period	T_s	0.001	s

goal set:

$$
\begin{aligned}
S &= [0, 30] \times [-10, 30/r_t] \times [-10, 100]\,, \\
I &= \left\{ x \in S \;\middle|\; \frac{5}{3.6} \le x_1 \le 25,\ \frac{x_1}{r_t + \delta_r} \le x_2 \le \frac{x_1}{r_t - \delta_r},\right. \\
&\qquad\qquad \left. 0 \le x_3 \le 0.1 \right\}, \\
O &= \{ x \in S \mid x_1 \le 5/3.6 \}\,.
\end{aligned}
$$

Given this safe set, the upper bound on the braking distance compared to the braking distance obtained from simulation is quite conservative. The bounds on the safe set could be chosen to be tighter, but this comes at the cost of longer computation times of the used SMT solver, assuming for the chosen bound a solution exists.

CONTROLLER DESIGN

For optimal control design, we use a discrete-time model obtained by numerically integrating the continuous-time dynamics (4.44) using the fourth-order Runge-Kutta method with the sampling period of $T_s = 0.001$ s. The state is the car velocity, $x_k = [v_k, \omega_k]^\top$, and the reward function is defined as:

$$r_{k+1} = \rho(x_k, u_k, x_{k+1}) = -x^\top Q x \tag{4.47}$$

with $Q = \mathrm{diag}(1, 0)$ a weighting matrix, specifying that the car velocity must reach zero, regardless of the wheel angular velocity.

The parameters of the fuzzy value iteration algorithm are listed in Table 4.10. The number of membership functions for each state variable was chosen quite large (31) in order to get a dense coverage of the state-space domain of interest. The discount factor $\gamma = 0.999$ is selected close to one, so that not too much discounting takes place even at the end of a typical closed-loop transient which lasts about 1200 samples ($\gamma^{1200} \approx 0.3$).

The resulting policy is:

$$u(x) \quad = \quad \mathrm{sat}\big(k_1 x_1 + k_2 x_2 + k_0\big), \tag{4.48}$$

with $k_1 = 474.4$, $k_2 = 152.2$ and $k_0 = 1091.4$. This policy is shown in Figure 4.11. For an initial condition of 90 km/h with a zero wheel slip and a sampling time of 0.001 seconds, we obtain from simulation a braking distance of 81.7874 meter. In comparison, in the case of a wheel lock, the braking distance is 90 meter.

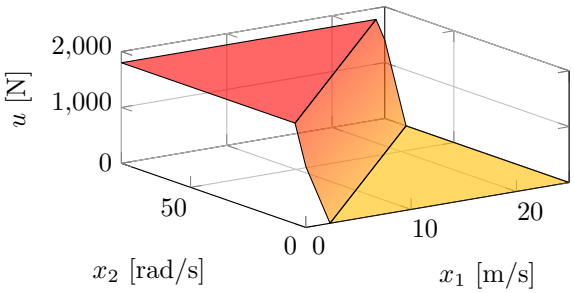


Figure 4.11: Piecewise linear policy (4.48) for wet asphalt with water level of 3 mm.

Table 4.11: Production rules $\mathcal{P}$.

$\mathcal{N}$	Rules
$\langle \text{pol} \rangle$	$::= \langle \text{pol} \rangle + \langle \text{pol} \rangle \mid \langle \text{const} \rangle \times \langle \text{mon} \rangle$
$\langle \text{mon} \rangle$	$::= \langle \text{var} \rangle \mid \langle \text{var} \rangle \times \langle \text{mon} \rangle$
$\langle \text{var} \rangle$	$::= x_1 \mid \ldots \mid x_3$
$\langle \text{const} \rangle$	$::= \text{Random Real} \in [-10, 10]$

Verification

The LBF synthesis is implemented in Mathematica 11.1 and performed on a desktop with Intel Xeon CPU E5-1660 v3 3.00 GHz using 14 parallel CPU cores. Given the specification sets S, I, O, we bias our search by having a grammar that imposes a template for the LBFs that consists of a polynomial plus a pre-defined barrier function. With the safe set written as $S = \Pi_{i=1}^{3}[\underline{s}_i, \overline{s}_i]$, we use a pre-defined barrier function $B : \mathbb{R}^3 \to \mathbb{R}$ of the form:

$$B(x) = \sum_{i=1}^{3} \frac{\langle \text{const} \rangle}{x_1 - \underline{s}_i + \varepsilon}, \tag{4.49}$$

where ε is a parameter that is chosen to be $\varepsilon = 0.001$. In our grammar, the starting symbol of the LBF is then selected to be

$$V(x) = \mathcal{S} = \langle \text{const} \rangle + \langle \text{pol} \rangle + B(x), \tag{4.50}$$

where $\langle \text{const} \rangle$ and $\langle \text{pol} \rangle$ denote nonterminals of a constant and polynomial. Besides the starting symbol, the remainder of the grammar is chosen to be as given in Table 4.11.

We consider a population of 28 individuals with a maximum of 500 generations and fix the number of CMA-ES generations to be 40. For the sample-based fitness, we start per inequality with a set of 100 samples, which can be complemented with up to 300 counterexamples, where a first-in-first out principle is used. The rates of the genetic operators are 0.5 for both crossover and mutation and the maximum tree recursion depth is chosen to be 6.

Synthesis is performed over 8 independent runs, in which 5 times an LBF is found before the maximum number of generations is met. For the 5 successful runs, the statistics

Table 4.12: Statistics on the number of generations and total time for 5 successful LBF synthesis runs.

	Min	Max	Mean	SD
# generations	142	437	257.4	118.2
Total time [min]	51.3	315.6	161.1	104.4

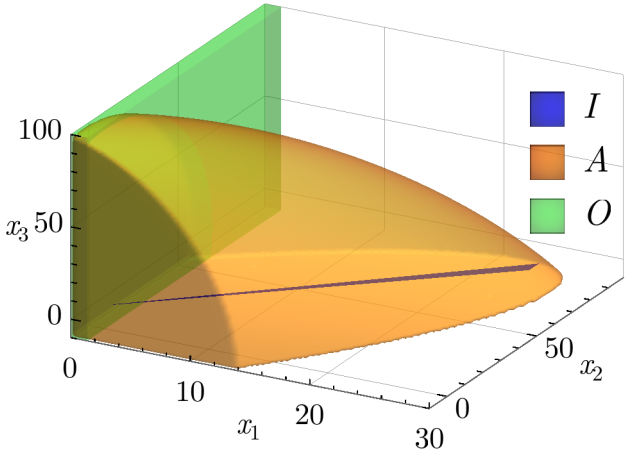


Figure 4.12: Sublevel set A of the synthesized LBF, initial set I and goal set O. Trajectories starting in I remain in A until they eventually reach O.

on the number of generations and elapsed time is shown in Table 4.12. An example of a found solution is:

$$\begin{aligned} V(x) = &- 590553. + 985.64x_1 + 2303.02x_1^2 + 1230.37x_2 \\ &- 1035.76x_1x_2 + 167.855x_2^2 - 1003.36x_3 \\ &+ 94.5467x_1x_3 + 6.13646x_1^2x_3 + 69.6233x_3^2. \\ B(x) = &\frac{685.651}{0.001 + x_1} + \frac{621.366}{10.001 + x_2} + \frac{631.618}{10.001 + x_3}. \end{aligned}$$

The corresponding sublevel set A and sets I and O are shown in Figure 4.12. Note that set A can be seen as a forward invariant sublevel up until O is reached.

By the existence of an LBF, specification $\texttt{CS}_1$ holds w.r.t. the safe set, initial set and goal set. This implies that for all trajectories starting in the initial set, eventually the target velocity op 5km/h is reached and the braking distance up to that point is guaranteed to be below 100 meters. Note that if we select the initial set to be equal to the found sublevel set A, the conditions in (4.8) still hold, hence for all trajectories starting in A specification $\texttt{CS}_1$ holds.

4.9. Conclusion

In this chapter we proposed a framework for formal controller synthesis for hybrid systems, by means of co-evolution of controllers and Lyapunov-like functions. Additionally, we discussed a specialized framework for nonlinear systems with sampled-data controllers, and used the synthesis of LBF for the verification of near-optimal controllers obtained through reinforcement learning.

The methods have been shown for systems with up to 5 states (in case of the sampled-data system modelled as a hybrid system). However, general conclusions about scalability and computation time are highly speculative. To improve the convergence, the method is best used in combination with expert-knowledge by means of the grammar.

5

Reachability-based synthesis

In this chapter we propose a counterexample-guided inductive synthesis framework for the formal synthesis of closed-form sampled-data controllers for nonlinear systems to meet general STL specifications. Rather than stating the STL specification for a single initial condition, we consider an (infinite) set of initial conditions. Candidate solutions are proposed using genetic programming, which evolves controllers based on a finite number of simulations. Subsequently, the best candidate is verified using reachability analysis; if the candidate solution does not satisfy the specification, an initial condition violating the specification is extracted as a counterexample. Based on this counterexample, candidate solutions are refined until eventually a solution is found. The resulting sampled-data controller is expressed as a closed-form expression, enabling the implementation in embedded hardware with limited memory and computation power. The effectiveness of our approach is demonstrated for multiple systems.

This chapter has been published in [157]. Specifically, all sections with the exception of Section 5.1 and 5.5 are verbatim from [157]; Section 5.5 is an extension on [157, Section 6].

5.1. Introduction

In the previous chapter, we used genetic programming to co-evolve controllers and candidate certificate functions, where the latter enabled the verification of the closed-loop system. However, by also synthesizing both a certificate function next to the controller, the search space is increased considerably. In this chapter, we only synthesize a controller and verify the closed-loop system by means of reachability analysis [6]. Whereas in the previous chapter certificate functions could only address a subset of signal temporal logic (STL) specifications, in this chapter, we synthesize controllers for full STL specifications. However, instead of considering general hybrid systems, we restrict our focus to continuous systems with disturbances and a sampled-data implementation of the controller. The approach proposed in this chapter takes again the shape of a CEGIS framework, where we utilize the recent work on model checking for STL [125] and counterexample generation using reachability analysis [82].

Previous work using reachability analysis for formal controller synthesis for reach-avoid problems include [138, 139, 141]. In [138, 141] a framework is proposed to synthesize linear controllers for a sequence of time intervals, whereas [139] proposed an MPC-based approach. Our main contributions are twofold: first of all, our framework is able to synthesize closed-form sampled-data controllers for full STL specifications; our second contribution is the definition of quantitative semantics for RTL (see Section 2.3.3), and we prove that the quantitative semantics is sound and complete. Similar to the quantitative semantics of STL (see Section 2.3.1), these quantitative semantics provide a measure of how robustly a formula is satisfied.

This chapter is organized as follows. First, the problem definition and solution approach are defined in Section 5.2. In Section 5.3 we define the quantitative semantics of RTL. In Section 5.4 and 5.5, we detail how candidate solutions are proposed and verified, respectively. Section 5.6 discusses dealing with conservatism. The effectiveness of the approach is demonstrated on several case studies in Section 5.7. The results are discussed in Section 5.8 and the chapter is concluded in Section 5.9.

5.2. Problem definition and solution approach

In this chapter we consider disturbed continuous open-loop systems $\Sigma_{\text{ol}} = (\mathbb{R}^n, F_{\text{ol}})$, where

$$F_{\text{ol}}(x, u) = \{f(x, u, \omega) \mid \omega \in \Omega\}, \tag{5.1}$$

with $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^n$, and $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and $\omega \in \Omega \subset \mathbb{R}^l$ denote states, inputs and bounded disturbances, respectively. Note that under this model, the disturbance $\omega \in \Omega$ acting on the system can change at every time instant. We consider *disturbance realizations* $w : \mathbb{R}_{\geq 0} \to \Omega$, which are time-dependent realizations of this uncertainty parameter ω. We consider sampled-data time-varying state-feedback controllers $\kappa : \mathbb{R}_{\geq 0} \times \mathbb{R}^n \to \mathbb{R}^n$ such that $u(t) = \kappa(t_k, \xi(t_k))$ for all $t \in [t_k, t_k + \eta)$, where t_k denotes the k-th sampling instant and η denotes the sampling time. This results in a sampled-data closed-loop system Σ^{sd} with data $(\mathbb{R}^{n+1}, F'_{\text{ol}}, \kappa', \eta)$ (see Definition 2.2.7), where $F'_{\text{ol}}(s, u) = (F_{\text{ol}}(s_{\text{x}}, u), 1)$, $\kappa'(s) = \kappa(s_{\text{t}}, s_{\text{x}})$, $s = (s_{\text{x}}, s_{\text{t}})$.

The goal of this chapter is formalized in the following:

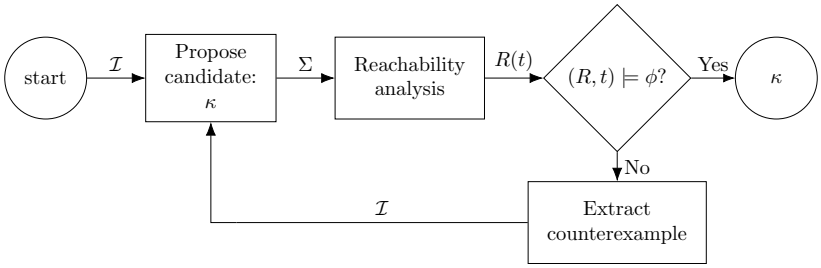


Figure 5.1: Schematic overview of the algorithm.

Problem 5.2.1

Given an STL formula φ satisfying Assumption 2.3.1, initial set $I \subset \mathbb{R}^n$, and the open-loop system (5.1), synthesize a closed-form sampled-data time-varying controller $\kappa : \mathbb{R}_{\geq 0} \times \mathbb{R}^n \to \mathbb{R}^m$ such that the closed-loop system Σ^{sd} satisfies

$$\forall \xi \in \mathcal{S}_{\Sigma^{\text{sd}}}(I) : (\xi, 0) \models \varphi \tag{5.2}$$

5

In Theorem 2.3.1 in Section 2.3.3, we have seen that (5.2) can be proven by translating the STL formula φ to the RTL formula Ψ using the transformation Υ, and subsequently proving $(R, 0) \models \Psi$. In this chapter, we propose a counterexample-guided inductive synthesis (CEGIS) framework to synthesize a controller such that $(R, 0) \models \Psi$, thereby solving Problem 4.2.1. The proposed framework consists of iteratively proposing a controller obtained through GGGP[1] and then formally verifying the RTL formula using reachability analysis. The proposed controller is designed based on a set of simulated trajectories, which correspond to pairs of initial conditions and disturbance realizations. The underlying idea is that these simulations are relatively fast to compute and provide a sensible search direction for the synthesis, whereas the reachability analysis verifies the resulting controller.

For a given open-loop system Σ_{ol}, STL formula φ, initial set I, and grammar $(\mathcal{N}, \mathcal{S}, \mathcal{P})$, the algorithm is initialized as follows:

I1) The RTL formula Ψ is computed using $\Psi = \Upsilon(\varphi)$ (see Theorem 2.3.1).

I2) The set of pairs of initial conditions and disturbance realizations $\mathcal{I}$ is initialized by randomly choosing n_{s} initial conditions $\{x^1, \ldots, x^{n_{\text{s}}}\} \subset I$ and with random disturbance realizations $w^i : \mathbb{R}_{\geq 0} \to \Omega$, such that $\mathcal{I} = \left\{\left(x^1, w^1\right), \ldots, \left(x^{n_{\text{s}}}, w^{n_{\text{s}}}\right)\right\}$.

Given the initialized data, the algorithm goes through the following cycle, illustrated in Figure 5.1, where each cycle is referred to as a *refinement*:

A1) A candidate solution is proposed using GGGP, based on simulation trajectories corresponding to the set $\mathcal{I}$.

[1]While GGGP evolves a population of controllers, only the controller with the highest fitness is returned as the proposed controller.

A2) For the given candidate controller, the reachable set is computed.

A3) Based on the reachable set, either:

(a) $(R,t) \models \Psi$, thus a controller solving Problem 4.2.1 is found.

(b) $(R,t) \not\models \Psi$, and a counterexample is extracted in the form of an initial condition for which there exists a disturbance realization s.t. the RTL specification is violated. For this initial condition, a disturbance realization is optimized. This pair of initial condition and disturbance realization is added to $\mathcal{I}$ and the algorithm returns to step A1).

(c) $(R,t) \not\models \Psi$ and a maximum of refinements is reached, therefore the algorithm is terminated.

To quantify the violation or satisfaction of an RTL formula, we introduce quantitative semantics for RTL in the next section. The proposal of a candidate controller in step A1) is discussed in Section 5.4. The verification and counterexample generation in step A3) is discussed in Section 5.5.

5.3. Quantitative semantics

Inspired by the quantitative semantics of STL [40, 45], we define quantitative semantics for RTL in this section. These quantitative semantics provide a *robustness measure* on how well the formula is satisfied. For an RTL formula Ψ with propositional subformulae ψ, the quantitative semantics is given by functions $P(R,\Psi,t)$ and $\varrho(x,\psi)$, respectively, recursively defined as:

$$\begin{aligned}
\varrho(x,\text{true}) &= +\infty,\\
\varrho(x,h(x)\geq 0) &= h(x),\\
\varrho(x,\neg\psi) &= -\varrho(x,\psi),\\
\varrho(x,\psi_1\wedge\psi_2) &= \min(\varrho(s,\psi_1),\varrho(s,\psi_2)),\\
P(R,\mathcal{A}\psi,t) &= \min_{x\in R(t)} \varrho(x,\psi),\\
P(R,\Psi_1\vee\Psi_2,t) &= \max(P(R,\Psi_1,t),P(R,\Psi_2,t)),\\
P(R,\Psi_1\wedge\Psi_2,t) &= \min(P(R,\Psi_1,t),P(R,\Psi_2,t)),\\
P(R,\bigcirc_a\Psi,t) &= P(R,\Psi,t+a).
\end{aligned}$$

The quantitative semantics of STL are sound and complete [39, 45]. The quantitative semantics of RTL also have these properties:

Theorem 5.3.1 (Soundness and completeness)**.** *Let Ψ be an RTL formula, R a reachable set, and t a time instance, then:*

1) $P(R,\Psi,t) > 0 \Rightarrow (R,t)\models\Psi$ *and* $(R,t)\models\Psi \Rightarrow P(R,\Psi,t)\geq 0$,

2) $P(R,\Psi,t) < 0 \Rightarrow (R,t)\not\models\Psi$ *and* $(R,t)\not\models\Psi \Rightarrow P(R,\Psi,t)\leq 0$.

Remark 5.3.1. *Note that $P(R,\Psi,t)=0$ does not imply $(R,t)\models\Psi$ nor $(R,t)\not\models\Psi$. This is because on the boundary of an inequality, the distinction between inclusion or exclusion is lost within the quantitative semantics. That is, if $\varrho(x,\psi)=0$, we also have $\varrho(x,\neg\psi)=0$, hence the quantitative semantics of two mutually exclusive logic formulae evaluate to the same value.*

The proof of Theorem 5.3.1 can be found in Appendix A.10. Consider an STL formula φ satisfying Assumption 2.3.1 for some c and the corresponding RTL formula $\Psi=\Upsilon(\varphi)$ in the form of (2.11). Using the equivalences $\bigcirc_a(\Psi_1\wedge\Psi_2)=\bigcirc_a\Psi_1\wedge\bigcirc_a\Psi_2$ and rewriting ψ_{ijk} in disjunctive normal form, we can express the RTL formula as:

$$\Psi=\bigwedge_{i\in\mathsf{I}}\bigvee_{j\in\mathsf{J}_i,k\in\mathsf{K}_{ij}}\Psi'_{ijk}, \tag{5.3a}$$

$$\Psi'_{ijk}=\bigcirc_{j\frac{c}{2}}\mathcal{A}\bigvee_{a\in\mathsf{A}^{ijk}}\bigwedge_{b\in\mathsf{B}_a^{ijk}}h_{ab}^{ijk}(x)\sim 0, \tag{5.3b}$$

where A^{ijk} and B_a^{ijk} denote finite index sets, $\sim\in\{\geq,>\}$ and $h_{ab}^{ijk}(x)\sim 0$ is a predicate over x. Using the quantitative semantics defined in Section 5.3, the robustness measure of this RTL formula is given by

$$P(R,\Psi,0)=\min_{i\in\mathsf{I}}\left(\max_{j\in\mathsf{J}_i,k\in\mathsf{K}_{ij}}P(R,\Psi'_{ijk},0)\right), \tag{5.4a}$$

$$P(R,\Psi'_{ijk},0)=\min_{x\in R\left(j\frac{c}{2}\right)}\left(\max_{a\in\mathsf{A}^{ijk}}\left(\min_{b\in\mathsf{B}_a^{ijk}}h_{ab}^{ijk}(x)\right)\right). \tag{5.4b}$$

5.4. CANDIDATE CONTROLLER SYNTHESIS

In this section, we detail step A1) of the proposed algorithm in Section 5.2, i.e., the proposal of a candidate controller. The candidate controller is synthesized using GGGP, by maximizing an approximation of the robustness measure, which is based on a finite number of simulated trajectories. The sampling time is equal to $c/2$ to coincide with the time instances at which the robustness measure $P(R,\Psi,0)$ is evaluated. For an RTL formula of the form (2.11), the first and the final time instances of relevance τ_0 and τ_f, are given by $\tau_0=0$ and $\tau_\mathrm{f}=\frac{c}{2}\max_{i\in\mathsf{I}}|\mathsf{J}_i|$, respectively. Given a candidate controller $\kappa:\mathbb{R}_{\geq 0}\times\mathbb{R}^n\to\mathbb{R}^m$, a set of pairs of initial conditions and disturbance realizations $\mathcal{I}$, and a time instance τ_q, we consider an approximated reachable set $\hat{R}^\kappa_\mathcal{I}(\tau_q)$ formed by all corresponding simulated trajectories $x:\mathbb{R}_{\geq 0}\to\mathbb{R}^n$:

$$\hat{R}^\kappa_\mathcal{I}(\tau_q)=\{x(\tau_q)\mid(x(\tau_0),w)\in\mathcal{I}\}.$$

Provided this set $\hat{R}^\kappa_\mathcal{I}(\tau_q)$, we approximate the robustness measure by $P(\hat{R}^\kappa_\mathcal{I},\Psi,0)$.

5.4.1. OUTLINE OF THE CANDIDATE CONTROLLER SYNTHESIS

The proposal of a candidate controller in step A1) undergoes the following steps, which are also illustrated in Figure 5.2:

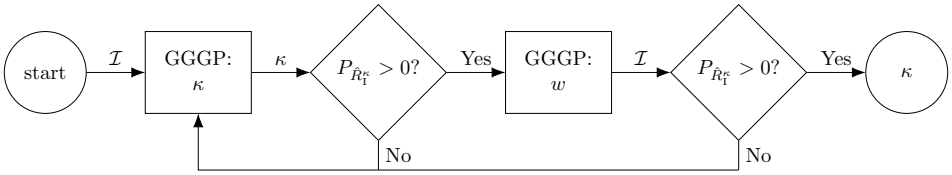


Figure 5.2: Schematic overview of the synthesis of candidate controller.

A1.a) We synthesize an analytic expression $\kappa : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^m$ by using GGGP to solve:

$$\arg\max_{\kappa} \; P(\hat{R}_{\mathcal{I}}^{\kappa}, \Psi, 0). \tag{5.5}$$

If for the resulting controller κ the robustness measure approximation $P(\hat{R}_{\mathcal{I}}^{\kappa}, \Psi, 0)$ is negative, this optimization step in (5.5) is repeated. Otherwise, the algorithm continues to the next step.

A1.b) For each initial condition x^i in $\mathcal{I}$, an analytic expression for a disturbance realization $w^i : \mathbb{R} \to \Omega$ is synthesized using GGGP, in which the robustness measure approximation is minimized, i.e.:

$$\begin{aligned} \arg\max_{w^i} \quad & -P(\hat{R}_{\mathcal{I}}^{\kappa}, \Psi, 0), \\ \text{subject to} \quad & \mathcal{I} = \{(x^i, w^i)\}. \end{aligned} \tag{5.6}$$

If the corresponding robustness degree approximation $P(\hat{R}_{\mathcal{I}}^{\kappa}, \Psi, 0)$ is negative, the algorithm returns to step A1.a). Otherwise, if for all updated disturbance realizations the robustness measure approximation is positive, i.e., $\forall i$, we have $P(\hat{R}_{\{(x^i,\omega^i)\}}^{\kappa}, \Psi, 0) > 0$, the algorithm returns a candidate controller.

5.4.2. Reference-tracking controllers

To speed up the synthesis, we impose a structure to the solution, based on a nominal reference trajectory $x_{\text{ref}}(t)$ and a corresponding feedforward input $u_{\text{ff}}(t)$. That is, we consider time-varying reference-tracking controllers of the form:

$$\kappa(t, x(t)) = u_{\text{ff}}(t) + \kappa_{\text{fb}}(t, x(t) - x_{\text{ref}}(t)). \tag{5.7}$$

where $\kappa_{\text{fb}} : \mathbb{R}_{\geq 0} \times \mathbb{R}^n \to \mathbb{R}^m$ is a time-varying feedback controller. The feedforward input and reference trajectory can be computed beforehand as follows:

R1) Given a point $x_0 \in int(I)$, (e.g. the centroid of I if I is convex), an analytic expression for $u_{\text{ff}} : \mathbb{R} \to \mathbb{R}^m$ is synthesized using GGGP, by maximizing the approximated robustness measure for a nominal trajectory starting at x_0, i.e. a trajectory with no disturbance:

$$\begin{aligned} \arg\max_{u_{\text{ff}}} \quad & P(\hat{R}_{\mathcal{I}}^{u_{\text{ff}}}, \Psi, 0), \\ \text{subject to} \quad & \mathcal{I} = \{(x_0, \mathbf{0}_l)\}. \end{aligned}$$

R2) Given the feedforward input u_{ff}, an analytic expression for the corresponding reference trajectory $x_{\text{ref}} : \mathbb{R} \to \mathbb{R}^n$ is synthesized using GGGP, by fitting an expression to simulated solution $x_i(\tau_k)$ for $i \in \{1, \ldots, n\}$, based on the Euclidean norm of the error vector $e_i = [e_i(\tau_0), \ldots, e_i(\tau_{\text{f}})]$, with $e_i(\tau_k) = x_i(\tau_k) - x_{\text{ref},i}(\tau_k)$, i.e., maximizing:

$$\arg\max_{x_{\text{ref},i}} (1 - \|e_i\|)^{-1}.$$

Using the synthesized pair $(u_{\text{ff}}(t), x_{\text{ref}}(t))$, the user-defined grammar used within GGGP can be used to enforce the structure of a time-varying reference controller in (5.7) within step A1), as is demonstrated in the case studies in Section 5.7.

5.5. Counterexample generation and verification

In this section we detail step A3) of the algorithm. First, we detail how a (tight) upperbound on the robustness measure is derived. Secondly, if this bound implies that the RTL formula is violated, a corresponding counterexample is extracted. Since an upperbound can only be used to invalidate the controller, we conclude this section by presenting a method to formally verify whether the RTL formula is satisfied.

5.5.1. Robustness measure bounds

In this chapter, we consider polynomial zonotopes $\mathcal{PZ}$ as the set representation of the reachable set:

Definition 5.5.1 (Polynomial zonotope). *Given a generator matrix $G \in \mathbb{R}^{n \times h}$ and exponent matrix $E \in \mathbb{Z}_{\geq 0}^{p \times h}$, a polynomial zonotope $\mathcal{PZ}$ is defined as*

$$\mathcal{PZ} := \left\{ \sum_{i=1}^{h} \left(\Pi_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha_k \in [-1, 1] \right\}.$$

The vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_p]^T$ is referred to as the parameterization vector of the polynomial zonotope.

Remark 5.5.1. *In this definition, without loss of generality and for the ease of exposition, we only consider (dependent) generators G and we omit independent generators; for the full definition we refer to [81].*

The use of polynomial zonotopes is motivated by its useful properties for counterexample generation, discussed in Section 5.5.2. Consider a parameterization vector $\boldsymbol{\alpha}$ and a reachable set $R(t)$ expressed as a polynomial zonotope. The corresponding point in the reachable set $z(\boldsymbol{\alpha}, R(t)) \in \mathbb{R}^n$ is given by:

$$z(\boldsymbol{\alpha}, R(t)) = \sum_{i=1}^{h} \left(\Pi_{k=1}^{p} \alpha_k^{E_{R(k,i)}} \right) G_{R(\cdot,i)}, \tag{5.8}$$

where E_R and G_R denote the exponent matrix and generator matrix of $R(t)$, respectively. Given an RTL formula ψ in the form of (5.3) and a reachable set $R : \mathbb{R}_{\geq 0} \to 2^{\mathbb{R}^n}$, the robustness measure is upperbounded as follows:

B1) For all subformulae Ψ'_{ijk} in (5.3b), the corresponding robustness sub-score (5.4b) is computed by solving the following nonlinear optimization problem over the corresponding set $R(jc/2)$:

$$p^*_{ijk} = \min_{\boldsymbol{\alpha}_{ijk}} \left(\max_{a \in \mathsf{A}^{ijk}} \left(\min_{b \in \mathsf{B}^{ijk}_a} h^{ijk}_{ab}(z(\boldsymbol{\alpha}_{ijk}, R(\tfrac{jc}{2}))) \right) \right). \tag{5.9}$$

B2) Given the robustness sub-scores p^*_{ijk}, compute the full robustness measure (5.4a):

$$p^* = \min_{i \in \mathsf{I}} \max_{j \in \mathsf{J}_i, k \in \mathsf{K}_{ij}} p^*_{ijk}. \tag{5.10}$$

B3) As we rely on nonlinear optimization, we cannot guarantee to find the global optimum p^*, but rather an upperbound $\hat{p}$, such that $P(R, \Psi, 0) = p^* \leq \hat{p}$. Given $\hat{p}$, either:

(a) $\hat{p} < 0$, hence the RTL specification is violated. In this case, given the argument $(ijk)^*$ solving (5.10), we use the parameterization vector $\boldsymbol{\alpha}_{(ijk)^*}$ to extract a counterexample, as described in Section 5.5.2.

(b) $\hat{p} \geq 0$, hence the RTL specification is potentially satisfied. However, to guarantee this, we perform an additional verification step, described in Section 5.5.3.

Remark 5.5.2. *To use gradient-based optimization, the* max *and* min *function can be approximated by the smooth and differentiable function*

$$\underset{a \in \mathsf{A}}{\mathrm{M}_\beta}(x_a) = \frac{\sum_{a \in \mathsf{A}} x_a e^{\beta x_a}}{\sum_{a \in \mathsf{A}} e^{\beta x_a}}, \tag{5.11}$$

where A *denotes an iterator set and for* $\beta \to \infty$, $\mathrm{M}_{\beta\, a \in \mathsf{A}}(x_a) \to \max_{a \in \mathsf{A}} x_a$ *and* $\beta \to -\infty$, $\mathrm{M}_{\beta\, a \in \mathsf{A}}(x_a) \to \min_{a \in \mathsf{A}} x_a$.

5.5.2. Counterexample generation

If the RTL formula is not satisfied, we want to obtain a counterexample, which can be subsequently used to refine the controller design. This counterexample is a pair of initial condition and disturbance realization (x, w), such that the corresponding trajectory results in a violation of the RTL formula. As mentioned before, we use polynomial zonotopes as the set representation of the reachable set. The benefit of polynomial zonotopes as set representation is that dependencies between points in subsequent reachable sets is maintained under the reachability analysis operations [82]. That is, for a reachable set $R : \mathbb{R}_{\geq 0} \to 2^{\mathbb{R}^n}$ and parameterization vector $\boldsymbol{\alpha}$, we have for two time instances t and τ:

$$\xi(t) = z(\boldsymbol{\alpha}, R(t)) \implies \xi(\tau) = z(\boldsymbol{\alpha}, R(\tau)). \tag{5.12}$$

A counterexample corresponding to the optimization steps in B1) and B2) is extracted as follows. Given the reachable set R, the argument $(ijk)^*$ solving (5.10) in step B2) and the corresponding parameterization vector $\boldsymbol{\alpha}_{(ijk)^*}$ from (5.9) in step B1), the corresponding initial condition is given by $\xi(0) = z(\boldsymbol{\alpha}_{(ijk)^*}, R(0))$. This procedure based on polynomial zonotopes is described in exact detail in [82]. For this counterexample $x = \xi(0)$, a

disturbance realization w is optimized similarly to step A1.b), i.e., GGGP is used to solve the following problem:

$$\arg\max_{w} -P\left(\hat{R}^{\kappa}_{\{(x,w)\}}, \Psi, 0\right).$$

The pair (x, w) is subsequently added to $\mathcal{I}$. This new set $\mathcal{I}$ is then used to improve upon the synthesized controller in step A1).

5.5.3. Verification

Computing the robustness degree (5.4) for general formulae results in nonlinear optimization problems, originating from the set representation of $R(t)$ and the function $h^{ijk}_{ab} : \mathbb{R}^n \to \mathbb{R}$ (see also Section 5.5.2). Therefore, instead of exactly computing the robustness degree in order to formally verify the RTL formula, we employ Satisfiability Modulo Theories (SMT) solvers [15], which are capable of verifying first-order logic formulae. The subformula (5.3b) holds if the following first-order logic formula holds:

$$\forall x \in R\left(j\frac{c}{2}\right) : \bigvee_{a \in \mathsf{A}^{ijk}} \bigwedge_{b \in \mathsf{B}^{ijk}_{a}} h^{ijk}_{ab}(x) \sim 0, \tag{5.13}$$

where again $\sim \in \{\geq, >\}$. Suitable SMT solvers to verify (5.13) include Z3 [32] when $R(jc/2)$ and h^{ijk}_{ab} are expressed as polynomials, and dReal [51] when these are expressed as general nonlinear expressions [2]. Given the Boolean answer to the subformula (5.3b) for all ijk, it is trivial to compute the Boolean answer to (5.3a).

5.6. Dealing with conservatism

Due to both the conservatism in the reachability analysis and the transformation from STL to RTL, it is possible that $(R, 0) \not\models \Psi$, whereas $\forall \xi(0) \in I$, $(\xi, 0) \models \varphi$, i.e., the desired STL specification holds for all initial conditions, whereas based on the reachability set, the RTL specification is not met. To counter this, the reachability analysis can be made less conservative by refining settings such as the time steps or Taylor order (see [7]). Secondly, the transformation Υ could be performed for a smaller time-discretization parameter c to obtain a less conservative RTL formula Ψ.

Issues due to conservatism can also be dealt with within the synthesis of a candidate controller in step A1). For example, the population of controllers within GGGP could be further optimized w.r.t. the robustness measure approximation, such that the added robustness could potentially compensate for the conservatism within the reachability analysis. Additionally, controllers can be optimized with respect to both robustness measure and complexity, as less complex controllers might result in less conservatism within the reachability analysis. This results in a multi-objective optimization problem. In this work we consider the fitness criteria for complexity to be defined as the the number of non-terminals of an individual. We use the non-dominated sorting algorithm NSGA-II [33], a Pareto optimal-aware sorting algorithm, which ranks candidate controllers based on the Pareto optimality of both fitness criteria. This rank is then used as fitness value within

[2] Recall that dReal implements a δ-complete decision procedure [50]. However, if the reachable set is robust w.r.t. the RTL formula, this caveat has no consequence.

Table 5.1: General settings for each of the case studies. The number of individuals, GGGP generations and CMA-ES generations are shown for each controller component and disturbance realizations.

System	n_s	Individuals				GGGP generations				CMA-ES generations			
		u_{ff}	x_{ref}	κ	w^i	u_{ff}	x_{ref}	κ	w^i	u_{ff}	x_{ref}	κ	w^i
Car	7	14	14	14	14	30	10	3	3	20	10	10	3
Path planning	10	28	28	14	14	30	50	3	3	40	40	10	3
Aircraft	5	28	42	14	14	50	50	5	5	40	60	10	3

Table 5.2: Production rules $\mathcal{P}$.

$\mathcal{N}$	Rules
$\langle \text{expr} \rangle$	$::= \langle \text{pol} \rangle \mid \langle \text{pol} \rangle \times \langle \text{trig} \rangle \mid \langle \text{expr} \rangle + \langle \text{expr} \rangle$
$\langle \text{trig} \rangle$	$::= \tanh(\langle \text{pol} \rangle) \mid \sin(\langle \text{pol} \rangle) \mid \cos(\langle \text{pol} \rangle)$
$\langle \text{pol} \rangle$	$::= 0 \mid \langle \text{const} \rangle \mid \langle \text{const} \rangle \times \langle \text{mon} \rangle \mid \langle \text{pol} \rangle + \langle \text{pol} \rangle$
$\langle \text{mon} \rangle$	$::= t \mid t \times \langle \text{mon} \rangle$
$\langle \text{const} \rangle$	$::=$ Random Real $\in [-1, 1]$

the selection. To make sure the controller with the best robustness measure is always maintained within the population, this controller is always directly copied into the new generation.

Finally, there is a gap between the approximated reachable set $\hat{R}^{\kappa}_{\mathcal{I}}$ and the reachability analysis. There are two sources that can cause significant mismatches between this approximation and actual robustness measure. The first source is truncation errors of the integration scheme. Secondly, due to the added conservatism within reachability analysis, the reachable set can contain additional trajectories which are not admitted by the original system. To bridge this mismatch, we can consider an optional error signal ε added to the simulated trajectory $x(\tau_q)$, which is co-synthesized with the disturbance realizations, as will be shown in the case studies in the next section.

5.7. Case studies

In this section we demonstrate the effectiveness of the proposed framework on a car benchmark, path planning problem and aircraft landing manoeuvre. The case studies are performed this section is performed using an Intel Xeon CPU E5-1660 v3 3.00GHz using 14 parallel CPU cores. The GGGP and CMA-ES algorithms are both implemented in Mathematica 12 and the reachability is performed using CORA in MATLAB. For the nonlinear optimization and verification in Section 5.5, we use particle swarm optimization of the global optimization toolbox in MATLAB, and the SMT solver dReal with $\delta = 0.001$, respectively.

Across all benchmarks, the probability rate of the crossover and mutation operators being applied on a selected individual are 0.2 and 0.8, respectively. Each generation, parameters within an individual are optimized using CMA-ES. Benchmark-specific settings are shown in Table 5.1, which include the number of simulations n_s, number of individu-

als, and the number of GGGP and CMA-ES generations. Note that the number of GGGP generations for κ and w^i is the number of generations per step A1.a) and A1.b), and not the total of GGGP generations per proposal of a controller in step A1), which depends on the number of times step A1.a) and A1.b) are repeated. For each case study, we use a grammar with nonterminals and production rules as shown in Table 5.2. These nonterminals correspond to general expressions $\langle \text{expr} \rangle$, trigonometric functions $\langle \text{trig} \rangle$, polynomial expression $\langle \text{pol} \rangle$, monomials $\langle \text{t} \rangle$ and constants $\langle \text{const} \rangle$. The expressions are formed by polynomials, a product of a polynomial and trigonometric functions, and a sum of two expressions. The trigonometric functions are restricted to hyperbolic tangents, sines and cosines with polynomial arguments. The polynomials are restricted to polynomials over time t. Note that per case study, different starting trees are used, such that potentially only a subset of the grammar is available. E.g., if the starting tree is $\langle \text{pol} \rangle$, candidate solutions are restricted to polynomial solutions.

We use Runge-Kutta as numerical integration scheme. To keep a constant number of initial conditions in $\mathcal{I}$, counterexamples are added using a first-in, first-out principle. To compensate for the gap between the simulation and the reachability analysis (as discussed in Section 5.6), we consider an added error signal bounded by the scaled vector field of the dynamics f, parameterized by

$$\varepsilon(t, x) = \delta\sigma(t) f(t, x(t), u(t), w(t)), \tag{5.14}$$

where δ is a constant and $\sigma : \mathbb{R}_{\geq 0} \to [-1, 1]^{n \times n}$ a time-varying diagonal matrix which determines the sign and magnitude of the error signal. The constant δ is optimized after each reachability analysis such that the mismatch between the robustness measure and the approximated robustness measure is minimized, i.e.:

$$\arg\min_{\delta} \left\| P(R, \Psi, 0) - P\left(\hat{R}^{\kappa}_{\{(x,w)\}}, \Psi, 0\right) \right\|, \tag{5.15}$$

where $\{(x, w)\}$ is the counterexample pair computed in Section 5.5.2.

Finally, in reporting the synthesized controllers, its parameters are rounded from six to three significant numbers for space considerations.

5.7.1. Car benchmark

Let us consider a kinematic model of a car from [138]:

$$\begin{cases} f(x, u, w) = (u_1 + w_1, u_2 + w_1, x_1 \cos(x_2), x_1 \sin(x_2)^T, \\ I = [19.9, 20.2] \times [-0.02, 0.02] \times [-0.2, 0.2]^2, \\ \Omega = [-0.5, 0.5] \times [-0.02, 0.02]. \end{cases}$$

where the states x_1, x_2, x_3, x_4 denote the velocity, orientation, and x and y position of the car, respectively. Furthermore, u_1 and u_2 denote the inputs and w_1 and w_2 disturbances. The sampling time of the sampled-data controller is set to be 0.025 seconds. Similarly to [138], we consider a "turn left" maneuver over a time interval $T = [0, 1]$, where within T, the trajectories stay within the safe set S and at the final time instant, the system is in the goal set, captured by the STL specification:

$$\varphi_1 = \square_{[0,1]} \varphi_S \wedge \square_{\{1\}} \varphi_O, \tag{5.16}$$

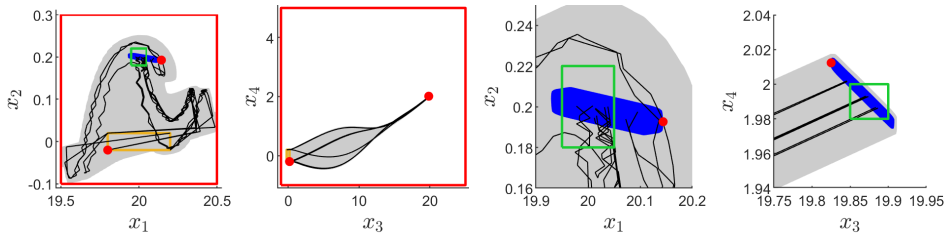


Figure 5.3: Reachable set for the first controller for the car benchmark, which violates the desired controller specification. Figures (c) and (d) illustrate the reachable set near the goal set. Red dots: a point in the final reachable set that is outside of the goal set and it's corresponding initial state, yellow: initial set, green: goal set G, gray: reachable set, red: safe set S, blue: reachable set at $t = 1$, black: example of simulation traces.

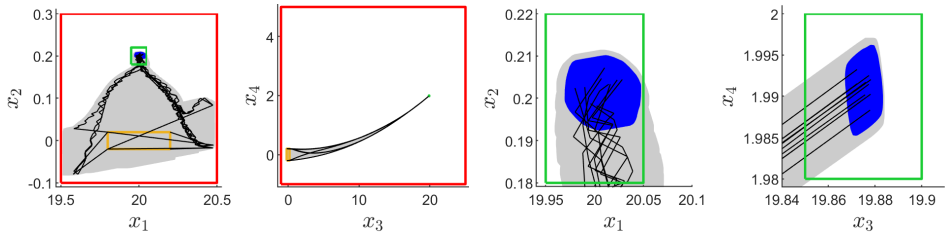


Figure 5.4: Reachable set for the final controller for the car benchmark, which formally satisfies the desired controller specification. Figures (c) and (d) illustrate the reachable set near the goal set. Sets and simulation traces are indicated as in Figure 5.3.

where φ_S, φ_O denote the logic formulae capturing the set membership of S and O (see Section 2.3). We consider the following safe set S and goal set O:

$$S = [19.5, 20.5] \times [-0.1, 0.3] \times [-1, 25] \times [-1, 5],$$
$$O = [19.95, 20.05] \times [0.18, 0.22] \times [19.85, 19.9] \times [1.98, 2].$$

To guide the synthesis, we impose the reference-tracking controller structure from Section 5.4.2 and therefore we first design a feedforward signal and reference trajectory using GGGP. For $u_{\text{ff}}, x_{\text{ref}}$, we use polynomial expressions as a function of time t, for the feedback law κ we restrict the search space to reference-tracking controllers which are linear in the tracking error and polynomial in time:

$$\kappa(x, t) = u_{\text{ff}}(t) + K(t)(x - x_{\text{ref}}), \tag{5.17}$$

and for w^i we consider saturated polynomials in time. This is done using the grammar with starting trees:

$$\begin{aligned}\mathcal{S}_{u_{\text{ff}}} &= (\langle \text{pol} \rangle, \langle \text{pol} \rangle)^T,\ \mathcal{S}_{x_{\text{ref},i}} = \langle \text{pol} \rangle,\\ \mathcal{S}_{\kappa} &= u_{\text{ff}} + \begin{pmatrix} \langle \text{pol} \rangle, \dots, \langle \text{pol} \rangle \\ \langle \text{pol} \rangle, \dots, \langle \text{pol} \rangle \end{pmatrix} (x - x_{\text{ref}}),\\ \mathcal{S}_{w^i} &= \left(\text{sat}_{(\underline{\omega}_1, \overline{\omega}_1)}(\langle \text{pol} \rangle), \text{sat}_{(\underline{\omega}_2, \overline{\omega}_2)}(\langle \text{pol} \rangle)\right)^T.\end{aligned}$$

Here, $\text{sat}_{(\underline{\omega}_i, \overline{\omega}_i)}$ denotes a saturation function such that $w^i(t) \in \Omega$, where

$$\text{sat}_{(\underline{\omega}_i, \overline{\omega}_i)}(x) = \max(\underline{\omega}_i, \min(x, \overline{\omega}_i)). \tag{5.18}$$

Finally, for each disturbance realization, we co-evolve the error signal ε^i in (5.14), which is dependent on the candidate controller κ and disturbance realization w^i:

$$\begin{aligned}\mathcal{S}_{\varepsilon^i} &= \delta\sigma f(t, x, \kappa(x), w^i),\\ \sigma &= \text{diag}(\text{sat}_{(-1,1)}(\langle \text{pol} \rangle), \dots, \text{sat}_{(-1,1)}(\langle \text{pol} \rangle)),\end{aligned}$$

where diag denotes a diagonal matrix. For the simulations and reachability analysis, we use a sampling time of 0.025 seconds and 0.0125 seconds, respectively.

First, a feedforward control input and reference trajectory for a nominal initial condition are synthesized as described in Section 5.4.2. An example of a found feedforward controller and corresponding reference trajectory are shown in Table 5.5. For 10 independent runs, the average synthesis times of u_{ff} and the reference trajectory per dimension $x_{\text{ref},i}$ are shown in Table 5.4. Using these u_{ff} and x_{ref} as building blocks for the controller, κ is synthesized as described in step A1). An example of a synthesized $K(t)$ in (5.17) is given by

$$K(t) = \begin{pmatrix} -41.5 & -6.48t^2 & -84.3958 & 9.45 \\ 3.58 & -30.1 & -8.22 & 3.62t1 - 49.2t^2 \end{pmatrix}.$$

The corresponding reachable set is shown in Figure 5.3. We observe that the final reachable set is not within the goal set. The red dots represent the violation and the corresponding initial condition. For this violation x, a disturbance realization w is optimized and the pair (x, w) is added to $\mathcal{I}$ and fed back to the GGGP algorithm. After refining the controller iteratively, an example of a controller satisfying φ_1 after 3 refinements is shown in Table 5.5. The corresponding reachability analysis is shown in Figure 5.4 and it shows that for this controller the controller specification is formally met.

For 10 independent synthesis runs of κ, the statistics on the number of generations, number of refinements, complexity in terms of number of nonterminals, and computation time is shown in Tables 5.3 and 5.4, and Figure 5.8. In most cases, a solution was obtained around 3 refinements. However, due to the stochastic nature of the approach, in one case it took 20 refinements before a solution was found.

5.7.2. INPUT SATURATION

In our general framework, we do not canonically consider input saturation. Input saturation can be considered in multiple ways, such as restricting the grammar of the controller

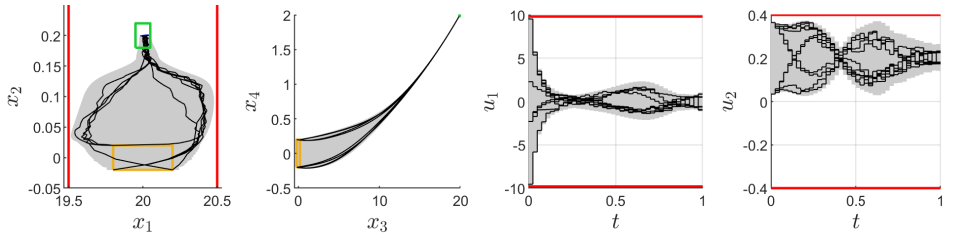


Figure 5.5: Reachable set of the controller for the car benchmark under input constraints. Figure (c) and (d) show the reachable set of the input over time. Sets and simulation traces are indicated as in Figure 5.3.

to include a saturation function, or even a continuous approximation using e.g. a sigmoid function. However, the downside of such an approach is that the reachability analysis under these functions is typically challenging for state-of-the-art reachability tools, due to the strong nonlinearity or hybrid nature. Instead, for illustrative purposes, we incorporate the constraint within the STL specification, such that for all states in the reachable set the saturation bounds are not exceeded. Let us revisit the car benchmark, where we consider the same input constraints as in [138], namely $u \in \overline{U} = [-9.81, 9.81] \times [-0.4, 0.4]$. The STL specification is extended to:

$$\varphi_2 = \varphi_1 \wedge \square_{[0,1]} \varphi_U \tag{5.19}$$

with

$$U = \left\{ x \in \mathbb{R}^n \mid \kappa(x) \in \overline{U} \right\}. \tag{5.20}$$

The synthesis statistics are shown in Tables 5.3 and 5.4, and Figure 5.8. An example of a synthesized $K(t)$ in (5.7) is given by

$$K(t) = \begin{pmatrix} -18.1 + 18.2t - 65.9t^6 & 0.22t \\ 0 & -8.26 - 41.8t \\ -29.6 - 48.7t & 0 \\ -11.2t & -33.1t^2 \end{pmatrix}^T.$$

The corresponding reachability set is shown in Figure 5.5. In most cases, a solution was found in around 4 to 5 refinements, with the exceptions of two runs with 20 and 40 refinements, respectively.

5.7.3. Path planning

Let us consider the path planning problem for a simple robot adopted from [95]. We deviate from [95] in considering the system in continuous time and consider bounded disturbances. The system is described by:

$$\begin{cases} f(x, u, w) = (u_1 + w_1, u_2 + w_2, x_1, x_2)^T, \\ I = \{0\}^2 \times [0.5, 1.5]^2, \\ \Omega = [-0.05, 0.05]^2, \end{cases}$$

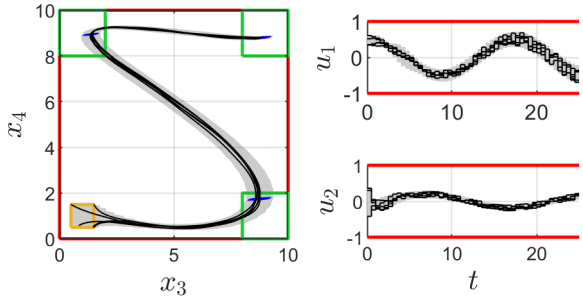


Figure 5.6: Reachable set of a found controller for the path planning benchmark. (a) Reachable set of the x-y position. (b) Reachable set of the input over time. Yellow: initial set, gray: reachable set, red: safe set S and input constraints, green: target sets P_1, P_2, and P_3, black: selection of simulated trajectories, blue: reachable sets at certain time instances within one of the target sets.

where the state vector represents the x-velocity, y-velocity, x-position and y-position, respectively. The sampling time of the sampled-data controller is chosen to be 0.5 seconds. Similar to [95], we consider the specification in which the system needs to remain in a safe set S and eventually visit regions P_1, P_2 and P_3:

$$\varphi' = \square_{[0,25]} \Psi_S \wedge \Diamond_{[5,25]} \Psi_{P_1} \wedge \Diamond_{[5,25]} \Psi_{P_2} \wedge \Diamond_{[5,25]} \Psi_{P_3}. \tag{5.21}$$

with $S = \{x \in \mathbb{R}^n \mid (x_3, x_4) \in [0,10]^2\}$, $P_1 = \{x \in \mathbb{R}^n \mid (x_3, x_4) \in [8,10]^2\}$, $P_2 = \{x \in \mathbb{R}^n \mid (x_3, x_4) \in [8,10] \times [0,2]\}$, $P_3 = \{x \in \mathbb{R}^n \mid (x_3, x_4) \in [0,2] \times [8,10]\}$. In [95], the input is constrained s.t. $u \in \overline{U} = [-1,1]^2$. Similar to Section 5.7.2, we impose this constraint through the STL specification, yielding the following STL specification:

$$\varphi = \varphi' \wedge \square_{[0,25]} \varphi_U, \tag{5.22}$$

where U is given by (5.20). We consider the same controller structure and grammar as the previous benchmark, with the exception of the grammar of the feedfoward input and reference trajectory. For these elements, we extend the grammar to expressions which can include trigonometric functions, by using the grammar in Table 5.2 and the following starting trees $\mathcal{S}_{u_{\text{ff}}} = (\langle \text{expr} \rangle, \langle \text{expr} \rangle)$ and $\mathcal{S}_{x_{\text{ref},i}} = \langle \text{expr} \rangle$. For the simulations and reachability analysis, we use a sampling time of 0.5 seconds. The statistics on the synthesis is again shown in Tables 5.3 and 5.4. An example of the controller elements u_{ff}, x_{ref} and $K(t)$ of a synthesized controller are shown in Table 5.5 and Figure 5.8. The corresponding reachable set of the state and input is shown in Figure 5.6. Across 10 independent runs, commonly in 1 to 2 refinements a solution was found, with one run requiring 8 refinements.

5.7.4. Landing maneuver

Let us consider the landing aircraft maneuver, adopted from [124]. The system model is given by

$$\begin{cases} f(x,\nu,w) = \begin{pmatrix} \frac{1}{m}(\nu_1 \cos\nu_2 - D(\nu_2,x_1) - mg\sin x_2) \\ \frac{1}{mx_1}(\nu_1 \sin\nu_2 + L_2(\nu_2,x_1) - mg\cos x_2) \\ x_1 \sin x_2 \end{pmatrix}, \\ D(\nu_2,x_1) = (2.7 + 3.08(1.15 + 4.2\nu_2)^2)x_1^2, \\ L(\nu_2,x_1) = (68.6(1.25 + 4.2\nu_2))x_1^2, \\ \nu_i = u_i + \omega_i, \;\; i = 1,2, \\ I = [80,82] \times [-2°,-1°] \times \{55\} \\ \Omega = [-5\cdot 10^3, -5\cdot 10^3] \times [-0.25°, 0.25°], \end{cases}$$

where the states x_1, x_2, x_3 denote the velocity, flight path angle and the altitude of the aircraft, ν_i denotes a disturbed input, where u_1 denotes the thrust of the engines and u_2 the angle of attack. Finally, $D(\nu, x_1)$ and $L(\nu, x_1)$ denote the lift and drag, respectively, and $m = 60 \cdot 10^3$ kg, $g = 9.81\text{m/s}^2$. The sampling time of the sampled-data controller is set at $\eta = 0.25$ seconds. Compared to [124], we do not consider measurement errors, but the proposed framework can be adapted arbitrarily to accommodate this type of disturbance. We define the following safe set, goal set and input bounds:

$$\begin{aligned} S =& [58,83] \times [-3°, 0°] \times [0, 56], \\ G =& [63,75] \times ([-2°,-1°] \times [0, 2.5]) \\ & \cap \{x \in \mathbb{R}^3 \mid x_1 \sin x_2 \geq -0.91\}, \\ \overline{U} =& [0, 160 \cdot 10^3] \times [0°, 10°] \end{aligned}$$

and consider the following specification:

$$\varphi = (\varphi_S \wedge \varphi_U)\mathcal{U}_{[18,20]}\varphi_G, \tag{5.23}$$

where the set U is given by (5.20). That is, trajectories are always within the safe set and satisfy the input constraints, until between 18 and 20 seconds the goal set is reached.

We use the same controller structure and grammar as the path planning problem. For the simulations and reachability analysis, we use a sampling time of 0.25 seconds. The algorithm settings are shown in Table 5.1. The statistics of 10 independent synthesis runs are again shown in Tables 5.3 and 5.4, and Figure 5.8. An example of the controller elements u_{ff}, x_{ref} and $K(t)$ of a synthesized controller are shown in Table 5.5. The corresponding reachable set of the altitude over time, as well as the reachable sets of the pitch angles at multiple time instances are shown in Figure 5.7.

5.8. Discussion

In this section we discuss the main results from Section 5.7 and compare them to the results in the literature. Recall that a GGGP *generation* is the cycle of creating a new population through fitness evaluation, selection and applying genetic operators. A *refinement* is defined as the cycle of proposing a candidate solution based on GGGP, validation using reachability analysis, and extracting counterexamples. Therefore, in each refinement,

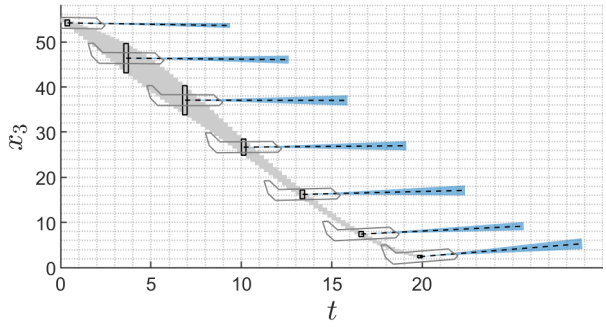


Figure 5.7: Time evolution of the reachable set of the altitude x_3 under a synthesized controller for the landing maneuver. Gray: Reachable set over time of the altitude x_3. Blue: the set of the aircraft pitch $x_2 + u_2$ for 8 time intervals.

Table 5.3: Statistics over an average of 10 independent synthesis runs. Total gen.: total number of GGGP generations for κ before a solution was found; Total ref.: total number of refinements; Complexity: number of total nonterminals within the genotype of the synthesized controller; min: minimum; med: median; max: maximum.

System	Total gen.			Total ref.			Complexity		
	min	med	max	min	med	max	min	med	max
Car	63	205.5	1410	3	6	19	14	27	69
Constrained car	84	318	933	2	5	8	24	35.5	56
Path planning	3	16.5	117	1	2.5	9	8	11.5	15
Aircraft	45	342.5	1165	2	5	16	24	36	58

there are one or multiple GGGP generations. First of all, Figure 5.8a shows a polynomial relation between the number of refinements and the total number of GGGP generations. Secondly, Figure 5.8b shows a polynomial relation between the number of refinements versus the total computation time. Finally, Figure 5.8c illustrates that more refinements does not imply that complexity of the controller increases. However, the complexity of the found controller does seem to be dependent on the system and STL specification.

While the computation time is related to the number of refinements, this relationship depends on the STL specification and the dynamics. For the car benchmark without and with input constraints, we observe that the added constraints within the STL specification increased the required number of generations, and typically required more time per refinement. Hence, the total computation time heavily depends on the STL specification, as expected. Additionally, we observe an increase in the median of the complexity of the resulting controllers. The input-constrained car and path planning benchmarks are both four-dimensional systems, where the STL specification of the latter is more involved. Regardless, the path planning problem has a lower computation time and requires less generations and number of refinements, indicating a dependency between the computation time and the dynamics of the system, which is also as expected.

In [138], the synthesis time for the car benchmark is around 10 seconds, which is significantly shorter than the synthesis time of the proposed framework. The resulting con-

Table 5.4: Time statistics over an average of 10 independent synthesis runs. Time FF: average computation time of the feedforward components; Time: total time of the controller synthesis (excluding the feedforward synthesis); GP κ: synthesis of candidate κ using GGGP; GP ω: disturbance realization optimization; RA: reachability analysis; CE: counterexample extraction; SMT: verifying the specification through an SMT solver; min: minimum; med: median, max: maximum. The average contribution percentages do not sum up to one, as the contribution of routines such as writing (SMT) files are not displayed.

System	Time FF [s]		Time [min]			Average contribution to total time [%]				
	u_{ff}	$x_{\text{ref},i}$	min	med	max	GP κ	GP ω	RA	CE	SMT
Car	45.1	1.2	16.5	41.6	204.1	37.9	26.2	3.15	19.3	3.44
Constrained car	-	-	28.0	61.2	117.0	42.5	17.2	1.70	15.8	9.19
Path planning	254.0	19.1	14.1	23.8	61.8	7.61	9.50	3.05	17.2	27.8
Aircraft	708.2	46.2	44.0	165.1	422.8	36.7	22.5	12.9	10.29	7.71

Table 5.5: Examples of synthesized controllers. Numerical values are rounded for space considerations.

System	Car (unconstrainted)	Path planning	Aircraft
u_{ff}	$\begin{pmatrix} 0.01835 \\ 0.1995 \end{pmatrix}$	$\begin{pmatrix} 0.500\cos(0.362t + 0.0733) \\ -0.190\sin(0.678\ -0.324t) \end{pmatrix}$	$\begin{pmatrix} 255.68 + 107.57t^2 \\ 0.00956 + 0.00419t \end{pmatrix}$
x_{ref}	$\begin{pmatrix} 19.999 + 0.020567t \\ 0.19954t \\ 19.981t - 0.10838t^4 \\ 1.9915t^2. \end{pmatrix}$	$\begin{pmatrix} 0.03t - 3.81\cos(0.361t) + 4.72 \\ 1.38\sin(0.361t) + 0.024 \\ 0.406t + 1.88\cos(0.312t + 0.949), \\ 0.427 - 0.583\cos(0.765\ -0.325t) \end{pmatrix}$	$\begin{pmatrix} 81.5 - 0.380t - 1.28\sin(0.393 + 0.164t) \\ (-0.164 - 1.59 \cdot 10^{-3}t)\cos(0.103t) + 0.138\cos(0.120t) \\ 55.7 - 0.674\cos(0.354t) - 2.96t\sin(0.788 + 0.062t) \end{pmatrix}$
$K(t)$	$\begin{pmatrix} -43.4 & -8.28t^5 \\ 3.94 & -33.3 \\ -89.6 & -6.21 \\ 307.3t^2 & -10.1 \end{pmatrix}^T$	$\begin{pmatrix} -0.264 & 0 \\ -0.125t & -0.204 \\ 0 & -0.781 \\ 0.209t & -1.35 \end{pmatrix}^T$	$\begin{pmatrix} -2.67t^3 & -0.00607 \\ -0.407 - 0.0636t & -0.0217 - 0.237t - 0.0348t^2 \\ -0.788 - 0.461t & -0.00023t^2 \end{pmatrix}^T$

troller consists of a linear controller for each sampling time, resulting in 10 controllers in total. For longer time horizons and/or finer time discretization, this number of controllers increases. On the other hand, our method is able to find a single controller, independent of the sampling time.

Comparing the proposed framework to MPC approaches, such as the approach used for the path planning problem in [95], we obtain a closed-form controller for which the STL specification is guaranteed, whereas MPC-based approaches require online optimization to compute the controller input.

For the aircraft benchmark, the abstraction-based method in [124] yields a controller which can be seen as a look-up table, in which the state space is partitioned into over 2.26 million states. For each region within this partition, a finite set of admissible inputs are stored, resulting in a nondeterministic controller. This controller is stored as a binary decision diagram (BDD) with the size of 2.87 MB and can be further reduced to 0,15 MB by removing nondeterminism (see [170]). Additionally, to parse the BDD format, special libraries are required. On the other hand, our controller can be stored in a text file within 916 bytes and is simply expressed as single analytic function. The synthesis time in [124] is 674 seconds for the abstraction and 26 seconds for the controller synthesis, which is again significantly shorter than the presented framework. However, one could leverage that storage space in embedded hardware is finite, whereas bounds on offline computation time are typically less restrictive. Moreover, the abstraction yields a finite transition system with $9.38 \cdot 10^9$ transitions. For higher dimensional systems, due to the curse of

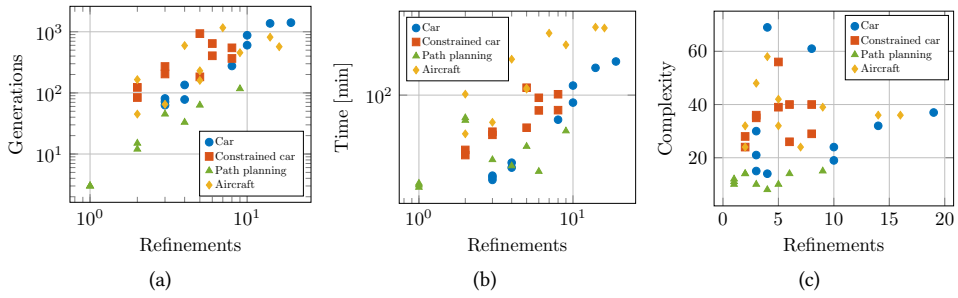


Figure 5.8: Number of refinements versus (a) number of GGGP generations, (b) time in minutes, and (c) complexity of the controller, measured in number of nonterminals.

dimensionality, the platform used to synthesize the controller can run into memory constraints. While the proposed framework in this work is computationally intensive, it does not suffer from the same curse of dimensionality w.r.t. memory constraints. Additionally, our framework is capable of guaranteeing the specification over continuous-time trajectories, as opposed to over discrete-time trajectories, as is done in [124].

Relying on GP, the proposed framework is not a complete method. That is, the method is not guaranteed to find a solution in a finite number of iterations, regardless of its existence. Nevertheless, for the presented case studies, in 10 independent runs a solution was always found. Since the search space is navigated nondeterministically, we observed that the number of GGGP generations, number of refinements and computation time can vary significantly for each run.

Across all benchmarks, the offline computation time for the proposed method is significantly larger than corresponding references. However, there are several elements in which the computation time can be improved. First of all, GGGP and the reachability tool are implemented in Mathematica and Matlab. By implementing these elements in lower-level programming languages, a speed-up is expected. Analyzing the contribution to the total computation time in Table 5.3, we observe that, in general, GGGP takes up the majority of the computation time. GGGP is highly parallelizable and is in this work not fully exploited, as we only consider 14 individuals, matching the number of used processor cores. By further exploiting the parallelizable nature of GGGP and therefore exploring a larger part of the search space each generation, a significant speed-up is expected. Thirdly, by limiting to a fragment of STL, e.g., by restricting $h(s)$ to be linear, computing the robustness degree can be simplified and therefore improve the computation time of counterexamples. If additionally the robustness measure is upper bounded in a non-conservative manner, the use of SMT solvers becomes redundant. This would significantly reduce the computation time for benchmarks such as the path planning problem. Finally, we imposed input constraints through the STL specification. By using saturation functions in our grammar, the input constraints are satisfied by definition, simplifying the synthesis. However, as caveat, discontinuous functions such as saturation functions significantly complicate the reachability analysis.

5.9. Conclusion

We have proposed a framework for CEGIS-based correct-by-construction controller synthesis for STL specifications based on reachability analysis and GGGP. The effectiveness has been demonstrated based on a selection of case studies. While the synthesis time is significantly longer compared to the methods in e.g. [138] and [124], the proposed method results in a compact closed-form analytic controller which is provably correct when implemented in a sampled-data fashion. This enables the implementation in embedded hardware with limited memory and computational resources.

6

Discussion

In this chapter we discuss the main differences between the methodologies based on certificate functions and reachability analysis from Chapter 4 and 5, respectively. Additionally, we discuss how both methods can be extended to cover a wider class of systems or specifications in Sections 6.2 and 6.3.

6.1. Comparison

In this section we compare the two approaches: based on certificate functions and employing reachability analysis, in terms of scalability with respect to the computational cost, initial set size, system dimensions, and the capability to deal with input saturation and general hybrid effects, and finally the class of specifications the method addresses.

Computational cost First, let us consider the computational complexity of the fitness function for a reach-while-stay specification w.r.t. a safe set S and goal set O that are expressed the form:

$$Y = \left\{ s \in \mathbb{R}^n \;\middle|\; \bigwedge\nolimits_{i=1}^{n_Y} b_{Y,i}(s) \leq 0 \right\}, \tag{6.1}$$

where $Y \in \{S, O\}$, $n_Y > 0$ and $b_{Y,i} : \mathbb{R}^n \to \mathbb{R}$ for all $i \in \{1, \dots n_Y\}$.

For the certificate-based approach, let us consider the reach-avoid specification $\varphi = \varphi_S \mathcal{U}_{[0,\infty)} \varphi_O$. The sample-based fitness is computed by computing for a number of state-space samples a satisfaction measure (4.11) for each of the LBF conditions. For the reach-avoid specification, the worst-case complexity of (4.11) is $O(n_S)$, where n_S denotes the number of inequalities used to define the safe set in (6.1). As a result, the complexity of the sample-based fitness is $O(n_S n_{\text{samp}})$, where n_{samp} denotes the total number of state-space samples considered. For purely continuous-time systems (see Section 4.8) the complexity is $O(n_{\text{samp}})$ and for the specialized framework for continuous-time systems with sampled data controllers (see Section 4.7.2) the complexity is $O(n_q n_{\text{samp}})$, where n_q denotes the number of controller modes.

For the reachability-based approach, let us consider the reach-avoid specification $\varphi = \square_{[0,T]}\varphi_S \wedge \square_{\{T\}}\varphi_O$[1]. For this STL specification, the RTL formula is given by $\Psi = \bigwedge_{i\in \mathsf{I}} \bigcirc_{\frac{c}{2}} \mathcal{A}\psi_S \wedge \bigcirc_T \mathcal{A}\psi_O$, where $\mathsf{I} = \{0, 1, \ldots, 2T/c\}$, $\psi_S = \varphi_S$ and $\psi_O = \varphi_O$[2]. The corresponding robustness measure is given by

$$P(R, \Psi, 0) = \min\left(\min_{i\in \mathsf{I}}\left(\min_{x\in R(i)} \rho(x, \psi_S)\right), \min_{x\in R(T)}(\rho(x, \psi_O))\right). \tag{6.2}$$

The fitness is computed by performing n_s numerical simulations to form an approximated reachability set $\hat{R}$ and computing the corresponding robustness measure. For the simulations, assuming a Runge-Kutta numerical integration scheme with a time step of $c/2$, the complexity is $O(\frac{T}{c}n_s)$. Let us assume the complexity of the $\max$ and $\min$ operators to be $O(n)$, where n denotes the number of function arguments. The complexity of $P(\hat{R}, \Psi, 0)$ is $O(\frac{T}{c}n_s n_S)$, where we assumed $\frac{T}{c}n_s n_S \gg n_s n_O$.

To summarize, the complexity of the certificate-based approach scales w.r.t. the number of considered state-space samples and, depending of the class of system, the definition of the safe set and the number of controller modes. On the other hand, the complexity of the reachability-based approach scales with the number of simulated trajectories, the sampling time, the considered time horizon, and the definition of the considered sets. Whereas the certificate-based approach is completely independent of the time, the complexity of the reachability approach scales with the time horizon and the time step c.

In terms of verification, for the the certificate-based approach the verification solely relies on SMT solvers, whereas for the reachability-based approach it relies on both reachability analysis and SMT solvers. For the certificate-based approach, the number of inequalities to verify with an SMT solver is dependent on the specification, i.e. reach-while-stay or reach-and-stay-while-stay, whereas for the reachability-based approach, the number of inequalities to verify with an SMT solver depends on both the specification and the sampling time chosen for the reachability analysis.

Finally, let us discuss the total computation time. For the certificate-based approach, the total computation time depends on the number of considered state-space samples, but also the system dynamics and system order. For the reachability-based approach, the computation time depends on the number of simulated trajectories, numerical integration scheme and its step size, but also the STL specification and system dynamics. However, for both methods, more experiments and analysis are required to solidify claims regarding the relation between the computation time and these factors.

Scalability w.r.t. initial set For the certificate-based approach, the size of the initial set is of limited influence. However, for better convergence, more test samples of the initial set could be considered. On the other hand, for the reachability-based approach, the impact of the initial set's size is significant. First of all, to cover the full range of representable system behaviors starting in the initial set, a higher number of simulations can be considered. While an added sampled state in the certificate-based approach corresponds to a single additional function evaluation, the additional simulation in the reachability-based

[1] Alternatively we could consider $\varphi = \varphi_S \mathcal{U}_{[0,T]}\varphi_O$, but its RTL formula is less practical for illustrative purposes.
[2] Note that this RTL formula is not expressed in the standard form in (2.11)

approach has a significantly higher computation cost. Secondly, the increased size of the initial set can also have negative consequences for the reachability analysis. That is, in the employed reachability tool CORA [8], a larger initial set is likely to either introduce additional conservatism, as the abstraction errors in the reachability analysis increase with larger sets, or the problem is split up into several reachability problems, increasing the computation cost.

Scalability w.r.t. system dimension For the certificate-based approach, we have seen in Section 4.6.7 that higher system dimensions correlate with longer computation times. This is expected, as the complexity of the co-synthesized certificate functions significantly increases for higher-dimensional systems and more samples are required to cover the sets. Additionally, in some cases no solution was found. However, this can partly be countered by supplying additional expert-knowledge. The reachability-based approach has been shown to be effective on larger system dimensions. Nevertheless, as discussed before, the computation time is dependent on a wide range of factors and further research is required to establish more solid conclusions.

Input saturation Input saturation can be applied naturally within the certificate-based approach, without significantly impacting the synthesis time. This is done by either directly using discontinuous functions to bound the input, e.g. by using a saturation function, or by considering a finite discretized set of controller inputs as done in Section 4.7. On the other hand, for the reachability-based approach, saturation significantly complicates the synthesis, as discussed in Section 5.7.2. This is because discontinuous functions significantly affect the complexity of the reachability analysis. However, as we have seen, it is possible to include the input saturation as an additional specification, such that for all trajectories the saturation bounds are never met. However, this added constraint can result in longer synthesis times, as the complexity of the approach scales with the number of constraints and because the controller has to evolve additional nonlinearities to counter input constraint violations.

General hybrid effects As we have seen in Chapter 4, the certificate-based approach is capable of handling general hybrid effects, such as sampled-data systems and jumps in the state space. For the reachability-based approach, we have only considered sampled-data systems; the extension to hybrid systems is discussed in more detail in Section 6.3.

Complex specifications For the certificate-based approach, we only considered two specific types of specifications, both reach-avoid problems. However, these problems can be used as building blocks for more intricate specifications, as will be discussed in Section 6.2. The reachability-based approach is capable of handling STL specifications satisfying Assumption 2.3.1. In practise, this means that this method only deals with finite trajectories. However, a similar extension as will be described in 6.2 can be applied here.

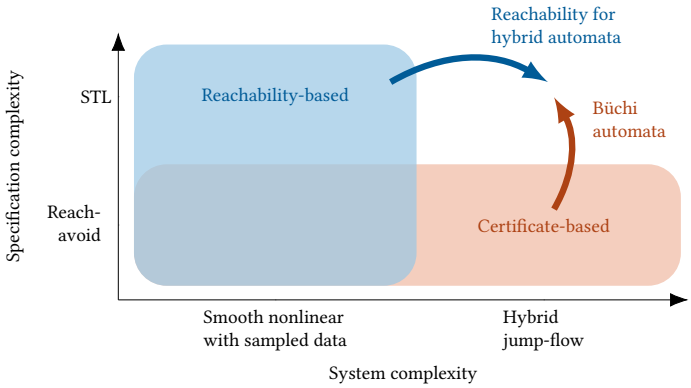


Figure 6.1: Overview of the addressed class of systems and specifications and future extensions.

6.2. Extension of certificate-based approaches

The certificate-based approach presented in Chapter 4 has been applied to reach-avoid problems, however, it can be extended to full temporal logic properties. Similar to e.g. [20, 36, 80, 121, 165], this could be done by combining Büchi automata and our current approach to simple safe reachability. Intuitively, the TL specification would be split up into reach-avoid sub-problems, and for each sub-problem a controller is synthesized. The overall controller becomes a hybrid controller with multiple modes, e.g. in the form an automaton, rather than a single closed-form expression, as compared to the reachability-based approach.

6.3. Extension of reachability-based approaches

The reachability-based approach in Chapter 5 has been presented for continuous-time systems with sampled-data controllers, but can be extended to general hybrid systems, similar to the extension of [138] to piecewise affine systems in [141]. In Chapter 5, we used the reachability tool CORA, which also supports reachability analysis for hybrid automata, enabling the extension to hybrid systems. However, in the proposed reachability-based approach, we relied on simulations to guide the proposal of candidate controllers; the simulation of hybrid systems is more intricate, as one needs to make sure the simulated trajectories do not miss the jump set[3] due to numerical errors [9]. Similarly, the reachability analysis becomes more intricate, and an increase in computational cost and conservatism is expected. However, similar to [141], this can be (partly) countered by designing the controller such that the number of jumps of the hybrid system are minimized.

6.4. Conclusion

In this chapter we briefly compared the two approaches proposed within this dissertation on a selection of criteria. Per category, an indicative assessment of the performance is summarized in Table 6.1. In conclusion, the respective methods are best used for:

[3]Or in case of a hybrid automata, the guard set.

Table 6.1: Comparison of the synthesis approaches based on certificate functions (CF) and reachability analysis (RA).

	CF-based	RA-based
Computation cost	+	−
Scalability w.r.t. initial set	++	−
Scalability w.r.t. system dimension	−	+
Input saturation	+ + +	−
General hybrid effects	+	−−
Complex specifications	−	++

- Certificate-based: low-dimensional systems with potential hybrid effects, such as input saturation and jumps, and a large initial set.
- Reachability-based: 'higher-dimensional' continuous-time systems with a small initial set, subjected to (finite horizon) STL specifications.

Finally, we discussed the extension of both methods to a wider class of systems or specifications, illustrated in Figure 6.1.

7

Conclusions and recommendations

In this concluding chapter we summarize the main contributions of the dissertation and provide an outlook for potential future work.

7.1. Conclusions

The goal of this thesis is to develop a framework for automatic synthesis of closed-form controllers for hybrid systems, such that temporal logic specifications are formally guaranteed. The closed-loop nature of the controllers enables the implementation in embedded hardware with limited memory and/or limited computation power, as opposed to e.g. controllers in the form of enormous look-up tables or controllers relying on online optimization methods. To achieve this goal, we have proposed two CEGIS frameworks, in which we combined the proposal of candidate controllers through GGGP with formal verification and counterexample generation. The two proposed frameworks differ in the verification method, i.e., indirectly by means of co-synthesized certificate functions, or directly through reachability analysis.

In Chapter 3 we introduced our grammar-guided genetic programming algorithm. The use of grammar-guided genetic programming allows for the synthesis of closed-form expressions, without the need of pre-defining a template solution, i.e., it is capable of discovering the correct controller structure automatically. Furthermore, the use of a grammar enables the restriction of the search space and enables the incorporation of expert-knowledge. While expert-knowledge can greatly speed up the synthesis, it is completely optional.

In Chapter 4 we introduced an indirect method, in which the verification relies on the co-synthesis of a controller and a certificate function. This method is suitable for hybrid systems modelled as jump-flow systems with reach-avoid specifications. In this chapter, we introduced a Lyapunov barrier function which by its existence implies the desired specification, and we introduced several relaxations. Additionally, we considered

a specialized synthesis approach for continuous open-loop systems with sampled-data controllers, based on a control Lyapunov barrier function. Finally, we demonstrated how this framework can also be used for the formal verification of (near) optimal controllers, obtained through reinforcement learning. The effectiveness of the methods in this chapter was illustrated on several benchmarks from the literature.

In Chapter 5, we introduced a direct method, in which the verification relies on reachability analysis. This method is suitable for continuous open-loop systems with STL specification, where the controller is implemented in a sampled-data fashion. This method relies on model checking for STL, based on reachability analysis, in which an STL formula is transformed to an RTL formula. To be able to quantify the performance with respect to an RTL formula, we introduced quantitative semantics for RTL. The effectiveness of this approach was shown on several benchmarks found in literature.

In Chapter 6, we compared the certificate-based and reachability-based methods. The former is better suited for low-dimensional hybrid systems with large initial sets, whereas the latter is better suited for comparatively higher-dimensional continuous open-loop systems with more intricate specifications. However, both methods can synthesize closed-form sampled-data controllers, which simplify the implementation in digital platforms with limited memory and computation power. Finally, both methods have a potential to be extended to handle general hybrid systems with general STL specifications.

In conclusion, the two frameworks presented in this dissertation have the potential to overcome all shortcomings listed in Section 1.3. Both frameworks return correct-by-construction closed-form controllers, which solve problems 1 and 2, i.e. their compact representation and relatively low online computation cost enable the implementation in embedded hardware. While the presented methods deal with either general hybrid systems or general STL specifications, we proposed realistic extensions to general hybrid systems with general STL specifications, addressing problems 3 and 4. Finally, using grammar-guided genetic programming, the use of expert-knowledge is optional, which addresses problem 5.

7.2. Recommendations for future work

Building upon the foundations laid out in this thesis, in this section we discuss open challenges and explore possible directions for future research.

- In the case studies for both methods in Chapters 4 and 5, it was shown that the computation time depends on a range of factors and no conclusive statements on the scalability could be made. Influencing factors depend on an interplay of the dynamics, specification, and the supplied expert-knowledge. Further research is required to solidify claims regarding these relations.
- In this work we have limited our synthesis to static feedback controllers. In future work, one could extend our framework to the design of dynamic controllers, e.g. controllers including an observer.
- The computation time of the presented methods can be improved. First of all, the benefit of GP is that it is highly parallelizable. In the presented work, parallelization has only been utilized on a small scale. In future work, it is highly encouraged

to further exploit parallelization to improve on the synthesis time. Secondly, in our work, we used a relatively simple implementation of GGGP and efficiency can potentially be improved in a number of ways, see e.g. the survey in [31]. Finally, the frameworks presented in this work are implemented in high-level languages, such as Mathematica and Matlab. To help improve the computation time, it is advised to implement these approaches in lower-level languages such as C/C++.

- As we have seen in Section 6, both certificate-based and reachability-based approaches, can be extended in future work to a wider class of specifications and systems, respectively. However, despite these extensions, each method has their different use cases based on their respective strengths and weaknesses, as was discussed in Section 6.3.

- While genetic programming has the advantage of synthesizing compact expressions, the expressive power of neural networks is widely acknowledged in a broad range of studies. In the recent works [1, 27], neural networks are combined with formal verification, illustrating the potential of neural networks as Lyapunov functions and/or controllers. Along the same lines, in future work, one can replace genetic programming with neural networks and adapt the verification accordingly.

- In Chapter 4.8, we employed reinforcement learning and the automatic synthesis of a certificate function for the synthesis of a near-optimal controller with safety and reachability guarantees. There are great similarities between a Lyapunov function and the value function in Reinforcement learning [114]. An interesting direction for future work is to exploit these similarities for simultaneous synthesis and verification.

- In this work, we employed automatic formal control synthesis methods with respect to a given model. In the absence of such a model, similar evolutionary methods as used in this dissertation can be employed to derive a model, see e.g., [60, 78, 137]. In future work, the merging of these methods into an integrated framework for both system identification and formal controller synthesis, similar to, e.g., [134], would further increase the automation of end-to-end controller design.

A

Mathematical proofs

A.1. Proof Theorem 4.3.1

Given that $\phi(0,0) \in I$, if $\phi(0,0) \in O$, (4.2) holds trivially. For $\phi(0,0) \in I \backslash O$, from (4.5a) we have that $V(\phi(0,0)) \leq 0$ and thus $\phi(0,0) \in A^* := A \backslash O$. From condition (4.5b) and the definition of A in (4.6) we have $A \cap \partial S = \emptyset$. Consider a hybrid time interval $[t_j, t_{j+1}] \times \{j\} \subseteq \mathrm{dom}\phi$. For almost all $t \in [t_j, t_{j+1}]$ such that $\phi(t,j) \in A^*_C$, we have from (4.5d) that

$$\frac{d}{dt} V(\phi(t,j)) \leq \max_{f \in F(\phi(t,j))} \langle \nabla V, f \rangle \leq -\gamma_{\mathrm{c}},$$

i.e. V decreases along the flow, hence solutions remain in the sublevel set $A \subset S$ and thus cannot leave the safe set within an arbitrarily small time step. From (4.5c) it follows that all jumps starting from A^*_D jump to the safe set S. Moreover, from condition (4.5e) it follows that for $\phi(t,j) \in A^*_D$ with $(t, j+1) \in \mathrm{dom}\phi$:

$$V(\phi(t, j+1)) \leq V(\phi(t,j)) - \gamma_{\mathrm{d}},$$

i.e. V decreases along a jump, hence solutions remain in the sublevel set A. Summarizing, all $\phi(t,j) \in A^*$ remain in $A \subset S \subset C \cup D$ under an arbitrarily small interval of time and/or jump.

Now, by contradiction, we prove that eventually all trajectories starting in A^* enter O. Consider a complete solution which always remains within A^*. Since $V(s)$ is continuous and S is compact, $V[S] \subset \mathbb{R}$ is compact and hence $V[S \backslash O] \subseteq V[S]$ is bounded, i.e. $\exists e \in \mathbb{R}$ such that $\forall s \in A^*$, $V(s) \geq e$. Using $\forall (t,j) \in \mathrm{dom}\phi : V(\phi(t,j)) \in A^*$, equation (4.5e), integrating both sides of (4.5d), and $V(\phi(0,0)) \leq 0$, we have

$$V(\phi(t,j)) \leq -t\gamma_{\mathrm{c}} - j\gamma_{\mathrm{d}}. \tag{A.1}$$

Since the maximal solution ϕ is complete, j is unbounded and/or t is unbounded, which implies in both cases that there exists a finite T and J such that $V(\phi(T,J)) < e$ and thus $\phi(T,J) \notin A^*$, contradicting the premise. Since all $\phi(t,j) \in A^*$ cannot leave $A \subset S \subset$

A

$C \cup D$ within an arbitrarily small interval of time and/or a number of jumps, the only possibility is that there exists a $(T, J) \in \text{dom}\phi$ such that $\phi(T, J) \in O$, and thus (4.2) holds. □

A.2. Proof Corollary 4.3.1

From Theorem 4.3.1 we have that for all maximal solutions $\phi \in \mathcal{S}_{\mathcal{H}_{\text{cl}}}(I)$, there exists a pair $(T, J) \in \text{dom}\phi$ such that $\phi(T, J) \in O$. Analogous to the proof of Theorem 4.3.1, conditions (4.7a), (4.7b), and (4.7c) imply that $\forall \phi(t, j) \in O$, $\exists (T_1, J_1) \in \text{dom}\phi$ such that $\phi(T_1, J_1) \in B$.

Since $B := \{s \in O \mid V(s) \leq \beta\}$ and O is compact, it follows that B is compact. Condition (4.7b) implies that

$$\forall s \in \partial B \cap C, \forall f \in F(s) : \langle \nabla V(s), f \rangle \leq -\gamma_{\text{c}}. \tag{A.2}$$

Combining this with (4.7d), we have that all states $\phi(t, j) \in \partial B \cap C$ cannot reach ∂O. Therefore it follows that during flow, trajectories starting in B remain within $B \subset O$. From (4.7e) we have if $\{(t, j), (t, j + 1)\} \subset \text{dom}\phi$ and $\phi(t, j) \in B \cap D$, it follows that $\phi(t, j + 1) \in B$, hence for all jumps starting in B the solutions ϕ remain within $B \subset O$. Summarizing, solutions within B stay within B and thus B is forward invariant. Since $O \subset S$, we have that (4.3) holds. □

A.3. Proof Proposition 4.4.1

From condition (4.8a) and the definition of A in (4.6) we have $A \cap (\partial S_{\text{x}} \times S_{\text{q}} \times \mathcal{T}) = \emptyset$. During flow, the discrete states ϕ_{q} remain constant and the timer states ϕ_{t} remain within $\mathcal{T}$. Therefore, the solution can only escape the safe set $S := S_{\text{x}} \times S_{\text{q}} \times \mathcal{T}$ through the boundary of the safe set of continuous states $\partial S_{\text{x}} \times S_{\text{q}} \times \mathcal{T}$. Analogous to the proof of Theorem 4.3.1, trajectories in A^*_C result in a decrease along V, and therefore trajectories cannot leave the sublevel set A and thus neither the safe set S within an arbitrarily small time step. Analogous to the proof of Theorem 4.3.1, jumps from A^*_D remain in A.

Now, one can show by contradiction that complete solutions cannot remain forever in A^*. Again, $\exists e \in \mathbb{R}$ such that $\forall s \in A^*$, $V(s) \geq e$. Let us denote the number of jumps resulting from $(D_{\text{s}}, G_{\text{s}})$ and $(D_{\text{t}}, G_{\text{t}})$ by j_{s} and j_{t}, respectively. Using $\forall (t, j) \in \text{dom}\phi : V(\phi(t, j)) \in A^*$, equations (4.8b), (4.8c), integrating both sides of (4.5d), and $V(\phi(0, 0)) \leq 0$, yields

$$V(\phi(t, j)) \leq -t\gamma_{\text{c}} - j_{\text{s}}\gamma_{\text{d}}. \tag{A.3}$$

By the definition of the dynamics of the timer states we have that j_{t} depends on time, i.e.: $j_{\text{t}}(t) = \sum_{i=1}^{n_{\text{t}}} \left\lfloor \frac{t + \phi_{\text{t},i}(0,0)}{\eta_i} \right\rfloor$. Since the maximal solution ϕ is complete, t is unbounded and/or $j = j_{\text{s}} + j_{\text{t}}$ is unbounded because t is unbounded or j_{s} is unbounded. In all cases there exists a finite T and J such that $V(\phi(T, J)) < e$ and thus $\phi(T, J) \notin A^*$. The remainder of the proof is analogous to the proof of Theorem 4.3.1. □

A.4. Proof Corollary 4.4.1

This proof is analogous to the proof of Corollary 4.3.1, where Proposition 4.4.1 is used instead of Theorem 4.3.1. Analogous to condition (4.7d) in Corollary 4.3.1, condition (4.9c)

yields that all states $\varphi(t,j) \in \partial B \cup C$ cannot reach $\partial O_{\mathrm{x}} \times O_{\mathrm{q}} \times \mathcal{T}$. Since the discrete states $\phi_{\mathrm{q}}(t,j)$ remain constant during flows and the timer states $\phi_{\mathrm{t}}(t,j)$ always stay within $\mathcal{T}$, it follows that during flow, trajectories starting in B remain within $B \subset O := O_{\mathrm{x}} \times O_{\mathrm{q}} \times \mathcal{T}$. The remainder of the proof is analogous to Corollary 4.3.1. □

A.5. Proof Corollary 4.4.2

From the proof of Corollary 4.4.1 it follows that $\forall \phi \in \mathcal{S}_{\mathcal{H}_{\mathrm{cl}}}(I)$, $\exists (T,J) \in \mathrm{dom}\phi$ such that $\forall (t,j) \in E_{\geq (T,J)}$, $\phi(t,j) \in B$. Since $B \cap D_{\mathrm{s}} = \emptyset$, the only jumps taking place for $(t,j) \in E_{\geq (T,J)}$ are because $\phi(t,j) \in D_{\mathrm{t}}$, i.e. due to timer updates. Since every jump induced by a timer state has a fixed minimal dwell-time of η_i and there are only a finite number of timer states, it follows that all solutions $\phi \in \mathcal{S}_{\mathcal{H}_{\mathrm{cl}}}(I)$ are non-Zeno. □

A.6. Proof Corollary 4.4.3

As a consequence of the conditions in Theorem 4.3.1, Proposition 4.4.1 or Corollaries 4.3.1 and 4.4.1, after a jump $\phi(t,j) \in S \subset C \cup D$ and under Assumption 4.4.2, we have that $\phi(t,j) \notin D$. Therefore solutions can only be extended through flow, along which the LBF decreases. □

A.7. Proof Theorem 4.7.1

For $\xi(t_0) \in I$ it follows from (4.22a) and the definition of A that $\xi(t_0) \in A$. From (4.22c) it follows that for all $\xi(t_k) \in A \backslash O$ there exists a $q \in Q$ such that $\forall t \in [t_k, t_k + h]$: $\dot{V}_q(\xi(t_k), \xi(t)) \leq -\gamma$. Selecting such a mode using controller (4.23), applying the comparison theorem (see e.g. [77]), and using $\forall x \in A$, $V(x) \leq 0$, it follows that $\forall k \in \mathbb{Z}_{\geq 0}$, $\forall t \in [t_k, t_k + h]$, $\forall \xi(t_k) \in A \backslash O$: $V(\xi(t)) \leq V(\xi(t_k)) - \gamma h \leq -\gamma h$. Therefore, $\xi(t_k) \in A \backslash O$ implies $\forall t \in [t_k, t_k + h]$, $V(\xi(t))$ will decrease and thus cannot reach ∂S, as from (4.22b) we have $\forall x \in \partial S$: $V(x) > 0$. Since $V(x)$ is continuous and S is compact, $V[S] \subset \mathbb{R}$ is compact and hence $V[S \backslash O] \subseteq V[S]$ is bounded, $V(\xi(t))$ will decrease until in finite time $\xi(t)$ leaves $A \backslash O$ and can only enter O, therefore (4.2) holds. □

A.8. Proof Corollary 4.7.1

From Theorem 4.7.1 we have that there exists a time $t_K \geq t_0$ such that $\xi(t_K) \in O$. Analogous to the proof of Theorem 4.7.1, from (4.24b) it follows that $\forall \xi(t_K) \in O$, $\xi(t)$ with $t \geq t_K$ enters in finite time $O \cap B$. From the definition of B and since every sublevel set $L_c := \{x \in S \mid V(x) \leq c\}$ is compact, it follows that B is compact and thus $O \cap B$ is compact. From (4.24b) and controller (4.23) we have that $\forall x \in \partial(O \cap B), z \in R_q(x)$: $\dot{V}_q(x,z) \leq -\gamma$. Combining this with (4.24a), we have that all states $\xi(t) \in \partial(O \cap B)$ cannot reach ∂O and $V(\xi(t))$ decreases, thus these trajectories will remain within $O \cap B$. Therefore it follows that $O \cap B \subseteq O$ is forward invariant. As $O \subseteq int(S)$, we have that (4.3) holds. □

A

A.9. Proof Theorem 4.7.2

This proof is by contradiction. By definition of the CLBF, for all $x \in A \backslash O$, there always exists a q such that $\max_{z \in R_q(x)} \dot{V}_q(x,z) \leq -\gamma$. Assume that when using switching law (4.30), we have $\max_{z \in R_{q_k}(\xi(t))} \dot{V}_{q_k}(\xi(t),z) > -\gamma$. It then follows from (4.29) that $\min_{q \in Q}(\dot{V}_q(x,x) + \alpha_q(x)) < \dot{V}_{q_k}(x,x) + \alpha_{q_k}(x)$. This directly contradicts the switching law

$$q_k = \min_{q \in Q}(\dot{V}_q(x,x) + \alpha_q(x)).$$

Hence switching law (4.30) can only select a q_k such that $\max_{z \in R_{q_k}(\xi(t_k))} \dot{V}_{q_k}(\xi(t_k),z) \leq -\gamma$, which is guaranteed to exist by the design of the CLBF. The remainder of the proof is analogous to the proof of Theorem 4.7.1. □

A.10. Proof of Theorem 5.3.1

Theorem 5.3.1 is proven by induction over the structure of the RTL formula Ψ and subformula ψ. This is only done for the first statement in Theorem 5.3.1, as the second statement is logically equivalent to the first, i.e.:

$$P(R,\Psi,t) > 0 \Rightarrow (R,t) \models \Psi \equiv (R,t) \not\models \Psi \Rightarrow P(R,\Psi,t) \leq 0,$$
$$(R,t) \models \Psi \Rightarrow P(R,\Psi,t) \geq 0 \equiv P(R,\Psi,t) < 0 \Rightarrow (R,t) \not\models \Psi.$$

- **Case $\psi = \textbf{true}$:** By definition $x \models \psi$ and $\varrho(x,\psi) > 0$.
- **Case $\psi = h(x) \geq 0$:** For this formula ψ, the quantitative semantics is given by $\varrho(x,\psi) = h(x)$. **(i)** If $\varrho(x,\psi) > 0$, then $h(x) > 0$, thus from the semantics it follows that $x \models \psi$. **(ii)** If $x \models \psi$, then from the semantics we have $h(x) \geq 0$, thus from the quantitative semantics it follows that $\varrho(x,\psi) \geq 0$.
- **Case $\psi = \neg\psi_1$:** For this formula ψ, the quantitative semantics is given by $\varrho(x,\neg\psi_1) = -\varrho(x,\psi_1)$. **(i)** If $\varrho(x,\neg\psi_1) > 0$, then $\varrho(x,\psi_1) < 0$. By the induction hypothesis, we get $x \not\models \psi_1$ and thus from the semantics it follows that $x \models \neg\psi_1$. **(ii)** If $x \models \neg\psi_1$, then from the semantics we have $x \not\models \psi_1$. By the induction hypothesis and the equivalence $\varrho(x,\psi) > 0 \Rightarrow x \models \psi \equiv x \not\models \psi \Rightarrow \varrho(x,\psi) \leq 0$, we get $\varrho(x,\psi_1) \leq 0$, thus $\varrho(x,\neg\psi_1) \geq 0$.
- **Case $\psi = \psi_1 \wedge \psi_2$:** For this formula ψ, the quantitative semantics is given by $\varrho(x,\psi_1 \wedge \psi_2) = \min(\varrho(x,\psi_1), \varrho(x,\psi_2))$. **(i)** If $\varrho(x,\psi_1 \wedge \psi_2) > 0$, then $\varrho(x,\psi_1) > 0$ and $\varrho(x,\psi_2) > 0$. By the induction hypothesis, we get $x \models \psi_1$ and $x \models \psi_2$, thus from the semantics it follows that $x \models \psi_1 \wedge \psi_2$. **(ii)** If $x \models \psi_1 \wedge \psi_2$, then from the semantics we have $x \models \psi_1$ and $x \models \psi_2$. By the induction hypothesis, we get $\varrho(x,\psi_1) \geq 0$ and $\varrho(x,\psi_2) \geq 0$, thus $\varrho(x,\psi_1 \wedge \psi_2) \geq 0$.
- **Case $\Psi = \mathcal{A}\psi$:** For this formula Ψ, the quantitative semantics is given by $P(R,\mathcal{A}\psi,t) = \min_{x \in R(t)} \varrho(x,\psi)$. **(i)** If $P(R,\mathcal{A}\psi,t) > 0$, then $\forall x \in R(t) : \varrho(x,\psi) > 0$. By the induction hypothesis, $\forall x \in R(t) : x \models \psi$, thus from the semantics we have $(R,t) \models \mathcal{A}\psi$. **(ii)** If $(R,t) \models \mathcal{A}\psi$, then from the semantics we have $\forall x \in R(t) : x \models \psi$. By the induction hypothesis, we get $\forall x \in R(t) : \varrho(x,\psi) \geq 0$, thus $P(R,\mathcal{A}\psi,t) \geq 0$.

- **Case** $\Psi = \Psi_1 \vee \Psi_2$**:** For this formula Ψ, the quantitative semantics is given by $P(R, \Psi_1 \vee \Psi_2, t) = \max(P(R, \Psi_1, t), P(R, \Psi_2, t))$. **(i)** If $P(R, \Psi_1 \vee \Psi_2, t) > 0$, then $P(R, \Psi_1, t) > 0$ or $P(R, \Psi_2, t) > 0$. By the induction hypothesis, we get $(R, t) \models \Psi_1$ or $(R, t) \models \Psi_2$, thus from the semantics it follows that $(R, t) \models \Psi_1 \vee \Psi_2$. **(ii)** If $(R, t) \models \Psi_1 \vee \Psi_2$, then from the semantics we have $(R, t) \models \Psi_1$ or $(R, t) \models \Psi_2$. By the induction hypothesis, we get $P(R, \Psi_1, t) \geq 0$ or $P(R, \Psi_2, t) \geq 0$, thus $P(R, \Psi_1 \vee \Psi_2, t) \geq 0$.

- **Case** $\Psi = \Psi_1 \wedge \Psi_2$**:** For this formula Ψ, the quantitative semantics is given by $P(R, \Psi_1 \wedge \Psi_2, t) = \min(P(R, \Psi_1, t), P(R, \Psi_2, t)$. **(i)** If $P(R, \Psi_1 \wedge \Psi_2, t) > 0$, then $P(R, \Psi_1, t) > 0$ and $P(R, \Psi_2, t) > 0$. By the induction hypothesis, we get $(R, t) \models \Psi_1$ and $(R, t) \models \Psi_2$, thus from the semantics it follows that $(R, t) \models \Psi_1 \wedge \Psi_2$. **(ii)** If $(R, t) \models \Psi_1 \wedge \Psi_2$, then from the semantics we have $(R, t) \models \Psi_1$ and $(R, t) \models \Psi_2$. By the induction hypothesis, we get $P(R, \Psi_1, t) \geq 0$ and $P(R, \Psi_2, t) \geq 0$, thus $P(R, \Psi_1 \wedge \Psi_2, t) \geq 0$.

- **Case** $\Psi = \bigcirc_a \Psi_1$**:** For this formula Ψ, the quantitative semantics is given by $P(R, \bigcirc_a \Psi_1, t) = P(R, \Psi, t + a)$. **(i)** If $P(R, \bigcirc_a \Psi_1, t) > 0$, then $P(R, \Psi_1, t + a) > 0$. By the induction hypothesis, we get $(R, t + a) \models \Psi_1$, thus from the semantics we have $(R, t) \models \bigcirc_a \Psi_1$. **(ii)** If $(R, t) \models \bigcirc_a \Psi_1$, then from the semantics we have $(R, t + a) \models \Psi_1$. By the induction hypothesis, we get $P(R, \bigcirc_a \Psi_1, t) \geq 0$.

□

B

Standard forms of the LBF and CLBF inequalities

The conditions in Proposition 4.4.1, Corollary 4.4.1, Defintion 4.7.1 and Corollary 4.7.1 can be written in the standard form (4.10):

$$\varphi := \forall x \in X : \left(\bigwedge\nolimits_{i=1}^{k} \left(\bigvee\nolimits_{j=1}^{l_i} f_{ij}(x) \leq 0 \right) \right),$$

as shown in Table B.1 and B.2. In these tables $c > 0$ is an arbitrary positive constant, used to cast strict inequalities to non-strict inequalities. In Table B.2, for conditions φ_3 and φ_5, the state x is partitioned as $x = (s_x, s_{\tau,1}, s_{e,1}, \ldots, s_{\tau,|Q|}, s_{e,|Q|})$, and $\dot{V}_q(x)$ is used to denote

$$\dot{V}_q(x) = \langle \nabla V(r_q(s_x, s_{\tau,q}, s_{e,q})), f(r_q(s_x, s_{\tau,q}, s_{e,q}), g_q(s_x)) \rangle \,, \tag{B.1}$$

where r_q is given in (4.7.3).

We briefly demonstrate how the inequalities formulated over a sublevel set can be formulated in the standard form (4.10). Given a general sublevel set $L_X^a(V) := \{x \in X \mid V(x) \leq a\}$, a logic formula of the form $\forall x \in L_X^c : f(x) \leq 0$, can be expressed using a logical implication as $\forall x \in X : V(x) \leq a \implies f(x) \leq 0$, which is equivalent to $\forall x \in X : V(x) > a \vee f(x) \leq 0$. Putting this expression in standard form yields:

$$\forall x \in X : -V(x) + a + c \leq 0 \vee f(x) \leq 0, \tag{B.2}$$

with some $c > 0$, which is in the standard form (4.10).

B

Table B.1: Standard form (4.10) of the LBF conditions in Proposition 4.4.1 and Corollary 4.4.1.

φ_i	eq.	X	$f_{ij}(x)$
φ_1	(4.5a)	I	$f_{11}(x) = V(x)$.
φ_2	(4.8a)	$\partial S_{\mathrm{x}} \times S_{\mathrm{q}} \times \mathcal{T}$	$f_{11}(x) = -V(x) + c$.
φ_3	(4.5c)	$\{(x_1, x_2) \in (S \backslash O \cap D) \times \mathbb{R}^n \mid x_2 \in G(x_1)\}$	$f_{i1}(x) = -V(x_1) + c$, $f_{i2}(x) = b_{S,i}(x_2),\ i \in \{1, \dots, i_S\}$.
φ_4	(4.5d)	$\{(x_1, x_2) \in (S \backslash O \cap C) \times \mathbb{R}^n \mid x_2 \in F(x_1)\}$	$f_{11}(x) = -V(x_1) + c$, $f_{12}(x) = \langle \nabla V(x_1), x_2 \rangle + \gamma_{\mathrm{c}}$.
φ_5	(4.8b)	$\{(x_1, x_2) \in (S \backslash O \cap D_{\mathrm{s}}) \times \mathbb{R}^n \mid x_2 \in G_{\mathrm{s}}(x_1)\}$	$f_{11}(x_1) = -V(x_1) + c$ $f_{12}(x) = V(x_2) - V(x_1) + \gamma_{\mathrm{d}}$.
φ_6	(4.8c)	$\{(x_1, x_2) \in (S \backslash O \cap D_{\mathrm{t}}) \times \mathbb{R}^n \mid x_2 \in G_{\mathrm{t}}(x_1)\}$	$f_{11}(x) = -V(x_1) + c$, $f_{12}(x) = V(x_2) - V(x_1)$.
φ_7	(4.7a)	$\{(x_1, x_2) \in (O \cap D) \times \mathbb{R}^n \mid x_2 \in G(x_1)\}$	$f_{i1}(x) = V(x_1) - \beta + c$, $f_{i2}(x) = b_{S,i}(x_2),\ i = \{1, \dots, i_S\}$.
φ_8	(4.7b)	$\{(x_1, x_2) \in (O \cap C) \times \mathbb{R}^n \mid x_2 \in F(x_1)\}$	$f_{11}(x) = V(x_1) - \beta + c$, $f_{12}(x) = \langle \nabla V(x_1), x_2 \rangle + \gamma_{\mathrm{c}}$.
φ_9	(4.9a)	$\{(x_1, x_2) \in (O \cap D_{\mathrm{s}}) \times \mathbb{R}^n \mid x_2 \in G_{\mathrm{s}}(x_1)\}$	$f_{11}(x) = V(x_1) - \beta + c$, $f_{12}(x) = V(x_2) - V(x_1) + \gamma_{\mathrm{d}}$.
φ_{10}	(4.9b)	$\{(x_1, x_2) \in (O \cap D_{\mathrm{t}}) \times \mathbb{R}^n \mid x_2 \in G_{\mathrm{t}}(x_1)\}$	$f_{11}(x) = V(x_1) - \beta + c$, $f_{12}(x) = V(x_2) - V(x_1)$.
φ_{11}	(4.9c)	$\partial O_{\mathrm{x}} \times O_{\mathrm{q}} \times \mathcal{T}$	$f_{11}(x) = -V(x) + \beta + c$.
φ_{12}	(4.7e)	$\{(x_1, x_2) \in (O \cap D) \times \mathbb{R}^n \mid x_2 \in G(x_1)\}$	$f_{i1}(x) = -V(x_1) + \beta + c$, $f_{12}(x) = V(x_2) - \beta$, $f_{(k+1)2}(x) = b_{O,k}(x_2)$, $k \in \{1, \dots, i_O\},\ i \in \{1, \dots i_O + 1\}$.

Table B.2: Standard form (4.10) of the CLBF conditions in Defintion 4.7.1 and Corollary 4.7.1.

φ_i	eq.	X	$f_{ij}(x)$
φ_1	(4.22a)	I	$f_{11}(x) = V(x)$.
φ_2	(4.22b)	∂S	$f_{11}(x) = -V(x) + c$.
φ_3	(4.22c)	$S \backslash O \times \Pi_{q \in Q} E$	$f_{11}(x) = -V(s_x) + c$, $f_{1(q+1)}(x) = \dot{V}_q(x) + \gamma$, $q \in Q$.
φ_4	(4.24a)	∂O	$f_{11}(x) = -V(x) + \beta + c$.
φ_5	(4.24b)	$O \times \Pi_{q \in Q} E$	$f_{11}(x) = -V(s_x) + \beta$, $f_{1(q+1)}(x) = \dot{V}_q(x) + \gamma$,$q \in Q$.

Bibliography

[1] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo. Automated formal synthesis of Lyapunov neural networks. *arXiv preprint arXiv:2003.08910*, 2020.

[2] A. Abate, I. Bessa, L. Cordeiro, C. David, P. Kesseli, D. Kroening, and E. Polgreen. Automated formal synthesis of provably safe digital controllers for continuous plants. *Acta Informatica*, pages 1–22, 2020.

[3] A. A. Ahmadi, M. Krstic, and P. A. Parrilo. A globally asymptotically stable polynomial vector field with no polynomial Lyapunov function. In *CDC-ECE*, pages 7579–7580, 2011.

[4] D. Ahmed, A. Peruffo, and A. Abate. Automated and sound synthesis of Lyapunov functions with smt solvers. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 97–114. Springer International Publishing, 2020.

[5] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta. Q-learning for robust satisfaction of signal temporal logic specifications. In *IEEE Conf. on Decision and Control (CDC)*, pages 6565–6570, 2016.

[6] M. Althoff. *Reachability analysis and its application to the safety assessment of autonomous cars*. PhD thesis, Technische Universität München, 2010.

[7] M. Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pages 173–182. ACM, 2013.

[8] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[9] M. Althoff. Cora 2018 manual. *TU Munich*, 2018.

[10] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans. on Automatic Control*, 62(8):3861–3876, 2016.

[11] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer Berlin Heidelberg, 2011.

[12] Z. Artstein. Stabilization with relaxed controls. *Nonlinear Analysis: Theory, Methods & Applications*, 7(11):1163 – 1173, 1983.

[13] J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, and M. Woodger. Revised report on the algorithm language algol 60. *Commun. ACM*, 6(1):1–17, 1963.

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

[15] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability modulo theories. *Handbook of satisfiability*, 185:825–885, 2009.

[16] C. Belta and S. Sadraddini. Formal methods for control synthesis: An optimization perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):115–140, 2019.

[17] C. Belta, B. Yordanov, and E. A. Gol. *Formal methods for discrete-time dynamical systems*, volume 89. Springer, 2017.

[18] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems*, pages 908–918. Curran Associates, Inc., 2017.

[19] A. Bisoffi and D. V. Dimarogonas. A hybrid barrier certificate approach to satisfy linear temporal logic specifications. In *Annual American Control Conf.*, pages 634–639, 2018.

[20] A. Bisoffi and D. V. Dimarogonas. Satisfaction of linear temporal logic specifications through recurrence tools for hybrid systems. *IEEE Trans. on Automatic Control*, pages 1–1, 2020.

[21] I. Bladek, K. Krawiec, and J. Swan. Counterexample-driven genetic programming: Heuristic program synthesis from formal specifications. *Evolutionary Computation*, 26(3):441–469, 2018.

[22] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. Siam, 1994.

[23] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: model and optimal control theory. *IEEE Trans. on Automatic Control*, 43(1):31–45, 1998.

[24] R. W. Brockett. *Hybrid Models for Motion Control Systems*, pages 29–53. Birkhäuser Boston, 1993.

[25] L. Buşoniu, D. Ernst, R. Babuška, and B. De Schutter. Approximate dynamic programming with a fuzzy parameterization. *Automatica*, 46(5):804–814, 2010.

[26] J. Cámara, A. Girard, and G. Gössler. Synthesis of switching controllers using approximately bisimilar multiscale abstractions. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, page 191–200. ACM, 2011.

[27] Y.-C. Chang, N. Roohi, and S. Gao. Neural Lyapunov control. In *Advances in Neural Information Processing Systems*, pages 3245–3254. Curran Associates, Inc., 2019.

[28] P. Chen and Y.-Z. Lu. Automatic design of robust optimal controller for interval plants using genetic programming and kharitonov theorem. *Int. Journal of Computational Intelligence Systems*, 4(5):826–836, 2011.

[29] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Computer Aided Verification*, pages 258–263. Springer Berlin Heidelberg, 2013.

[30] K. Cpalka, K. Łapa, and A. Przybył. A new approach to design of control systems using genetic programming. *Information technology and control*, 44(4):433–442, 2015.

[31] V. K. Dabhi and S. Chaudhary. Empirical modeling using genetic programming: a survey of issues and approaches. *Natural Computing*, 14(2):303–330, 2015.

[32] L. de Moura and N. Bjørner. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer Berlin Heidelberg, 2008.

[33] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation*, 6(2):182–197, April 2002.

[34] J. V. Deshmukh, J. P. Kapinski, T. Yamaguchi, and D. Prokhorov. Learning deep neural network controllers for dynamical systems with safety guarantees: Invited paper. In *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pages 1–7, 2019.

[35] S. Di Cairano, W. P. M. H. Heemels, M. Lazar, and A. Bemporad. Stabilizing dynamic controllers for hybrid systems: A hybrid control Lyapunov function approach. *IEEE Trans. on Automatic Control*, 59(10):2629–2643, 2014.

[36] R. Dimitrova and R. Majumdar. Deductive control synthesis for alternating-time logics. In *Int. Conf. on Embedded Software (EMSOFT)*, pages 1–10, 2014.

[37] J. Ding, E. Li, H. Huang, and C. J. Tomlin. Reachability-based synthesis of feedback policies for motion planning under bounded disturbances. In *IEEE Int. Conf. on Robotics and Automation*, pages 2160–2165, 2011.

[38] A. I. Diveev and E. Y. Shmalko. Automatic synthesis of control for multi-agent systems with dynamic constraints. *IFAC-PapersOnLine*, 48(11):384–389, 2015.

[39] A. Donzé, T. Ferrère, and O. Maler. Efficient robust monitoring for STL. In *Computer Aided Verification*, pages 264–279. Springer Berlin Heidelberg, 2013.

[40] A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems*, pages 92–106. Springer Berlin Heidelberg, 2010.

[41] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari. Learning and verification of feedback control systems using feedforward neural networks. *IFAC-PapersOnLine*, 51(16):151 – 156, 2018. IFAC Conf. on Analysis and Design of Hybrid Systems (ADHS).

[42] M. Ehrgott. *Multicriteria optimization*. Springer Science & Business Media, 2005.

[43] A. E. Eiben and J. E. Smith. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[44] M. T. Emmerich and A. H. Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing*, 17(3):585–609, 2018.

[45] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343 – 352, 2009.

[46] S. S. Farahani, V. Raman, and R. M. Murray. Robust model predictive control for signal temporal logic synthesis. *IFAC-PapersOnLine*, 48(27):323 – 328, 2015. Analysis and Design of Hybrid Systems ADHS.

[47] P. J. Fleming and R. C. Purshouse. Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice*, 10(11):1223 – 1241, 2002.

[48] G. F. Franklin, J. D. Powell, A. Emami-Naeini, and J. D. Powell. *Feedback control of dynamic systems*, volume 3. Addison-Wesley Reading, MA, 1994.

[49] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Computer Aided Verification*, pages 379–395. Springer Berlin Heidelberg, 2011.

[50] S. Gao, J. Avigad, and E. M. Clarke. δ-complete decision procedures for satisfiability over the reals. In *Int. Joint Conf. on Automated Reasoning*, pages 286–300. Springer, 2012.

[51] S. Gao, S. Kong, and E. M. Clarke. dReal: An SMT solver for nonlinear theories over the reals. In *Int. Conf. on Automated Deduction*, pages 208–214. Springer, 2013.

[52] K. Garg and D. Panagou. Control-Lyapunov and control-barrier functions based quadratic program for spatio-temporal specifications. In *IEEE Conf. on Decision and Control (CDC)*, pages 1422–1429, 2019.

[53] P. Giesl and S. Hafstein. Review on computational methods for Lyapunov functions. *Discrete & Continuous Dynamical Systems - B*, 20:2291, 2015.

[54] A. Girard, G. Gössler, and S. Mouelhi. Safety controller synthesis for incrementally stable switched systems using multiscale symbolic models. *IEEE Trans. on Automatic Control*, 61(6):1537–1549, 2016.

[55] A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Trans. on Automatic Control*, 55(1):116–126, 2010.

[56] R. Gnadler, H.-J. Unrau, H. Fischlein, and M. Frey. *Ermittlung von My-Schlupf-Kurven an Pkw-Reifen*, volume 119 of *FAT-Schriftenreihe*. FAT, Frankfurt/M., 1995.

[57] R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid Dynamical Systems: modeling, stability, and robustness*. Princeton University Press, 2012.

[58] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*, volume 1, pages 69–93. Elsevier, 1991.

[59] E. Goubault, J. Jourdan, S. Putot, and S. Sankaranarayanan. Finding non-polynomial positive invariants and Lyapunov functions for polynomial systems through Darboux polynomials. In *American Control Conf.*, pages 3571–3578, 2014.

[60] G. J. Gray, D. J. Murray-Smith, Y. Li, K. C. Sharman, and T. Weinbrenner. Nonlinear model structure identification using genetic programming. *Control Engineering Practice*, 6(11):1341–1352, November 1998.

[61] C. Grosan and A. Abraham. *Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews*, pages 1–17. Springer Berlin Heidelberg, 2007.

[62] B. Grosman and D. R. Lewin. Lyapunov-based stability analysis automated by genetic programming. *Automatica*, 45(1):252 – 256, 2009.

[63] L. C. G. J. M. Habets, P. J. Collins, and J. H. van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Trans. on Automatic Control*, 51(6):938–948, 2006.

[64] H. Han, M. Maghenem, and R. G. Sanfelice. Sufficient conditions for satisfaction of formulas with until operators in hybrid systems. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2020.

[65] H. Han and R. G. Sanfelice. Linear temporal logic for hybrid dynamical systems: Characterizations and sufficient conditions. *Nonlinear Analysis: Hybrid Systems*, 36:100865, 2020.

[66] N. Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[67] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[68] P. He, L. Kang, C. G. Johnson, and S. Ying. Hoare logic-based genetic programming. *Science China Information Sciences*, 54(3):623–637, 2011.

[69] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.

[70] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[71] K. Hsu, R. Majumdar, K. Mallik, and A.-K. Schmuck. Multi-layered abstraction-based controller synthesis for continuous-time systems. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, page 120–129. ACM, 2018.

[72] Y. Jin, M. Olhofer, and B. Sendhoff. Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how? In *Proc. of the Annual Conf. on Genetic and Evolutionary Computation*, page 1042–1049. Morgan Kaufmann Publishers Inc., 2001.

[73] C. G. Johnson. Genetic programming with fitness based on model checking. In *Genetic Programming*, pages 114–124. Springer Berlin Heidelberg, 2007.

[74] J. Kapinski, J. V. Deshmukh, S. Sankaranarayanan, and N. Arechiga. Simulation-guided Lyapunov analysis for hybrid dynamical systems. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pages 133–142. ACM, 2014.

[75] G. Katz and D. Peled. Genetic programming and model checking: Synthesizing new mutual exclusion algorithms. In *Automated Technology for Verification and Analysis*, pages 33–47. Springer Berlin Heidelberg, 2008.

[76] G. Katz and D. Peled. Synthesizing, correcting and improving code, using model checking-based genetic programming. *Int. Journal on Software Tools for Technology Transfer*, 19(4):449–464, 2017.

[77] H. K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.

[78] D. Khandelwal. *Automating data-driven modelling of dynamical systems: an evolutionary computation approach*. PhD thesis, Technische Universiteit Eindhoven, 2020.

[79] E. S. Kim, M. Arcak, and M. Zamani. Constructing control system abstractions from modular components. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, page 137–146. ACM, 2018.

[80] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Trans. on Automatic Control*, 53(1):287–297, 2008.

[81] N. Kochdumper and M. Althoff. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *arXiv preprint arXiv:1901.01780*, 2019.

[82] N. Kochdumper, B. Schürmann, and M. Althoff. Utilizing dependencies to obtain subsets of reachable sets. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2020.

[83] P. Kokotović and M. Arcak. Constructive nonlinear control: a historical perspective. *Automatica*, 37(5):637 – 662, 2001.

[84] S. Kong, S. Gao, W. Chen, and E. Clarke. dReach: δ-reachability analysis for hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 200–205. Springer Berlin Heidelberg, 2015.

[85] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[86] J. R. Koza. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines*, 11(3):251–284, 2010.

[87] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic programming IV: Routine human-competitive machine intelligence*, volume 5. Springer US, 2003.

[88] J. Kubalík, E. Alibekov, and R. Babuška. Optimal control via reinforcement learning with symbolic policy approximation. *IFAC-PapersOnLine*, 50(1):4162 – 4167, 2017. IFAC World Congress.

[89] H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems.* John Wiley & Sons, Inc., 1972.

[90] T. S. Letia and A. O. Kilyen. Evolutionary synthesis of hybrid controllers. In *IEEE Int. Conf. on Intelligent Computer Communication and Processing (ICCP),*, pages 133–140, 2015.

[91] D. R. Lewin. Evolutionary algorithms in control system engineering. In *World Congress of IFAC*, pages 3–8, 2005.

[92] X. Li, Y. Ma, and C. Belta. A policy search method for temporal logic specified reinforcement learning tasks. In *Annual American Control Conf.*, pages 240–245, 2018.

[93] X. Li, Z. Serlin, G. Yang, and C. Belta. A formal methods approach to interpretable reinforcement learning for robotic planning. *Science Robotics*, 4(37), 2019.

[94] Y. Li and J. Liu. ROCS: A robustly complete control synthesis tool for nonlinear dynamical systems. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pages 130–135. ACM, 2018.

[95] L. Lindemann and D. V. Dimarogonas. Robust motion planning employing signal temporal logic. In *American Control Conf. (ACC)*, pages 2950–2955, May 2017.

[96] L. Lindemann and D. V. Dimarogonas. Control barrier functions for signal temporal logic tasks. *IEEE Control Systems Letters*, 3(1):96–101, 2019.

[97] L. Lindemann, C. K. Verginis, and D. V. Dimarogonas. Prescribed performance control for signal temporal logic specifications. In *IEEE Conf. on Decision and Control (CDC)*, pages 2997–3002, 2017.

[98] L. Lindemann and D. V. Dimarogonas. Feedback control strategies for multi-agent systems under a fragment of signal temporal logic tasks. *Automatica*, 106:284 – 293, 2019.

[99] J. Liu, N. Ozay, U. Topcu, and R. M. Murray. Synthesis of reactive switching protocols from temporal logic specifications. *IEEE Trans. on Automatic Control*, 58(7):1771–1785, 2013.

[100] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[101] K. Mallik, A. Schmuck, S. Soudjani, and R. Majumdar. Compositional synthesis of finite-state abstractions. *IEEE Trans. on Automatic Control*, 64(6):2629–2636, 2019.

[102] M. Mazo Jr., A. Davitian, and P. Tabuada. PESSOA: A tool for embedded controller synthesis. In *Computer Aided Verification*, pages 566–569. Springer Berlin Heidelberg, 2010.

[103] J. S. McGough, A. W. Christianson, and R. C. Hoover. Symbolic computation of Lyapunov functions using evolutionary algorithms. In *Proc. of the IASTED Int. Conf.*, volume 15, page 17, 2010.

[104] R. I. McKay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. O'Neill. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 11(3):365–396, 2010.

[105] P. J. Meyer, A. Girard, and E. Witrant. Compositional abstraction and safety synthesis using overlapping symbolic models. *IEEE Trans. on Automatic Control*, 63(6):1835–1841, 2018.

[106] K. Miettinen. *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012.

[107] S. Mouelhi, A. Girard, and G. Gössler. CoSyMA: a tool for controller synthesis using multi-scale abstractions. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pages 83–88. ACM, 2013.

[108] P. Ngatchou, A. Zarei, and A. El-Sharkawi. Pareto multi objective optimization. In *Proc. of the Int. Conf. on Intelligent Systems Application to Power Systems*, pages 84–91, 2005.

[109] M. O'Neill, L. Vanneschi, S. Gustafson, and W. Banzhaf. Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):339–363, 2010.

[110] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam. Fly-by-logic: Control of multi-drone fleets with temporal logic objectives. In *ACM/IEEE Int. Conf. on Cyber-Physical Systems (ICCPS)*, pages 186–197, 2018.

[111] A. Papachristodoulou and S. Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In *IEEE Conf. on Decision and Control (CDC)*, pages 3482–3487, 2002.

[112] I. Papusha, J. Fu, U. Topcu, and R. M. Murray. Automata theory meets approximate dynamic programming: Optimal control with temporal logic constraints. In *IEEE Conf. on Decision and Control (CDC)*, pages 434–440, 2016.

[113] G. Pola, P. Pepe, and M. D. Di Benedetto. Symbolic models for networks of control systems. *IEEE Trans. on Automatic Control*, 61(11):3663–3668, 2016.

[114] R. Postoyan, M. Granzotto, L. Buşoniu, B. Scherrer, D. Nešsić, and J. Daafouz. Stability guarantees for nonlinear discrete-time systems controlled by approximate value iteration. In *IEEE Conf. on Decision and Control (CDC)*, pages 487–492, 2019.

[115] S. Prajna and A. Jadbabaie. *Safety Verification of Hybrid Systems Using Barrier Certificates*, pages 477–492. Springer Berlin Heidelberg, 2004.

[116] S. Prajna and A. Rantzer. Convex programs for temporal verification of nonlinear dynamical systems. *SIAM Journal on Control and Optimization*, 46(3):999–1021, 2007.

[117] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia. Reactive synthesis from signal temporal logic specifications. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pages 239–248. ACM, 2015.

[118] H. Ravanbakhsh and S. Sankaranarayanan. Counter-example guided synthesis of control Lyapunov functions for switched systems. In *IEEE Conf. on Decision and Control (CDC)*, pages 4232–4239, 2015.

[119] H. Ravanbakhsh and S. Sankaranarayanan. Counterexample guided synthesis of switched controllers for reach-while-stay properties. *arXiv:1505.01180*, 2015.

[120] H. Ravanbakhsh and S. Sankaranarayanan. Robust controller synthesis of switched systems using counterexample guided framework. In *Int. Conf. on Embedded Software*, pages 1–10. IEEE, 2016.

[121] H. Ravanbakhsh and S. Sankaranarayanan. A Class of Control Certificates to Ensure Reach-While-Stay for Switched Systems. *arXiv:1711.10639*, 2017.

[122] E. Reichensdörfer, D. Odenthal, and D. Wollherr. Nonlinear control structure design using grammatical evolution and Lyapunov equation based optimization. In *Int. Conf. on Informatics in Control, Automation and Robotics*, pages 55–65. INSTICC, SciTePress, 2018.

[123] E. Reichensdörfer, D. Odenthal, and D. Wollherr. Automated nonlinear control structure design by domain of attraction maximization with eigenvalue and frequency domain specifications. In *Int. Conf. on Informatics in Control, Automation and Robotics*, pages 118–141. Springer, 2018.

[124] G. Reissig, A. Weber, and M. Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Trans. on Automatic Control*, 62(4):1781–1796, 2017.

[125] H. Roehm, J. Oehlerking, T. Heinz, and M. Althoff. STL model checking of continuous and hybrid systems. In *Automated Technology for Verification and Analysis*, pages 412–427. Springer International Publishing, 2016.

[126] M. Z. Romdlony and B. Jayawardhana. Stabilization with guaranteed safety using control Lyapunov–Barrier function. *Automatica*, 66:39 – 47, 2016.

[127] N. F. Roozenburg and J. Eekels. *Productontwerpen structuur en methoden*. Lemma, 1998.

[128] R. Ros and N. Hansen. A simple modification in CMA-ES achieving linear time and space complexity. In *Parallel Problem Solving from Nature*, pages 296–305. Springer Berlin Heidelberg, 2008.

[129] G. Rudolph. Convergence of evolutionary algorithms in general search spaces. In *Proc. of IEEE Int. Conf. on Evolutionary Computation*, pages 50–54, 1996.

[130] G. Rudolph and A. Agapie. Convergence properties of some multi-objective evolutionary algorithms. In *Proc. of the Congress on Evolutionary Computation*, volume 2, pages 1010–1016, 2000.

[131] M. Rungger and M. Zamani. SCOTS: A tool for the synthesis of symbolic controllers. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pages 99–104. ACM, 2016.

[132] C. Ryan and M. Keijzer. An analysis of diversity of constants of genetic programming. In *Genetic Programming*, pages 404–413. Springer Berlin Heidelberg, 2003.

[133] S. Sadraddini and C. Belta. Robust temporal logic model predictive control. In *Annual Allerton Conf. on Communication, Control, and Computing*, pages 772–779, Sep. 2015.

[134] S. Sadraddini and C. Belta. Formal guarantees in data-driven model identification and control synthesis. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pages 147–156. ACM, 2018.

[135] R. G. Sanfelice. On the existence of control Lyapunov functions and state-feedback laws for hybrid systems. *IEEE Trans. Automat. Control*, 58(12):3242–3248, 2013.

[136] R. G. Sanfelice. Robust asymptotic stabilization of hybrid systems using control Lyapunov functions. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, page 235–244. ACM, 2016.

[137] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[138] B. Schürmann and M. Althoff. Guaranteeing constraints of disturbed nonlinear systems using set-based optimal control in generator space. In *Proc. of the IFAC World Congress*, pages 12020–12027, 2017.

[139] B. Schürmann, N. Kochdumper, and M. Althoff. Reachset model predictive control for disturbed nonlinear systems. In *IEEE Conf. on Decision and Control (CDC)*, pages 3463–3470, 2018.

[140] H.-P. P. Schwefel. *Evolution and optimum seeking*. John Wiley & Sons, Inc., 1995.

[141] B. Schürmann, R. Vignali, M. Prandini, and M. Althoff. Set-based control for disturbed piecewise affine systems with state and actuation constraints. *Nonlinear Analysis: Hybrid Systems*, 36:100826, 2020.

[142] I. Sekaj. *Control algorithm design based on evolutionary algorithms*, pages 251–266. iConcept Press, 2011.

[143] I. Sekaj and J. Perkacz. Genetic programming - based controller design. In *2007 IEEE Congress on Evolutionary Computation*, pages 1339–1343, 2007.

[144] R. Sepulchre, M. Jankovic, and P. V. Kokotovic. *Constructive nonlinear control*. Springer Science & Business Media, 2012.

[145] Z. She, B. Xia, R. Xiao, and Z. Zheng. A semi-algebraic approach for asymptotic stability analysis. *Nonlinear Analysis: Hybrid Systems*, 3(4):588 – 596, 2009.

[146] S. Skogestad and I. Postlethwaite. *Multivariable feedback control: analysis and design*, volume 2. Wiley New York, 2007.

[147] A. Solar-Lezama, L. Tancau, R. Bodik, S. Seshia, and V. Saraswat. Combinatorial sketching for finite programs. In *Proc. of the 12th Int.Conf. on Architectural Support for Programming Languages and Operating Systems*, page 404–415. ACM, 2006.

[148] M. Srinivasan, S. Coogan, and M. Egerstedt. Control of multi-agent systems with finite time control barrier certificates and temporal logic. In *IEEE Conf. on Decision and Control (CDC)*, pages 1991–1996, 2018.

[149] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[150] P. Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer US, 2009.

[151] A. Taly, S. Gulwani, and A. Tiwari. Synthesizing switching logic using constraint solving. *Int. journal on software tools for technology transfer*, 13(6):519–535, 2011.

[152] W. Tan and A. Packard. Searching for control Lyapunov functions using sums of squares programming. In *Allerton Conf.*, pages 210–219, 2004.

[153] Y. Tazaki and J.-i. Imura. Bisimilar finite abstractions of interconnected systems. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pages 514–527. Springer Berlin Heidelberg, 2008.

[154] T. Tsuzuki, K. Kuwada, and Y. Yamashita. Searching for control Lyapunov-Morse functions using genetic programming for global asymptotic stabilization of nonlinear systems. In *IEEE Conf. on Decision and Control (CDC)*, pages 5114–5119, 2006.

[155] UNECE Regulation No13. Uniform provisions concerning the approval of vehicles of categories M, N and O with regard to braking, 2016.

[156] A. J. Van Der Schaft and J. M. Schumacher. *An introduction to hybrid dynamical systems*, volume 251. Springer London, 2000.

[157] C. F. Verdier, N. Kochdumper, M. Althoff, and M. Mazo Jr. Formal synthesis of closed-form sampled-data controllers for nonlinear continuous-time systems under STL specifications. *arXiv preprint arXiv: 2006.04260*, 2020.

[158] C. F. Verdier and M. Mazo Jr. Formal controller synthesis via genetic programming. *IFAC-PapersOnLine*, 50(1):7205 – 7210, 2017. IFAC World Congress.

[159] C. F. Verdier and M. Mazo Jr. Formal synthesis of analytic controllers for sampled-data systems via genetic programming. In *IEEE Conf. on Decision and Control (CDC)*, pages 4896–4901, 2018.

[160] C. F. Verdier, R. Babuška, B. Shyrokau, and M. Mazo Jr. Near optimal control with reachability and safety guarantees. *IFAC-PapersOnLine*, 52(11):230 – 235, 2019. IFAC Conf. on Intelligent Control and Automation Sciences (ICONS).

[161] C. F. Verdier and M. Mazo Jr. Formal controller synthesis for hybrid systems using genetic programming. *arXiv preprint arXiv:2003.14322*, 2020.

[162] P. A. Whigham. Grammatically-based genetic programming. In *Proc. of the workshop on genetic programming: from theory to real-world applications*, pages 33–41, 1995.

[163] P. Wieland and F. Allgöwer. Constructive safety using control barrier functions. *Proc. of the IFAC Symposium on Nonlinear Control Systems*, pages 462–467, 2007.

[164] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Trans. on Evolutionary Computation*, 1(1):67–82, 1997.

[165] T. Wongpiromsarn, U. Topcu, and A. Lamperski. Automata theory meets barrier certificates: Temporal logic verification of nonlinear systems. *IEEE Trans. on Automatic Control*, 61(11):3344–3355, 2016.

[166] W. Xiang, P. Musau, A. A. Wild, D. M. Lopez, N. Hamilton, X. Yang, J. Rosenfeld, and T. T. Johnson. Verification for machine learning, autonomy, and neural networks survey. *arXiv:1810.01989*, 2018.

[167] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27):54 – 61, 2015.

[168] S. Yaghoubi and G. Fainekos. Worst-case satisfaction of STL specifications using feedforward neural network controllers: A lagrange multipliers approach. *ACM Trans. Embed. Comput. Syst.*, 18(5s), October 2019.

[169] M. Zamani, A. Abate, and A. Girard. Symbolic models for stochastic switched systems: A discretization and a discretization-free approach. *Automatica*, 55:183 – 196, 2015.

[170] I. S. Zapreev, C. Verdier, and M. Mazo Jr. Optimal symbolic controllers determinization for bdd storage. *IFAC-PapersOnLine*, 51(16):1 – 6, 2018. IFAC Conf. on Analysis and Design of Hybrid Systems (ADHS).

Acknowledgements

This dissertation is the product of 'four' years of research. The content might have shaped me more than I shaped it. In Japanese culture, there is an art of mending broken pottery with gold. Taking this as a metaphor for my PhD journey: it has taken up my parts and enriched me into something new, but it has also been the hammer that broke me in the first place. The pursuit of academic status has come with hard work, long hours until deep in the night, or sometimes even morning, just to be crushed by rejection letters months or even years later. However, one learns to deal with stress, cope with rejection, and put a lot of matters in life in perspective. But these life lessons are not the only golden seams that have mended the cracks in my pottery. I would not have been able to finish this journey without the people around me, either at work, or in my private life.

Starting at the beginning of this project, I would like to thank Gabriel Lopes, Manuel Mazo and Robert Babůska for this opportunity and their subsequent supervision. I would like to thank Gabriel for his guidance during my master thesis; he has played an important role in my pursuit of my doctoral degree. I would like to thank Manuel for his detailed feedback, a no-nonsense style of supervision, pushing me to a higher level, and not to forget, his patience. He has been a great promotor and a true mentor. Finally, I would like to thank Robert for our insightful discussions and his role as co-promotor.

In the first few years, I had the joy of sharing the office with Anqi, the first PhD student of Manuel, who took on the role of unofficial mentor. His hard work and dedication was reflected by him always being in the office, so every time I 'burned the midnight oil', I was never alone. I am grateful for him showing me the ropes and being a tremendous office mate.

Starting their PhD around the same time, Tim and Andy have been great friends throughout these years. To blatantly rip-off the acknowledgements in Tim's dissertation: *the realization that we were all a little lost, made us realize we were not lost at all.*

After Anqi's graduation, I had the great pleasure of welcoming extremely talented new colleagues: Giannis, Gabriel and Daniel, with whom 'Manuel's minions' started to feel like a true group. It has been a great pleasure to discuss our work, daily life, and politics (so much politics!) over coffee and lunch. Of course, I would also like to thank all the post-docs enriching Manuel's group with their expertise, namely Anton, Ivan, Khushraj and Gururaj. I would also like to thank the other PhD students and staff of DCSC for creating such a nice working environment, including, but not limited to, Arman, Twan, Clara, Reinier, Sjoerd, Sebastian, Bart, Wouter, Linda, Vahab and Fahrid. I would also like to thank my master students Rick, Stijn, and Jonathan for their hard work and dedication. I would also like to thank all the secretaries: Erica, Francy, Heleen, Kiran, Kitty, Marieke, Martha and Mascha, for their kind support and the organization of all kinds of social events, such as the regular pub quizzes. Not to brag, but over the years I have accumulated multiple bronze and silver pub quiz medals (not because of my contributions; I was pretty worthless). Nevertheless,

the golden medal still eludes me, but not because of a lack of quizzes! Thank you for making the department well organized, but also a fun place to be.

This dissertation is part of the CADUSY project, supported by NWO domain TTW. I would like to thank our collaborators: Dhruv Khandelwal and Roland Tóth, and the entire user-committee: Koos van Berkel, Bruno Depraetere, Katya Vladislavleva, Sacha Emery and Paul Blank, for their valuable input. The in-depth discussions with Dhruv have always been a great pleasure. To work on a similar project and to be able to discuss with you, has made this journey feel less lonely.

A joy of the PhD has been the traveling around the world. I had the pleasure to have a research stay at the TUM in Munich. I would like to thank prof. Matthias Althoff and his group under the chair of Robotics, Artificial Intelligence and Real-Time Systems, for their welcoming and making me feel right at home. I want to especially thank Matthias, Bastian and Niklas for their collaboration and insights, which have truly enriched my work.

The unsung hero of my writing is my close friend Florian, who, despite having no gain in any sense, painstakingly proofread most of my papers, despite having no background in technology, let alone control. Besides aiding in my work, he, but also Rex, Gawain, Jarno, Dominic, Anshuman, Tim, and Rebecca, helped me brighten my spare time with their company and friendship.

During the tougher times of my journey, I had the pleasure of enjoying the unconditional love and support of my family; Els, Zeno, Jesse, Lucy and Olivier. Finally, I would like to thank Esmee (and all my past relationships) for their love and support throughout the years. They were always a bright beacon of love, joy and hope.

Cees Ferdinand Verdier
Delft, May 2020

List of Symbols

Set Theory

$\mathbb{N}$	Natural numbers
$\mathbb{Z}$	Integers
$\mathbb{Q}$	Rational numbers
$\mathbb{R}$	Real numbers

Logic

φ	Logic formula/ Signal temporal logic formula
ψ, Ψ	Reachset temporal logic formula
ρ	Quantitative semantic of STL
ϱ, P	Quantitative semantics of RTL
$\mathcal{U}_{[a,b]}$	Until operator
$\Box_{[a,b]}$	Always operator
$\Diamond_{[a,b]}$	Eventually operator
$\bigcirc_a$	Next operator

Genetic programming

$\mathcal{F}$	Fitness
$\mathcal{N}$	Nonterminals
$\mathcal{P}$	Production rules
$\mathcal{S}$	start symbol/tree

Hybrid systems

$\mathcal{H}$	Hybrid system
$\mathcal{H}_{\text{cl}}$	Closed-loop hybrid system
$\mathcal{S}_{\mathcal{H}}(I)$	Set of all maximal solutions of $\mathcal{H}$ starting from I
ϕ	Hybrid arc / solution to a hybrid system

C, D Flow and jump set

E Hybrid time domain

F, G Flow and jump map

$F_{\mathrm{ol}}, G_{\mathrm{ol}}$ Open-loop flow and jump map

Continuous systems

η Sampling time of a sampled-data controller

$\mathcal{S}_{\Sigma}(I)$ Set of all solutions of Σ starting from I

Σ Continuous time system

Σ^{sd} Continuous time system with sampled-data controller

ξ Continuous-time trajectory

Certificate-based synthesis

S, I, O Safe, initial and goal sets

V Lyapunov barrier function

Reachability-based synthesis

$R(t)$ Reachable set

Abbreviations

CBF control barrier function. 5

CEGIS Counterexample-guided Inductive Synthesis. 3, 9

CLBF control Lyapunov barrier function. 51, 52

CLF control Lyapunov function. 5

CMA-ES Covariance Matrix Adaptation Evolution Strategy. 29

EA Evolutionary Algorithm. 3, 24, 28

ES Evolution Strategy. 4, 24

GA Genetic Algorithm. 4, 24

GGGP Grammar-Guided Genetic Programming. 8, 24

GP Genetic Programming. 4, 24, 32

LBF Lyapunov barrier function. 34

MOO multi-objective optimization. 28

RL reinforcement learning. 58

RTL reachset temporal logic. 19

SMT Satisfiability Modulo Theories. 5, 20, 32

STL signal temporal logic. 2, 16, 68

TL temporal logic. 16

Curriculum Vitæ

Cees Ferdinand Verdier

02-07-1992 Born in Heemskerk, the Netherlands.

Education

2010—2013 BSc Mechanical Engineering (Cum Laude)
Delft University of Technology
Minor: Robotics
Thesis: Observer based synchronisation control for legged robots with substantially drifting sensors
Supervisor: dr. G. A. D. Lopes

2013—2015 MSc Systems and Control (Cum Laude)
Delft University of Technology
Thesis: Geometric control of a ball balancing robot
Supervisor: Dr. G. A. D. Lopes
Supervisor: Ir. H. Sandee
Supervisor: Ir. G. Heldens

2015–2020 PhD
Delft University of Technology
Thesis: Formal synthesis of analytic controllers
Promotor: Dr. M. Mazo Espinosa.
Copromotor: Prof. dr. R. Babuška

Visiting Scholar

2019 Chair of Cyber Physical Systems (Prof. Althoff)
Technical University of Munich

List of Publications

Journal papers

2. C. F. Verdier, N. Kochdumper, M. Althoff, and M. Mazo Jr., *Formal Synthesis Of Closed-form Sampled-data Controllers for Nonlinear Continuous-time Systems Under STL Specifications*, arXiv preprint arXiv:2006.04260, submitted to Automatica.

1. C. F. Verdier and M. Mazo, Jr., *Formal Controller Synthesis for Hybrid Systems Using Genetic Programming*, arXiv preprint arXiv:2003.14322, a brief version submitted to Automatica.

Conference papers

4. C. F. Verdier, R. Babuška, B. Shyrokau and M. Mazo Jr., *Near Optimal Control With Reachability and Safety Guarantees*, 5th IFAC Conference on Intelligent Control and Automation Sciences (ICONS), 2019.

3. C. F. Verdier and M. Mazo Jr., *Formal Synthesis of Analytic Controllers for Sampled-Data Systems via Genetic Programming*, IEEE Conference on Decision and Control (CDC), 2018.

2. I. S. Zapreev, C. Verdier, M. Mazo Jr., *Optimal Symbolic Controllers Determinization for BDD storage*, 6th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS), 2018.

1. C. F. Verdier and M. Mazo, Jr., *Formal Controller Synthesis Via Genetic Programming*, IFAC-PapersOnLine, 2017.

Abstracts

2. C. F. Verdier and M. Mazo, Jr., *Formal Controller Synthesis Using Genetic Programming*, 39th Benelux Meeting on Systems and Control, 2019.

1. C. F. Verdier and M. Mazo, Jr., *Evolutionary Methods For Formal Controller Synthesis*, 38th Benelux Meeting on Systems and Control, 2018.

Posters

1. C. F. Verdier and M. Mazo, Jr., *Formal Controller Synthesis using Genetic Programming*, 6th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS), 2018.

TU Delft