Optimization Based Verification and Synthesis for Safe Autonomy

by

Shakiba Yaghoubi

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved May 2021 by the
Graduate Supervisory Committee:

Georgios Fainekos, Chair
Heni Ben Amor
Dimitri Bertsekas
Giulia Pedrielli
Sriram Sankaranarayanan

ARIZONA STATE UNIVERSITY

May 2021

ABSTRACT

Autonomous systems should satisfy a set of requirements that guarantee their safety, efficiency, and reliability when working under uncertain circumstances. These requirements can have financial, or legal implications or they can describe what is assigned to autonomous systems. As a result, the system controller needs to be designed in order to comply with these -potentially complicated- requirements, and the closed-loop system needs to be tested and verified against these requirements. However, when the complexity of the system and its requirements increases, designing a requirement-based controller for the system and analyzing the closed-loop system against the requirement becomes very challenging. In this case, existing design and test methodologies based on trial-and-error would fail, and hence disciplined scientific approaches should be considered.

To address some of these challenges, in this dissertation, I present different methods that facilitate efficient testing, and control design based on requirements:

1. Gradient-based methods for improved optimization-based testing,

2. Requirement-based learning for the design of neural-network controllers,

3. Methods based on barrier functions for designing control inputs that ensure the satisfaction of safety constraints.

To my loving parents, Mitra & Abbas,

and to my beloved husband Mohammad.

ACKNOWLEDGEMENT

When I decided to step out of my comfort zone and started my journey to pursue a Ph.D. degree as an international student, I had never expected my life to be filled with so many wonderful people and experiences. I am grateful for receiving a lot of support from my advisor, colleagues, family, and friends. First and foremost, I am thankful to my advisor Prof. Georgios Fainekos who provided a healthy environment for doing high-quality research. My personality as a researcher was shaped to great extent by his enthusiasm and dedication to research. I owe him my ability to do independent research, as he always encouraged me to explore my ideas while helping me avoid research deadlocks with his vision. He was not only an academic advisor but also an understanding friend. None of my achievements during graduate school, including this dissertation, would have been possible without his help. I also would like to acknowledge the supporting funds NSF 1350420, NSF 1319560, NSF 1936997, NSF 1361926, and the NSF I/UCRC Center for Embedded Systems.

I would like to thank the rest of my committee members Prof. Heni Ben Amor, Prof. Dimitri Bertsekas, Prof. Giulia Pedrielli, and Prof. Sriram Sankaranarayanan for their comments and suggestions, and for openly sharing and discussing research ideas. Special thanks to Heni, and Giulia, whose personalities make every research discussion amusing and fun. A special thanks to Dimitri whose vision, knowledge, and support were a blessing to have, and a special thanks to Sriram who has remotely helped me with his broad knowledge.

My gratitude to my colleagues in the Cyber-Physical Systems Lab throughout the years, Bardh Hoxda, Erkan Tuncali, Keyvan Majd, Mohammad Hekmatnejad, Adel Dokhanchi, Kangjin Kim, Joe Campbell, and Sai Krishna Bashetty who created a friendly environment and let many collaborations with great outcomes form. Thanks to my other collaborators Lina Karam, Theodore Pavlic, Logan Mathensen, and Alireza Inanlouganji.

I am grateful to my colleagues at Toyota, Bardh Hoxha, Tomoya Yamaguchi, and Danil Prokhorov for their friendship, support, and guidance throughout my internship at Toyota. While this collaboration was fully remotely due to the pandemic, we were able to achieve

great results together.

I would like to thanks all my friends who have been like my family while far from home. A special thanks to my friend Tahora who welcomed me to the USA, and supported me in the first few weeks.

I wouldn't be here without the unconditional love and support from my parents. Mom, Dad, you are my strength and refuge. Thank you for all you've done for me and for believing in me. I could not challenge myself and overcome my fears of losing without your support. Thanks to my siblings, Fariba, Mohammad, and Maryam for always bringing a smile to my face during the ups and downs. Having you makes me the happiest! I love you! I am grateful to my grandparents for the never-ending love in their hearts. You make the world a better place! Even though I am grateful to all the members of my broader family for their encouragement and support, I would like to particularly acknowledge my aunt Mina, and my parents-in-law for all they did for me.

Finally, and most importantly, I would like to thank my husband, Mohammad. Thank you for walking beside me through all of the ups and downs of life. Thanks for supporting my dreams. Your love and companionship have empowered me, and this hasn't been possible without you. I love you!

TABLE OF CONTENTS

Page

LIST OF TABLES vii

LIST OF FIGURES viii

PREFACE x

CHAPTER

1 INTRODUCTION & MOTIVATION 1

1.1 Literature Review, Contributions & Thesis Overview 2

1.1.1 Automatic Generation of Test Cases using Trajectory Optimization 4

1.1.2 Requirement-based Control Synthesis 7

1.2 Thesis Statement 10

1.3 Summary of Current Contributions 11

2 AUTOMATIC TEST GENERATION FOR TESTING USING OPTIMAL CONTROL BASED METHODS 19

2.1 Introduction 19

2.2 Preliminaries 20

2.2.1 System Models 20

2.2.2 Temporal logic requirements and robustness 22

2.3 Problem Formulation 24

2.4 Solution Approach and Experimental Results 26

2.4.1 Gradient-based Gray-box Falsification of Nonlinear Systems 27

2.4.2 Falsification Using Gradient-based Local Search in Space and Time 29

2.4.3 Falsification by Direct Search in the Function Spaces 34

2.4.4 Falsification of Hybrid Systems 40

2.5 Conclusions and Future Work 46

3 REQUIREMENT-BASED CONTROL SYNTHESIS 47

3.1 Introduction 47

3.2 Training Feedback Neural-network Controllers for TL Requirements 50

CHAPTER Page

3.2.1 Preliminaries 50
3.2.2 Problem Formulation 52
3.2.3 Solution Approach 53
3.2.4 Experimental Results 61
3.3 Feedback Neural-network Controllers for Reach-avoid Specifications using CBFs 69
3.3.1 Preliminaries 69
3.3.2 Control Barrier Functions in presence of Disturbance 71
3.3.3 High order Control Barrier Functions in Presence of Disturbance . 73
3.3.4 Control Optimization Problem with CBF constraints 76
3.3.5 Learning NN Controllers from Control Barrier Functions Using the DAGGER Algorithm 77
3.3.6 Reach Avoid Problem of a Water Vehicle Model 78
3.4 Stochastic Barrier Functions for risk bounding 81
3.4.1 Problem Formulation 81
3.4.2 Stochastic Control Barrier Functions 83
3.4.3 Bounded Risk Using Stochastic Control Barrier Functions 84
3.4.4 Risk Bounded Optimization-Based Control Design 86
3.4.5 Goal Set Reachability 87
3.4.6 Control Design in the Presence of Multiple Unsafe regions 89
3.4.7 Application to Nonholonomic Systems 90
3.4.8 SCBFs and CLFs for Nonholonomic Systems 91
3.4.9 Experimental Results 92
3.5 Conclusion and Future Work 94
REFERENCES 97

LIST OF TABLES

Table		Page
2.1	Experimental Results for the Space-Time Parametrization	34
2.2	Falsification Results of Steam Condenser System with RNN Controller Using Different Search Methods.	39
2.3	Comparing SA and SA+GD Results for the Maneuvering Object Example	45
3.1	Comparison of Training Using the Proposed Approach (LM) and CMA-ES	67

LIST OF FIGURES

Figure Page

1.1 Thesis Overview 10

2.1 Input Perturbation: The Left Plot Shows a Perturbation in Space ($\delta\sigma^{\mathbf{i}}$) and the Right Plot Shows One in Time ($\delta\tau^{\mathbf{i}}$). 30

2.2 HA for Calculating the Sensitivity to the Input Switch Times 30

2.3 Top: GD Increases the Air/Flow Ratio over and Undershoots Causing It to Find a Falsifying Trajectory. Bottom Two: Initial and Falsifying Engine Speed and Throttle Input. 32

2.4 Top: Maneuvering Object Trajectories: GD Gradually Improves the Trajectories and Finally Finds a Satisfying Trajectory. Bottom Two: Maneuvering Object Control Inputs, the Shifts along the Time Axis Are the Result of Time Descent and the Shifts along the F-axis Are the Results of Space Descent. . . . 33

2.5 The Falsification Framework. UR Stands for Uniform Sampling. 37

2.6 Simulink Model for Steam Condenser with Feedback RNN Controller 37

2.7 The System Robustness Is Reduced From .20633 Using a Constant Input $w(T) = 4$ To .00030222 Using the Local Optimal Input Shown in the Bottom Picture. 38

2.8 Trajectories B, A and C Improve Locally by Descending Toward the Unsafe Set, Guard g_{43} and Guard g_{23} Respectively. 41

2.9 Assuming $\tau_{X_0} < \tau_{X_0'}$, Trajectories Are under Different Dynamics for All the Times $t \in [\tau_{X_0}, \tau_{X_0'}]$, Where τ_{X_0} and $\tau_{X_0'}$ Are Transition Times for $\tilde{x}_i(X_0, .)$ and $\tilde{x}_i(X_0', .)$ Respectively. 42

2.10 Shown in the Top: Improving the Robustness of the Trajectories with Respect to the Specification φ_2 From .2950 To .8599. Red Arrows Show the Steepest Ascent Direction. Shown in the Bottom: Trajectories That Do Not Enter the Goal Set Location (Dashed Trajectories Here) Can Still Improve by Descending Toward the Guard Set (Dashed Line at $x_1 = 8$). 44

Figure Page

3.1 The Closed-loop System with Its NN Controller to Satisfy an STL Formula φ 52

3.2 Training Framework 54

3.3 Tangent Sigmoid Function 54

3.4 Vehicle Navigation: The Vehicle Should Visit Goals 1 and 2 in This Order While Avoiding the Unsafe Set Starting from a Set of Different Steering Angles. 64

3.5 Quadrotor Needs to Avoid the Unsafe Set and Reach the Goal Set During a Short Interval of Time. The Mission Is Shown in Blue. 66

3.6 Sample Trajectories in x_0^s, Top Left: After the Global Training, Top Right: Projection of Top Left Plot into the x-y Plane Bottom: Trajectories after Improving the NN Weights Using Back-propagation 67

3.7 Robustness Surfaces Using the Controllers Trained with LM and CMA-ES Approach. 68

3.8 The Invariant Sets of Example 1 for the Worst-case Disturbance $w \in [-0.1, 0.1]$. 76

3.9 Trajectories Initiated From x_0^s As Guided by (a) the Qps As Expert When $w = 0$ and (B) the Trained Nn Controller When Random Disturbance Is Applied to System 80

3.10 Top: Initial Positions of the Ego Vehicle and Traffic Participants. Bottom: Final Position of the Traffic Participants Alongside With the Time-stamped Trajectory of the Ego Vehicle From Start to Finish. Simulation Video Can Be Found At https://youtu.be/hqGe8h1erzA. 92

3.11 Figures Show Control Input Signals, Maximum Upper Bound to p_i^b over All the Traffic Participants, Minimum $h_i(X)$ over All the Traffic Participants Respectively. 93

PREFACE

This dissertation is the result of my five years of work at the School of Computing, Informatics, and Decision Systems Engineering, at the Arizona State University. This thesis unifies the theoretical background and the notation for several of my published papers and highlights my contributions there toward the design and analysis of safe Cyber Physical Systems (CPS). The thesis consists of two closely related topics that support safety of Cyber-Physical Systems.

In Chapter 2, I provide methods to accelerate falsification (finding an undesired system behavior w.r.t a given requirement) using local optimization for different classes of systems and under different assumptions.

In Chapter 3, I present methods for control synthesis that facilitate real-time computation. In Sections 3.1, and 3.2 I present two different methods for training neural network controllers: one based on general temporal logic formulas, and the other one which is more efficient based on reach-avoid specifications. In Section 3.3, I study the risk-bounded reach-avoid problem for stochastic systems and derive conditions that facilitate designing less conservative, yet risk-limited control policies. Finally, while most of the material in this dissertation has been reviewed by multiple referees and collaborators, typos or mistakes may still exist in it. Please report any mistakes you may find to: *syaghoub at asu.edu*

Shakiba Yaghoubi

May 2021

Chapter 1

INTRODUCTION & MOTIVATION

Autonomous Mobile Systems (AMS), industrial and medical robots, drones and smart grids are all examples of complex control systems that are getting more and more autonomous everyday. The virtues of autonomous systems are that they can expand upon human capabilities in terms of strength and speed, precision. Autonomous mobile systems for instance can go into places and situations where humans cannot [112]. Autonomous vehicles, on the other hand, can make mobility safe, secure, affordable, sustainable and accessible for all [18]. All such systems have strict performance and safety requirements. These requirements can have financial or legal implications, or they can describe a task the system should accomplish. Reach-avoid specifications are an example of such tasks that naturally arise in many applications, such as AMS motion planning, multi-agent coordination, and spacecraft autonomy. These specifications require completing a start-to-goal motion while complying with the safety constraints. Other systems, however, may need to comply with more complicated system requirements demanding to reason about nested logical, temporal, and spatial properties.

Control systems used in industrial domains are in general known as Cyber-Physical Systems (CPS). These systems are characterized by both continuous and discrete dynamics, with numerous subsystems interacting with each other in complex ways to enable their autonomy. This complexity makes the requirement-based design, construction, and verification of CPS a challenging problem that needs to be addressed by a cross-disciplinary community of researchers and educators.

There are mainly two ways to proceed with the system design based on requirements [20]:

1. **Design/Analysis**: An initial design is made based on experience, knowledge, or ob-

servations from a limited amount of data, etc. Then, the system is analyzed to determine whether the system satisfies the design requirements. The analysis is performed either 1) formally using mathematical proofs or 2) by testing the system against the requirements to see if there is any adverse condition that does not meet the design requirement with respect to which the design needs to improve.

2. **Synthesis**: The design method in this case is rigorous, and it automatically or semi-automatically constructs the system based on the design requirement, as well as the system's physical and input constraints to achieve a correct-by-construction design.

The main advantage of the latter approach is that after the design is completed the system is guaranteed to meet the requirements. However, such methods are not known for many real-world applications and, hence, the design/analysis cycle is usually used in industrial settings. The formal verification of the analysis cycle is a challenging problem for most CPS. Exhaustive verification methods either work on small problem instances or on restricted classes of systems, e.g, linear systems (see [79] for an overview of some of these challenges). Hence, semi-formal verification using testing is primarily used in the industry to increase the system's reliability with respect to given requirements of interest.

To address some of the challenges of the requirement-based design, in this thesis we focus on the following problems:

1. Optimal automatic test generation to reveal system adversaries as soon as possible,
2. Control synthesis for some classes of systems and requirements.

1.1 Literature Review, Contributions & Thesis Overview

As reflected in the previous section, the problems of testing, verification, and synthesis of autonomous systems are challenging and of utmost importance at the same time. Reported accidents [90, 54, 128] and recalls [25, 115] in recent years shows that there is a need for better testing, verification, and synthesis of CPS. Simply relying on a trial-error approach in

the design of these systems requires extensive time and high costs. The environments these systems operate in and the systems them-self involve many uncertainties despite which the system should behave properly. For instance, an autonomous vehicle is estimated to need billions of miles of test-driving so that its catastrophic failure rates becomes less than one per hour [58, 84].

To simplify the synthesis and analysis of CPS, Model-Based Design (MBD) is used for CPS prototypes. Using models, we can simplify the design, focus on the critical components of the system, and simulate the system to verify properties or find bugs as soon as possible. Hence, in order to make it possible to design the system and apply formal or semi-formal verification methods on the MBD, one needs to use mathematical formalization to specify the physical models. The mathematical model may be a non-stochastic model, such as a nonlinear dynamical model [80], a switched system model [95], or a hybrid system model [11], or it may be a non-deterministic system that accounts for possible uncertainties with stochastic nature such as a nonlinear stochastic system [87], or a stochastic hybrid system [106]. A dynamical model is represented by a system of ordinary differential equations (ODE) that in general may be nonlinear. Switched and hybrid systems model both the continuous and discrete states of the system and describe the evolution of the continuous states, as well as the changes that happen in the discrete components of the system [17]. The discrete components may represent different stages of computation or different modes of the system. To model these systems, one can use hybrid automata [124, 12]. On the other hand, stochastic systems can be modeled using Stochastic Differential Equations (SDE) [104, 106]. Stochastic differential equations consider disturbances and uncertainties with stochastic characteristics in the system's behaviors to represent uncertainties in the environment and possible errors in the modeling.

The physical models that we consider in this thesis for semi-formal verification (testing) are deterministic hybrid automata and nonlinear ODEs. The models we consider for requirement-based synthesis are deterministic nonlinear differential equations, and Stochas-

tic Differential Equations (SDE).

1.1.1 Automatic Generation of Test Cases using Trajectory Optimization

Several different tools [53, 27, 72, 59, 56, 83] have targeted verification of linear, nonlinear and hybrid systems. Despite recent advances in these verification tools, verification remains a challenging problem [66, 79] for general classes of models. In fact, verification tools are still limited in terms of the complexity of the models or the type of external inputs they can handle. Many of these tools suffer from exponential growth of complexity when system's dimension increases [51, Section 4.2]. Hence, semi-formal verification using testing is primarily used in the industry to increase the system's reliability w.r.t given requirements of interest. The problem of finding the system behaviors that violate given requirements is called a *falsification* problem. In a falsification problem, the system is tested against possible uncertainties that are applied to it, e.g., uncertainties in its initial condition, or exogenous inputs like wind gusts or user inputs, e.g., gas pedal. In Section 2.1 we will review some of the related works on the falsification problem.

The requirements of an autonomous system may include safety, reachability, or stability properties, but they can also be more general than that and include nested logical, spatial, and temporal specifications. To evaluate the system-level behavior of CPS w.r.t general complicated requirements, these requirements need to be specified in a logical formalism, e.g. Metric Temporal Logic (MTL) [99], or Signal Temporal Logic (STL) [85]. Notably, MTL and STL (collectively abbreviated as TL) are equivalent under simple assumptions (see [20, 32] for more information) and they are both very powerful for specifying complicated properties on real-valued signals.

Robustness of TL [50, 38] assigns a real value to signals based on how well they satisfy or falsify the TL requirement. A positive sign of this real value shows the satisfaction of the TL requirement and a negative sign of it shows violation. The absolute value of the robustness value shows how close or far the signal is to satisfying or violating the requirement. In this thesis, when it is necessary to formalize system requirements, we use STL semantics since

they're easier to analyze and differentiate.

Due to the properties of the TL robustness function, assuming that a falsifier (violating behavior) exists, a falsification problem can be solved by considering the TL robustness of a system trajectory/behavior as the objective function and solving the minimization problem by searching over the space of possible system trajectories as a result of uncertainties applied to the system. Several tools such as [16, 36, 146] exist that perform testing using optimization-guided falsification. S-TaLiRo for instance includes various global optimization methods such as Simulated Annealing (SA) [102], genetic algorithm [15], and stochastic optimization with adaptive restarts [101]. Due to the complexity and unclear semantics of simulators such as Matlab and Simulink [31], all these tools treat the system as a black-box and just observe the input/output behavior of the system via simulations. Consequently, these methods neglect some of the advantages that can be taken from the – partial – knowledge of the system model. Despite previous methods, there are works such as [7, 1, 134] that take this information into consideration.

The optimization-guided falsification problem is inherently difficult to solve, since:

1. Robustness of TL is, in general, a non-convex, non-differentiable function.

2. The robustness function is a function of the test inputs through the system model and, hence, it is arduous to predict how changing the test inputs would affect the value of the robustness function. This becomes even more challenging if we have partial to no information about the system model.

3. The system may need to be tested against exogenous inputs which are time-varying signals. In this case, the search space is an infinite-dimensional function space that many existing optimization methods cannot handle.

Despite the aforementioned challenges, it is desired to improve the testing stage performance to decrease the necessary time and cost of the testing phase and find any undesired behaviors as fast as possible and with the least number of system inquiries (since these

inquiries may be time-consuming). Hence, an automated framework with an optimized search algorithm that gets help from efficient optimization strategies is necessary. In this thesis, we address some of these challenges and work on some efficient methods for solving the optimization-guided falsification problem.

In [1], the first steps were taken towards addressing the problem of reducing the number of simulations for the TL falsification problem. This work deals with nonlinear systems without exogenous inputs where the search space is only the set of initial conditions of the system. In [3], authors extend their work to nonlinear systems with exogenous inputs where the search space was the set of piece-wise constant input signals (whose switching times were constant and given). In both of these works the system model is considered to be fully known and directions along which TL robustness is reduced – descent directions – are computed.

In this thesis, we extend the previous works in the following directions.

- In [134], we consider a gray-box model of the nonlinear system rather than a white-box model.
- In [135], we deal with a white-box hybrid automaton that models CPS.
- In [136], we extend the class of exogenous input signals from piece-wise constant to piece-wise constant with varying switching times.
- In [137], we work with the exogenous input signal without parametrizing them, and we directly search the infinite-dimensional search space of the functions.

In all the above works, descent changes in the initial conditions and exogenous inputs are computed to locally search for conditions with lower robustness values. The local search can be combined with global search algorithms or coverage-based algorithms [39] to provide asymptotic convergence or coverage.

1.1.2 Requirement-based Control Synthesis

In this thesis, we focus on control synthesis for deterministic and stochastic system models, with either general TL requirements or reach-avoid requirements.

Control synthesis w.r.t TL requirements: Designing controllers that enforce the closed-loop system to satisfy given TL requirements has been studied before [86]. The aforementioned tools, S-TaLiRo, and Breach can be used offline to design open-loop controllers that satisfy given requirements when the robustness maximization problem is solved instead of the robustness minimization problem. Other works study designing strategies that satisfy TL requirements based on Model Predictive Controllers (MPC) using Mixed Integer Linear Programs (MILP) as in [109]. Designing open-loop strategies using randomized tree search has also been considered in [127].

In order to be used online, open-loop controllers need to be designed and stored in advance, and MPC controllers need to be computed at the run-time. However, computing MPC controllers for high-order systems include solving large optimization problems at each step [67], so in general, they do not scale to higher-order systems since their run-time computation is challenging. On the other hand, storing the results of the offline computation of open-loop or MPC controllers (e.g., as a lookup table) requires too much memory [78], so alternative approaches like using Neural networks (NN) to approximate controllers [113, 77], and learn optimal policies offline [61] has been considered.

Recently, Reinforcement Learning methods that make policy learning w.r.t TL specifications possible have also been studied, see e.g., [57, 94]. Despite the recent advances using Reinforcement Learning methods, designing a reward function that represent the requirements is still an ongoing research topic [8, 93]. Furthermore, providing guarantees for the behavior of the systems trained using them is challenging. This is due to the complicated behavior of the Neural Networks which are used in these systems as function approximators.

Due to the aforementioned challenges in designing optimization-based controllers online and reinforcement learning methods that enforce TL requirements, in [139], we study the

problem of designing feedback Neural Network (NN) controllers for deterministic nonlinear systems that are trained directly with the TL robustness function as the objective function to be maximized. To further improve and robustify the design, the closed-loop system is tested to find adversarial examples (falsifying behaviors) and the NN is retrained according to these examples (see Fig. 3.2 for an overview of the framework).

Control synthesis w.r.t reach-avoid requirements: Reach-avoid specifications naturally arise in many applications, such as autonomous and assisted driving, multi-agents coordination, and autopilot aircraft. These specifications require completing a start-to-goal motion while complying with the safety constraints. The problem has been studied for linear, nonlinear, as well as stochastic models using different methods [70, 49]. In many applications, the problem is approached by computing forward reachable sets [55, 9, 100]. Despite the efforts for finding more efficient methods for their computation [81], reachable sets are still difficult to compute in real-time.

Inspired by Lyapunov functions [80], Control Barrier Functions (CBF) have been used recently for the synthesis of safety-critical systems without the need for the difficult task of computing the system's reachable sets [133]. They have enabled the design of provable safe feedback controllers for several different systems such as adaptive cruise control [14], bipedal robot walking and long term autonomy [13], merging control in a traffic network [132], and finite-time convergence in a multi-agent system [122].

The CBF separates the safe and unsafe set of states, and computes constraints based on its derivatives to control its evolution and prevent entering the unsafe states. These constraints along with constraints based on the Control Lyapunov Functions (CLF) are typically part of a constrained Quadratic Program (QP) whose solution computes sub-optimal control inputs that consider other performance objectives such as reachability.

If the relative degree of the CBF w.r.t the control inputs (the number of times we need to differentiate from the CBF until the control input appears) is higher than one, High Order CBFs (HOCBF) can be used [131] to compute safe controllers. In the presence

of "actuation errors" with hard bounds on their magnitude, input-to-state safety using CBFs has been studied in [82]. In this thesis (Section 3.4), we consider a more general class of external inputs (that can represent measurement, or actuation errors as well as environmental inputs like wind gusts and water currents) and compute constraints on the control input to reject the worst-case disturbance and guarantee safety [140]. The nonlinear programs that compute control inputs that achieve the reach-avoid specification may not be solvable in real-time. Hence, to accelerate control computation, we use imitation learning to train a NN to approximate the solutions of the nonlinear program. The idea of training a NN using supervised reinforcement learning to approximate Model Predictive Controllers (MPC) is used in [144] and [26], too.

When the disturbance has stochastic characteristics, barrier functions should be considered in a stochastic setting. Stochastic Barrier Functions (SBF) have been used in [107, 74] for verification of safety and temporal logic properties of stochastic systems. The authors in [116] use SBFs for finite-time stochastic system verification and feedback control design through solving sum-of-squares programs. However, finding such a closed-loop controller for more complicated systems and environments, and when uncertainty is higher, is not possible. Instead, one can solve QPs with constraints based on SBFs to compute control inputs in real-time. Using QPs to design real-time control inputs for stochastic systems that maximize the probability of invariance of a set C has been studied in [28] using CBFs for complete and incomplete information and in [117] using high-relative degree SBFs. The derived conditions in these works may not be feasible in many applications, and in others, they may result in very conservative control actions. The reason is that the conditions are designed to zero out the probability of eventually entering an unsafe set as $t \to \infty$ which is very conservative for many applications. To overcome this problem, in this thesis (Section 3.4.1), we derive certificates on the control inputs based on SBFs to bound the probability of "a failure in finite-time" to the desired value. This will allow for designing less conservative control inputs while managing the risk. We use these constraints in addition to

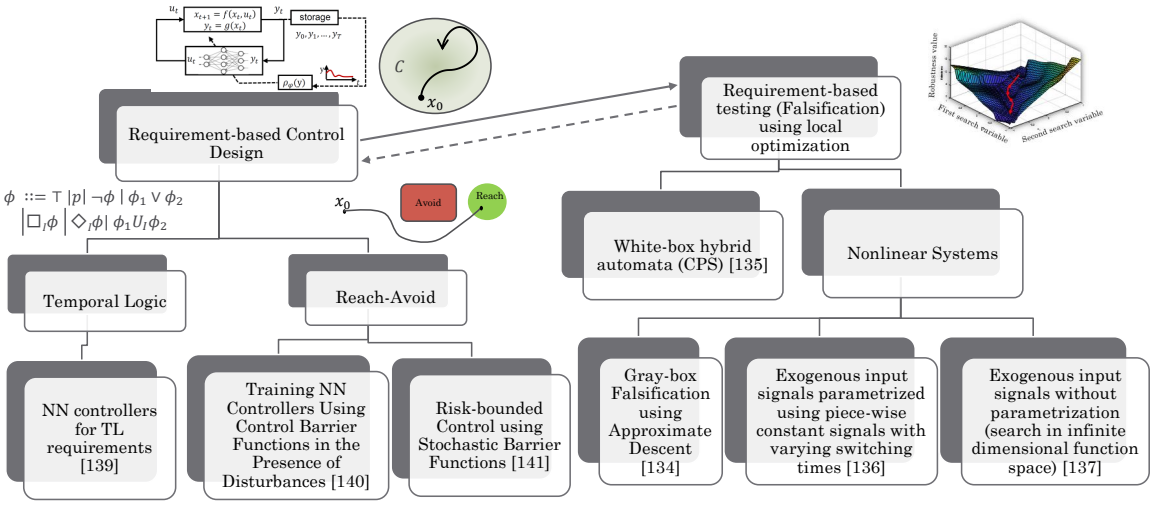


Figure 1.1: Thesis Overview

Lyapunov-based constraints in a quadratic program to solve the reach-avoid problem while "bounding the risk".

1.2 Thesis Statement

As mentioned in the previous section, in the interest of having reliable safe autonomous systems, in this thesis, we address some of the challenges related to the problems of 1) falsification of TL requirements of autonomous systems, 2) Requirement-based control synthesis. An overview of the thesis and its topics is represented in Fig. 1.1.

In Chapter 2, we propose methods based on trajectory optimization to solve the optimization-guided falsification problem more efficiently. We consider the problem under different assumptions about the system model, and the search space. In summary, we extend previous works in the following directons:

- In [134], we work with a gray-box model of the nonlinear system rather than a white-box model.
- In [135], we deal with a white-box hybrid automaton that models CPS.
- In [136] we extend the class of exogenous input signals from piece-wise constant to

piece-wise constant with varying switching times.

- In [137], we deal with the exogenous input signal without parametrization. In this work we directly search the infinite-dimensional search space of the functions.

In Chapter 3, we address some of the challenges in the control synthesis problem w.r.t general TL or reach-avoid requirements for deterministic or stochastic system models:

- In [139], we provide a framework for training NN controllers with TL rewards. NNs can compute control actions quickly and they are very efficient for use in real-time.
- In [140], we train NNs using a more efficient training method based on imitation learning for reach-avoid specifications using CBFs. We also consider uncertainties with hard bounds on their magnitude in the system and come up with certificates that reject them.
- In [141], we consider stochastic systems and compute certificates that guarantee the *probability of an imminent unsafe behavior* is bounded to a given desired value. This allows for computing less conservative control inputs while managing the risk.

1.3 Summary of Current Contributions

My contributions to the safety analysis and synthesis problems of CPS are summarized below:

Automatic Generation of Test Cases using Trajectory Optimization

What follows summarizes some of my contributions related to the first topic:

- **Shakiba Yaghoubi** and Georgios Fainekos, "Hybrid approximate gradient and stochastic descent for falsification of nonlinear systems," in *American Control Conference (ACC), 2017* [134].

In this paper, we provide an alternative method for locally minimizing the robustness value using approximate descent directions that do not require structural information about the system's model. This gray-box falsification method uses the linearized model of the system computed along system trajectories to approximate descent directions. These linearized models can be readily computed in most MBD tools like Simulink [31]. Later, we prove that this approximation can get arbitrarily close to the analytically computed descent directions. Finally, the search using the gradient descent method is combined with SA to bring together the advantages of both methods, and using experimental results we show the effectiveness of the proposed method.

- **Shakiba Yaghoubi** and Georgios Fainekos, "Local descent for temporal logic falsification of cyber-physical systems," in *Seventh Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems, 2017* [135].

  The main contribution in this paper is the extension of the results of [3] to computing local descent directions for falsification of TL specifications for hybrid systems. The extension is nontrivial since as discussed in the paper, the sensitivity analysis is challenging in the case of hybrid systems. To represent hybrid systems we use hybrid automata [124, 10] with non-linear dynamics in each mode and external inputs. Several examples of hybrid automata for which such a descent direction for TL specifications can be computed are presented. And finally, since the computed descent directions can only point toward local reduction of TL robustness, descent direction computations are combined with a stochastic optimization engine to improve the overall system falsification rate.

- **Shakiba Yaghoubi** and Georgios Fainekos, "Falsification of temporal logic requirements using gradient based local search in space and time," *IFAC-PapersOnLine* [136]

  The main contribution of this paper is that we provide a space-time parameterization

for the time-varying exogenous inputs to the system, and use gradient descent (GD) to adaptively change the amplitude and the switch time of the signal to find optimal cases to be tested against the system specifications. Because of the complexities that arise in the sensitivity calculation for hybrid systems [135], and to avoid unnecessary technicalities, the method is developed assuming that the system dynamics are smooth and nonlinear. However, the extension of the results in this paper to hybrid systems is easily achievable with the analysis in [135], and one of the studied benchmarks in the paper is also a hybrid system.

- **Shakiba Yaghoubi** and Georgios Fainekos, "Gray-box adversarial testing for control systems with machine learning components," in *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC), 2019* [137]

  Unlike our previous works [134, 136] in which the exogenous input signal is parameterized using a finite number of parameters, in this work the input signal is calculated using an optimal-control approach which searches directly in the infinite search space of the input functions. As an application, the method is used for adversarial test generation (falsification) for control systems with Recurrent NNs in the loop. It is shown experimentally that the framework vastly outperforms black-box system testing methods. Namely, in our case study, the proposed framework always returns falsifications when the black-box methods fail to do so.

Requirement-based Control Synthesis

What follows summarizes my contribution related to the second topic:

- **Shakiba Yaghoubi** and Georgios Fainekos, "Worst-case satisfaction of STL specifications using feed-forward neural network controllers: a Lagrange multipliers approach," in *ACM Transactions on Embedded Computing Systems (TECS)*, 2019. [139]

In this work, a state/output feedback NN controller is designed to satisfy system properties specified in STL. To improve the performance of the NN, we test the closed-loop system (including the NN) against the STL requirements. Any discovered adversarial examples in the testing phase are added into the training set and used to retrain the NN. This will allow efficient retraining using rare and important samples which may not be captured even with a dense sampling. The NN is trained using a gradient-based improvement formulation based on a state-based Lagrange multipliers approach.

- **Shakiba Yaghoubi**, Georgios Fainekos, and Sriram Sankaranarayanan, "Training neural network controllers using control barrier functions in the presence of disturbances" in *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020. [140]

  In this work, we first derive conditions on a barrier function that keep the system safe despite worst-case disturbances. We combine these constraints with constraints based on Lyapunov functions in a nonlinear program to compute control inputs that guide the system to a goal set while preserving safety. We argue that these nonlinear programs may be difficult to solve in real-time. Hence, we propose a framework to train state/output feedback NN controllers using imitation learning and the results of the nonlinear programs as expert demonstrations to satisfy reach-avoid specifications. The proposed framework is illustrated using an example of a water vehicle subject to water currents.

- **Shakiba Yaghoubi**, et al. "Risk-bounded Control using Stochastic Barrier Functions", in *IEEE Control Systems Letters*, 2020. [141]

  In this work, we design real-time controllers that react to uncertainties with stochastic characteristics and bound the probability of a failure in finite time to a given desired value. SCBFs are used to derive sufficient conditions on the control input that bound the probability that the states of the system enter an unsafe region within a finite

time. These conditions are combined with reachability conditions and used in an optimization problem to find the required control actions that lead the system to a goal set. We illustrate our theoretical development using a simulation of a lane-changing scenario in a highway with dense traffic.

Other Contributions

In this section, we summarize my other works related to the *automatic generation of test cases for finding falsifying system behaviors* which we will not be presented in detail in this thesis.

- Cumhur Erkan Tuncali, **Shakiba Yaghoubi**, Theodore P. Pavlic, and Georgios Fainekos, "Functional Gradient Descent Optimization for Automatic Test Case Generation for Vehicle Controllers", *IEEE International Conference on Automation Science and Engineering (CASE)*, 2017 [126].

  In this paper, we use the results from [3, 134] to design a hierarchical test generation framework that uses analytical functional gradient computations to optimize a system performance function. Starting from a random location, a descent direction is computed on a simplified model of the system dynamics, on which the gradients can be analytically computed. Then, the bisection technique is used to search for a minimal performance on the simulations of a high-fidelity model of the system. The system performance evaluations from the simulation outputs are used to guide the search by changing the bisection step size. Once a minimal performance is found with the bisection technique, the whole process is restarted at another random location which is selected by low-discrepancy sampling. A full-range adaptive cruise control from literature has been implemented as a case study to evaluate the results of this work.

- Adel Dokhanchi, **Shakiba Yaghoubi**, Bardh Hoxha, and Georgios Fainekos, "Vacu-

ity aware falsification for MTL request-response specifications," in *IEEE Conference on Automation Science and Engineering (CASE)*, 2017 [34].

This paper improves the search for falsifying system behaviors by focusing on vacuous signals. Vacuous signals cause the system testers to have a false sense of system correctness. Therefore, it is dangerous for CPS verifiers to not identify vacuous signals during testing. On the other hand, Request-Response requirements are typically used in CPS requirements. Falsification of Request-Response requirements is more challenging than other types of requirements. This is because Request-Response requirements have "if-then-else" components and if a signal does not satisfy the "if" part of the requirements, it will vacuously satisfy the "if-then-else" specification. In this paper, S-TaLiRo testing framework was modified so that it first satisfies the "if" statement and then, falsify the "if-then-else" specification. It was shown that the proposed solution can drastically improve the falsification framework for some of the benchmarks

- Logan Mathesen, **Shakiba Yaghoubi**, Giulia Pedrielli, and Georgios Fainekos, "Falsification of cyber-physical systems with robustness uncertainty quantification through stochastic optimization with adaptive restart," in *International Conference on Automation Science and Engineering (CASE)* 2019, [101].

  In this paper, an instance of the Stochastic Optimization with Adaptive Restart (SOAR) framework is implemented in the S-TaLiRo tool [16] for falsification of system requirements. The application of SOAR for falsification not only shows better falsification rates than existing benchmarking techniques but also provides confidence intervals over the predicted minimum system robustness when the system is unable to be observably falsified. This is the first time a falsification method provides uncertainty quantification on the minimum predicted robustness. This is an important benefit for practitioners since they can now decide whether further testing is needed.

- Mohammad Hekmatnejad, **Shakiba Yaghoubi**, Adel Dokhanchi, Heni Ben Amor, Aviral Shrivastava, Lina Karam, and Georgios Fainekos, "Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic," in *International Conference on Formal Methods and Models for System Design*, 2019, [65].

  The work in [119] provides an extensive study on the formalization of rules for a safe-driver model referred to as Responsibility-Sensitive Safety (RSS) model. In this paper, it is demonstrated that the RSS model can be encoded in assume-guarantee STL requirements. Due to space limitations, only two RSS scenarios are presented in detail, but the rest of the scenarios can be similarly encoded in STL. To motivate how the resulting STL requirements could be used in practice, multiple real driving data scenarios were monitored offline over some of the RSS rules written in STL. Interestingly, it is observed that the RSS rules are not frequently violated by human drivers assuming fast reaction times. The code for the case study is distributed with S-TaLiRo.

- In the following four works, we have presented the results of friendly competitions for solving challenging benchmarks in the falsification of dynamical systems' requirements.

  - A. Dokhanchi, **S. Yaghoubi**, B. Hoxha, and G. E. Fainekos, "Arch-comp17 category report: Preliminary results on the falsification benchmarks." in ARCH@ CPSWeek, 2017. [33]
  - Adel Dokhanchi, **S. Yaghoubi**, B. Hoxha, G. Fainekos, G. Ernst, Z. Zhang, P. Arcaini, I. Hasuo, and S. Sedwards, "Arch-comp18 category report: Results on the falsification benchmarks," 2018. [35]

– G. Ernst, P. Arcaini, A. Donze, G. Fainekos, L. Mathesen, G. Pedrielli, **S. Yaghoubi**, Y. Yamagata, and Z. Zhang, "Arch-comp 2019 category report: Falsification," EPiC Series in Computing, 2019. [47]

– G. Ernst, P. Arcaini, I. Bennani, A. Donze, G. Fainekos, G. Frehse, L. Mathesen, C. Menghi, G. Pedrinelli, M. Pouzet, and **S. Yaghoubi**. "ARCH-COMP 2020 Category Report: Falsification," EPiC Series in Computing, 2020. [46]

Chapter 2

AUTOMATIC TEST GENERATION FOR TESTING USING OPTIMAL CONTROL BASED METHODS

2.1 Introduction

Simulation-guided strategies for testing system properties are devised based on the notion of falsification. Several different methods for solving the simulation-guided falsification based on optimization [16, 36, 146, 24], classification and coverage [5], tree search [48], and learning [7, 120] exist. All these methods try to intelligently create test instances that reveal undesirable system behaviors with respect to a system requirement. For example S-TaLiRo ([16]) and Breach ([36]) are Matlab/C++ toolboxes that use global optimization solvers to search for counterexamples to system requirements expressed as TL formulas. To search over space of the varying time inputs to the system, the tools require a parameterization representation of the input signal. And they search the space of the uncertain parameters to find trajectories with minimal robustness values with respect to the given requirement. Different results show that testing can benefit from combining global search methods with local search hence works presented in this Chapter as well as works presented in [30, 88] focus on efficient strategies for local search and its combination with global search.

The first steps toward model-based falsification through local search using derivatives of the robustness function and system model were taken in [1, 3] in which the system is assumed to be a white-box nonlinear system, and possible exogenous inputs are considered to be piece-wise constant. In [92], authors use the well-known results in optimal control theory for nonlinear systems to search the input's function space of the system and locally minimize a cost function described in the form of standard optimal control cost functionals. Another related work that studies the search-based falsification of Hybrid systems is the multiple-shooting optimization technique [146].

In the rest of this Chapter we present our contributions related to to problem of falsification with the help of local optimization.

2.2 Preliminaries

2.2.1 System Models

Autonomous Continuous Nonlinear Systems with Exogenous Inputs

An autonomous nonlinear dynamical system with exogenous inputs can be described as:

$$\Sigma: \quad \dot{x}(t) = f(x(t), w(t)), \qquad x(0) = x_0 \in X_0 \tag{2.1}$$

where t denotes the time, $x(t) \in \mathbb{R}^{n_x}$ is the system state at time t, $w : \mathbb{R}_+ \to W$ is the exogenous input function that given a point in time t, returns a value $w(t)$ in the set $W \subset \mathbb{R}^{n_w}$, and x_0 is the system's initial condition which is assumed to be in $X_0 \subset \mathbb{R}^{n_x}$. The function $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ is the system flow, which is assumed to be at least one-time differentiable.

In most of the cases, we cannot find a closed form solution for the system Σ. However, assuming that the simulation time T is bounded, given an initial condition x_0, and an input w, we can find a numerical solution to the system Σ, which we denote with $\mathbf{x}(x_0, w)$ in order to emphasize its dependency on the initial condition and input. Note that $\mathbf{x}(x_0, w)$ is a sequence of tuples $\{\big(t, x(t)\big)\}$ where 't values' are chosen by the numerical solver. We show the value of the trajectory at time t with $\mathbf{x}(x_0, w, t)$, i.e, $\mathbf{x}(x_0, w, t)$ is the value $x(t)$ that states take at time t emphasizing the fact that the system has been initiated from x_0 and the input w was applied to it. If it's clear from the context, we may drop (x_0, w) from $\mathbf{x}(x_0, w)$ for the sake of brevity and write $\mathbf{x}$ for short.

Autonomous Hybrid Systems with Exogenous Inputs

Hybrid automaton (HA) is a model that facilitates specification and verification of hybrid systems. A hybrid automaton is specified using a tuple

$$\mathcal{H} = (L, X, W, Inv, \mathcal{E}, \Sigma) \tag{2.2}$$

where $L \subset \mathbb{N}$ is the set of discrete states or locations that the system switches through (each location attributes different continuous dynamics to the system), $X \subseteq \mathbb{R}^{n_x}$ is the continuous state space of the system, $W \subseteq \mathbb{R}^{n_w}$ is the space of system's exogenous input signals, and $Inv : L \rightarrow 2^{X \times \mathbb{R}^+}$ assigns an invariant set to each location. Also, $\mathcal{E}$ is a set of tuples $(E, Guard, Reset)$ that determine transitions between locations. Here, $E \subseteq L \times L$ is the set of control switches, $Guard : E \rightarrow 2^{X \times \mathbb{R}^+}$ is the guard condition that enables a control switch (i.e, the system switches from l_i to l_j when $(x(t), t) \in X \times \mathbb{R}^+$ satisfies $Guard((l_i, l_j))$) and, $Reset : E \times X \rightarrow L \times X$ is a reset map that given a transition $e = (l_i, l_j) \in E$ and a point x for which $Guard(e)$ is satisfied, maps x to a point in the invariant set of the next location $Inv(l_j)$. Finally, Σ defines the continuous dynamics in each location $l \in L$:

$$\Sigma(l): \quad \dot{x(t)} = f_l(x(t), w(t)), x(t) \in X, w(t) \in W \tag{2.3}$$

For more information about Hybrid systems please refer to [124], [11], [12]. We let $H = L \times X$ denote the *hybrid* state space of the hybrid automaton $\mathcal{H}$. $H_0 \subseteq H$ will denote a set of *intial conditions* of the system.

Assuming that the initial location l_0 is fixed, the numerical solution to $\mathcal{H}$ starting from a point $h_0 = (l_0, x_0) \in H_0$ and under the input $w(t) \in W$ is denoted with a *hybrid trajectory* $\eta(x_0, w) = (l(x_0, w), \mathbf{x}(x_0, w))$. At each point in time t, this trajectory points to a pair *(control location, state vector)*: $\eta(x_0, w, t) = (l(x_0, w, t), \mathbf{x}(x_0, w, t))$, where $l(x_0, w, t)$ is the location at time t, and $\mathbf{x}(x_0, w, t)$ is the continuous state at time t.

We can write the dynamical equations for the continuous system trajectory as:

$$\mathbf{x}(x_0, w, 0) = x_0$$

while $(\mathbf{x}(x_0, w, t), t) \in Inv(l)$:

$$\dot{\mathbf{x}}(x_0, w, t) = f_l(\mathbf{x}(x_0, w, t), w(t)) \tag{2.4}$$

if $\mathbf{x}(x_0, w, t^-) \in Guard((l_i, l_j))$ and $(\mathbf{x}(x_0, w, t^+), t) \in Inv(l_j)$:

$$\mathbf{x}(x_0, w, t^+) = Re((l_i, l_j), \mathbf{x}(x, w, t^-)) \tag{2.5}$$

The time in which the location l and consequently the right-hand side of the equation (2.4) changes, are called *transition times*. In order to avoid unnecessary technicalities, in the above equations, transition times are demonstrated as t^- and t^+, where t^- is the time right before the transition and t^+ is the time right after that. When we consider the trajectory in a compact time interval $[0, T]$ and η is not Zeno [1], the sequence of transition times is finite, and the numerical trajectory $\eta(x_0, w)$ can be found using numerical solvers. We assume our system does not exhibit Zeno behaviors.

2.2.2 Temporal logic requirements and robustness

Signal Temporal Logic is a logical formalism that allows efficient and unambiguous specifications of a wide variety of desired system properties beyond basic properties like stability and reachability. An STL formula is a composition of temporal and Boolean operations over predicates over signals. A predicate p_k represents a set of states defined using a real-valued function $\mu : X \to \mathbb{R}$ as $P_k = \{x \in X \,|\, \mu(x) > 0\}$. Let $P = \{p_1, ...p_L\}$ be the set of all predicate expressions of interest for an STL formula and $\mathcal{I} \subset \mathbb{R}_+$ be any non-empty interval. The set of all well-formed STL formulas φ is inductively defined as

$$\varphi \;:=\; \top \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \mathcal{U}_{\mathcal{I}} \psi,$$

where $p \in P$, and $\top$, $\neg$ and $\wedge$ are Boolean *true*, *negation* and *and* operations, and $\mathcal{U}$ is the *until* temporal operator, which requires ψ to be satisfied at some time in $\mathcal{I}$ and until then,

[1] η is *Zeno* if it does an infinite number of jumps in a finite amount of time. A hybrid system is Zeno if at least one of its trajectories is Zeno.

φ needs to be satisfied. The validity of a formula φ with respect to a signal $\mathbf{x}$ at time t is defined inductively as follows:

$$(\mathbf{x}, t) \models p_k \Leftrightarrow \mu_k(x_t) > 0$$

$$(\mathbf{x}, t) \models \neg\varphi \Leftrightarrow \neg((\mathbf{x}, t) \models \varphi)$$

$$(\mathbf{x}, t) \models \varphi \wedge \psi \Leftrightarrow (\mathbf{x}, t) \models \varphi \wedge (\mathbf{x}, t) \models \psi$$

$$(\mathbf{x}, t) \models \varphi \mathcal{U}_{\mathcal{I}} \psi \Leftrightarrow \exists t' \in (t + \mathcal{I}) \; s.t. \; (\mathbf{x}, t') \models \psi \wedge \forall t'' \in [t, t'), (\mathbf{x}, t'') \models \varphi$$

where x_t is the value of signal $\mathbf{x}$ at time t and $(t + \mathcal{I}) = \{t + c \,|\, c \in \mathcal{I}\}$. Other operators like Disjunction ($\vee$), Always ($\Box$) and Eventually ($\Diamond$) can be defined using the above operators (see [21]). The trajectory $\mathbf{x}$ satisfies φ if $(\mathbf{x}, 0) \models \varphi$. An STL formula is bounded-time if all its temporal intervals are bounded.

Definition 1. *The maximum trajectory length N required to decide satisfiability of an STL formula is called the formula horizon. The formula horizon is the maximum over the sums of all the nested upper bounds on the temporal operators.*

For instance the horizon of the formula $\Box_{[0,t_1]}(\Diamond_{[0,t_2]} p)$ is equal to $t_1 + t_2$. In this paper, we consider bounded time STL formulas.

The degree of satisfaction of an STL formula [38] which we call the robustness value, can be calculated using a real-valued function ρ_φ of a trajectory $\mathbf{x}$, and t such that $(\mathbf{x}, t) \models \varphi \equiv \rho_\varphi(\mathbf{x}, t) > 0$. The semantics of ρ_φ can be defined as follows:

$$\rho_{p_k}(\mathbf{x}, t) = \mu_k(x_t)$$

$$\rho_{\neg\varphi}(\mathbf{x}, t) = -\rho_\varphi(\mathbf{x}, t)$$

$$\rho_{\varphi\wedge\psi}(\mathbf{x}, t) = \min(\rho_\varphi(\mathbf{x}, t), \rho_\psi(\mathbf{x}, t))$$

$$\rho_{\varphi \mathcal{U}_{\mathcal{I}} \psi}(\mathbf{x}, t) = \max_{t' \in (t+\mathcal{I})} \left(\min \left(\rho_\phi(\mathbf{x}, t'), \min_{t'' \in [t, t')} \rho_\varphi(\mathbf{x}, t'') \right) \right)$$

Temporal operators like Always and Eventually, can be treated as conjunctions and disjunctions along the time axis. I would like to highlight that while I choose to work with STL as it is easier to analyze and differentiate, STL interpretation of robustness can differ

from the MTL robustness as defined in [50] for general functions of the form $\mu_k(x) > 0$. Using the interpretation in [50], the robustness w.r.t a predicate p_k is the signed distance (see [50] for more information) of x from the predicate set $P_k = \{x \in X | \mu_k(x) > 0\}$. However in [38], the robustness w.r.t this predicate is simply defined as $\mu_k(x)$ which fails to define a robustness tube around the signal within which all the trajectories satisfy the property. Nevertheless, for both semantics, positive sign of the robustness implies Boolean satisfaction and negative sign implies falsification.

In this section, the partial derivative of a function $f(x_1, x_2, ..., x_n)$ with respect to its ith argument is denoted with $\partial_i f$; if the function includes only one argument we show its derivative with respect to that argument with ∂f.

2.3 Problem Formulation

In this section, we focus on solving the falsification problem by the automated generation of optimal test cases. In other words, the purpose is to find a system's violating behavior with respect to some TL formula, as fast as possible. Consider the following problem:

Problem 2.3.1 (Falsification Problem)**.** *Assume that the dynamical system Σ as defined in Eq. (2.1) or the hybrid system $\mathcal{H}$ as in Eq. (2.2) needs to comply with the STL specification φ whose formula horizon is T for all the initial conditions $x_0 \in X_0$, and exogenous inputs w ($w(t) \in \mathbb{R}^{x_w}$) which are admissible with respect to some input constraints. Find an initial condition $x_0 \in X_0$ and an admissible input w such that the system trajectory $\mathbf{x}(x_0, w)$ does not satisfy φ, i.e, $\rho_\varphi(\mathbf{x}(x_0, w)) < 0$.*

Recall that the positive robustness values indicate satisfaction, while negative robustness values show a violation of the specification, and the absolute value of the robustness function indicates how robustly the specification is satisfied or violated. As a result, falsification of an STL property φ can be achieved through minimization of the robustness function ρ_φ and the Falsification Problem 2.3.1 can be altered as follows:

Problem 2.3.2 (Minimization of the Robustness Function)**.** *Find the optimal initial condition $x_0^* \in X_0$ and the optimal admissible input w^* such that the corresponding state trajectory $\mathbf{x}(x_0^*, w^*)$ of the dynamical system Σ as defined in Eq. (2.1) or the hybrid system $\mathcal{H}$ as in Eq. (2.2): minimizes the robustness function ρ_φ, i.e, $\rho_\varphi(\mathbf{x}(x_0^*, w^*)) \leq \rho_\varphi(\mathbf{x}(x_0, w))$ for all $x_0 \in X_0$ and admissible inputs w.*

Solving Problem 2.3.2 is extremely challenging due to the inherent non-differentiability and non-convexity of the cost function ρ_φ and its convoluted dependency on x_0 and w through the system dynamics.

In order to deal with the non-differentiability of the cost function, in [3], a differentiable function is introduced that can approximate the robustness function locally. The approximation is based on the fact that the robustness value for the trajectory $\mathbf{x}$ corresponds to the value of the trajectory at some critical time t_*, and a critical predicate $p_* \in P$, which corresponds to the set $P_* = \{x \mid \mu_*(x) > 0\}$[2]. In this case, the robustness value is the value that the function μ_* takes at $x(t_*)$, i.e, $\rho_\varphi(\mathbf{x}) = \mu_*(x(t_*))$. The variables p_* and t_* can be calculated using tools such as S-TaLiRo while evaluating the robustness.

Assume that $t_*^{\mathbf{i}}, p_*^{\mathbf{i}}$ are the critical time and predicate corresponding to the initial condition $x_0^{\mathbf{i}}$, and the input $w^{\mathbf{i}}$. The robustness of the trajectories $\mathbf{x}(x_0, w)$ with $x_0 \in \mathcal{N}(x_0^{\mathbf{i}}, \epsilon)$, and $w \in \mathcal{N}(w^{\mathbf{i}}, \epsilon)$ is approximated with the following cost function:

$$J^{\mathbf{i}}(x_0, w) = \mu_*^{\mathbf{i}}(\mathbf{x}(x_0, w, t_*^{\mathbf{i}})) \tag{2.6}$$

where $\mu_*^{\mathbf{i}}$ is the function corresponding to $p_*^{\mathbf{i}}$. The index $\mathbf{i}$ shows that this cost function approximates the robustness function locally in a neighboring of $x_0^{\mathbf{i}}, w^{\mathbf{i}}$. Note that the use of the superscript $\mathbf{i}$ will be motivated soon. Hence Problem 2.3.2 is changed to the following problem in [3]:

Problem 2.3.3 (Sequential Minimization of $J^{\mathbf{i}}$)**.** *Find descent directions $\delta x_0^{\mathbf{i}}$, and $\delta w^{\mathbf{i}}$ and*

[2]To avoid technicalities, we assume that in the case of the hybrid systems the critical predicate lies in a single location

the step size h such that the sequence $\{(x_0^{\mathbf{i}}, w^{\mathbf{i}})\}, \mathbf{i} = 1, ..., M$, where $x_0^{\mathbf{i}+1} = x_0^{\mathbf{i}} + h\delta x_0^{\mathbf{i}}$, $w^{\mathbf{i}+1} = w^{\mathbf{i}} + h\delta w^{\mathbf{i}}$ satisfies the following conditions:

1. *Initial conditions $x^{\mathbf{i}}$ are in X_0 and inputs $w^{\mathbf{i}}$ are admissible,*
2. *$J^{\mathbf{i}}(x_0^{\mathbf{i}+1}, w^{\mathbf{i}+1}) < J^{\mathbf{i}}(x_0^{\mathbf{i}}, w^{\mathbf{i}})$,*
3. *$0 < h < \bar{h}$ and for all $0 < h' < \bar{h}$, $J^{\mathbf{i}}(x_0^{\mathbf{i}} + h'\delta x_0^{\mathbf{i}}, w^{\mathbf{i}} + h'\delta w^{\mathbf{i}}) < J^{\mathbf{i}}(x_0^{\mathbf{i}}, w^{\mathbf{i}})$*
4. *$J^M(x_0^M, w^M) \leq 0$*

2.4 Solution Approach and Experimental Results

Assuming that the input $w \in W$ is piece-wise constant (or linear) with given fixed switching times (between consecutive pieces) τ^k, and parameters $\sigma^k, k = 1, ..., d$ such that $w(t) = \sigma^k, t \in [\tau^{k-1}, \tau^k)$ the problem is solved in [3] for the dynamical system (2.1), by computing the sensitivity matrices s_{x_0} and s_σ^k as follows:

$$\begin{aligned} \dot{s}_{x_0}(t) &= \partial_1 f(x(t), w(t)).s_{x_0}(t), \quad s_{x_0}(0) = I_{n_x} \\ \dot{s}_\sigma^k(t) &= \partial_1 f(x(t), w(t)).s_\sigma^k(t) + \delta_k.\partial_2 f(x(t), w(t)), \quad t \in [\tau^{k-1}, T], \\ s_\sigma^k(t) &= 0, \quad t \in [0, \tau^{k-1}] \end{aligned} \tag{2.7}$$

where δ_k is equal to one if $t \in [\tau^{k-1}, \tau^k]$ and zero, otherwise. Then the directions $\delta x_0^{\mathbf{i}}, \delta\sigma^{k,\mathbf{i}}$ can be computed as follows:

$$\begin{aligned} \delta x_0^{\mathbf{i}} &= -\partial\mu_*^{\mathbf{i}}\big|_{\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}}, t_*^{\mathbf{i}})}^\top s_{x_0}(t_*^{\mathbf{i}}) \\ \delta\sigma^{k,\mathbf{i}} &= -\partial\mu_*^{\mathbf{i}}\big|_{\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}}, t_*^{\mathbf{i}})}^\top s_\sigma^k(t_*^{\mathbf{i}}) \end{aligned} \tag{2.8}$$

where sensitivity matrices are computed for the trajectory $\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}})$. The step size h is chosen such that $x_0^{\mathbf{i}} + h\delta x_0^{\mathbf{i}} \in X_0$ and $w^{\mathbf{i}} + h\delta w^{\mathbf{i}} \in W$, where $\delta w^{\mathbf{i}} = \delta\sigma^{k,\mathbf{i}}$ for $t \in [\tau^{k-1}, \tau^k]$. In this thesis, we consider the Problem 2.3.3 under the following assumptions:

- Section 2.4.1 presents the results of [134] which studies the problem under the assumption that *the system is nonlinear but we only have partial knowledge about it and*

available uncertainties in the system is in terms of initial conditions and parameters of piece-wise constant exogenous inputs.

- Section 2.4.2 presents the results of [136] which studies the problem of the falsification of a nonlinear system and extends the results for the class of exogenous input signals from piece-wise constant with fixed switch times to *piece-wise constant signals with varying switching times.*

- Section 2.4.3 presents the results of [137] which studies the problem under the assumption that the system is nonlinear and extends the results to *non-parametrized exogenous input signals and directly searches the infinite-dimensional search space of the input functions.*

- Section 2.4.4 presents the results of [135] which studies the falsification problem of *white-box hybrid automata* that models CPS, assuming that exogenous inputs are piece-wise constant. The results of this paper can be extended to the case of piece-wise constant signals with varying switching times.

2.4.1 Gradient-based Gray-box Falsification of Nonlinear Systems

In order to remove the need for the full knowledge of f to compute Eq. (2.7), linearizations of the system -that can be provided using MBD tools like Simulink- are computed in sample points $\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}}, t_s)$ along the system trajectory $\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}})$. The linearization in each sample point is used instead of $\partial_1 f(x(t), w(t))$ and $\partial_2 f(x(t), w(t))$ in Eq. (2.7) until new information about the linearized system is achieved in the next sample point on the trajectory. It has been shown in [134] that as the number of samples in which linearized system is computed increases, the directions computed using this method get arbitrarily close to directions in Eq. (2.8). The reason is that the error in computing sensitivity matrices (2.7) can get arbitrarily small with an increased number of samples.

Also, since descent directions with a proper step size h only guarantee convergence to local minimums, in order to explore the search space, Alg. 1 that combines the local

Algorithm 1 Simulated annealing combined by gradient descent algorithm

Data: The system Σ in Eq. (2.1) or its Simulink model, set of initial conditions X_0, input range W, system specification φ, total number of iterations N, step size h, maximum number of iterations k_1 and k_2.

Set $i = 1$, and randomly select (x_0, w) considering X_0, W.

$r^* \leftarrow$ simulate Σ from (x_0, w), and find robustness value with respect to φ.

while $i <= N$ **do**

  for j = *1,...,*k_1 **do**

    $(x_0', w') \leftarrow$ GET_NEW_SAMPLE_BY_SA(x_0, u, X_0, W).

    $r \leftarrow$ SIMULATE_FIND_ROBUSTVAL$(x_0', w', \Sigma, \varphi)$.

    $Bool \leftarrow$ ACCEPT?(r, r^*, i), $i \leftarrow i + 1$

    if $Bool = 1$ **then**

      $(x_0, w) \leftarrow (x_0', w')$

      if $r < r^*$ **then**

        $(x_0^*, w^*, r^*) \leftarrow (x_0', w', r)$

      end

    end

  end

  $(\delta x_0, \delta w) \leftarrow$ COMPUTE_DESCENT(x_0, w, Σ), $h' \leftarrow h$

  for j = *1,...,*k_2 **do**

    $(x_0', w') \leftarrow$ APPLY_DESCENT$(\delta x_0, \delta w, x_0, w, h', X_0, W)$

    $r \leftarrow$ SIMULATE_FIND_ROBUSTVAL$(x_0', w', \Sigma, \varphi)$

    if $r < r*$ **then**

      $(x_0^*, w^*, r^*) \leftarrow (x_0', w', r)$

    else

      $h' \leftarrow h'/2$

    end

  end

end

return optimal initial condition x_0^*, optimal input w^* and the related optimal robustness value r^*.

search using descent directions with Simulated Annealing (SA) optimization algorithm is presented. The algorithm is compared with the pure SA algorithm on multiple benchmark examples to show that using local search can help to find unsafe behaviors faster.

2.4.2 Falsification Using Gradient-based Local Search in Space and Time

As piece-wise constant input signals with fixed switching times may not be a general enough parametrization method, in [136], we proposed piece-wise constant input signals with varying switching times. We indicate a piece-wise constant input w consisting of d pieces with a sequence $(\sigma^1, \tau^1, \sigma^2, \tau^2, ..., \tau^{d-1}, \sigma^d)$ where $0 < \tau^1 < ... < \tau^{d-1} < \tau^d = T$ and $\sigma^1, ..., \sigma^d \in W$. This input is described below and it is shown in Fig. 2.1.

$$w(t) = \sigma^k, \text{ for } t \in [\tau^{k-1}, \tau^k], k = 1, ..., d \tag{2.9}$$

In this case, the desired change in the system input $w^{\mathbf{i}}$ ($\delta w^{\mathbf{i}}$) is computed by computing $\delta\sigma^{k,\mathbf{i}}$ and $\delta\tau^{k,\mathbf{i}}$. For presentation purposes, we'll drop the superscript $\mathbf{i}$ in the sensitivity analysis, as the analysis can be performed for all the iterations similarly. In order to compute $\delta\tau^k$ the sensitivity $s_\tau^k(t)$ to switching times τ^k need to be computed. Assuming $\delta\tau^k > 0, \forall k = 1, ..., d-1$, for any $t < \tau_k$, $s_\tau^k(t) = 0$. Since $\delta\tau^k$ is assumed to be small in value, for any $t \in [\tau_k, \tau_k + \delta\tau^k]$, $\delta s_\tau^k(t) = f(x, \sigma^k) - f(x, \sigma^{k+1})\delta\tau_k$. So, as pictured in Fig 2.2, the sensitivity to switch times can be computed as:

$$\begin{aligned} s_\tau^k(t) &= 0, \quad t \in [0, \tau^k] \\ s_\tau^k(t) &= f(x, \sigma^k) - f(x, \sigma^{k+1}), \quad t = \tau^k \\ \dot{s}_\tau^k(t) &= \partial_1 f(x(t), w(t)).s_\tau^k(t), \quad t \in [0, \tau^k] \end{aligned} \tag{2.10}$$

Then $\delta\tau^{k,\mathbf{i}}$ can be computed as follows:

$$\delta\tau^{k,\mathbf{i}} = -\partial\mu_*^{\mathbf{i}}\big|_{\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}}, t_*^{\mathbf{i}})}^{\top} s_\tau^{k,\mathbf{i}}(t_*^{\mathbf{i}}) \tag{2.11}$$

where sensitivity matrices $s_\tau^{k,\mathbf{i}}$ are computed for the trajectory $\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}})$. Computing $\delta x_0^{\mathbf{i}}, \delta\sigma^{k,\mathbf{i}}$ using the Eq. (2.8), the step size h is chosen such that $x_0^{\mathbf{i}} + h\delta x_0^{\mathbf{i}} \in X_0$ and

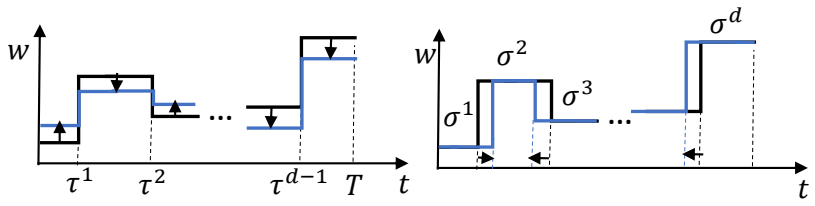


Figure 2.1: Input Perturbation: The Left Plot Shows a Perturbation in Space ($\delta\sigma^{\mathbf{i}}$) and the Right Plot Shows One in Time ($\delta\tau^{\mathbf{i}}$).

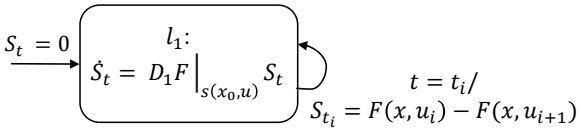


Figure 2.2: HA for Calculating the Sensitivity to the Input Switch Times

$w^{\mathbf{i}+1} \in W$, where $w^{\mathbf{i}+1}$ is computed by applying the perturbations $\delta\sigma^{k,\mathbf{i}}$ and $\delta\tau^{k,\mathbf{i}}$ to the parameters $\sigma^{k,\mathbf{i}}$ and $\tau^{k,\mathbf{i}}$ corresponding to $w^{\mathbf{i}}$.

Remark 1. *This analysis can be performed similarly when the system has more than one exogenous input signal.*

This input parametrization is appealing as it allows bounding the value of the exogenous inputs to the set $W = [\underline{W}, \overline{W}]$, and at the same time, it allows revealing system behaviors that are related to frequency properties of w. The step size h can be further tuned to find the largest step in the descent direction that reduces the cost $J^{\mathbf{i}}$ (or the robustness value). Furthermore, Alg. 1 can be used to combine a global search method like SA with the local search using gradient descent.

Two case studies are used in [136] to evaluate the performance of the space-time gradient descent in testing system requirements.

Powertrain system: This benchmark is introduced as the third model in [76]. Some preliminary results on the original model are presented in [33]. These results reveal some of the challenges of this model. The model we study in this paper is the stiff polynomial approximation of the original model. It is a closed-loop model of an engine under an

air/fuel controller. The closed-loop system consists of 5 states and takes two exogenous inputs: the throttle angle θ_{in} and, the engine speed ω. We test the system in the "Normal" operation mode w.r.t the following requirement: "The air/fuel (A/F) ratio remain in the invariant set $[14.56, 14.84]$ from $t = 3$ to the end of the simulation time T=50." During the test, the engine speed is supposed to remain constant but it can attain different values in $[900\pi/30, 1100\pi/30]$. The throttle input however is a time-varying input as a result of possibly different behaviors by the driver, but it is assumed to be bounded in the set $[0, 81.2]$. We used 1 parameter to describe ω and 21 parameters (11 for signal values and 10 for switch times) to describe θ_{in}. Starting from a trajectory with the robustness of 0.129, we find a falsifying trajectory with robustness -0.003 using GD in 4 iterations. The initial and final trajectories and the inputs are shown in Fig. 2.3.

Note that some properties for this model have been verified/falsified in [52], but they only test the system under time invariant uncertainties and assume that the driver behaviors (that affect θ_{in}) are limited to two specific behaviors. However, here, we test the system over different driver behaviors.

Maneuvering object : This benchmark is a hybrid model also used in [135]. The maneuvering object has a pair of off-centered thrusters as the control input. The location-based (hybrid) dynamics of the vehicle is as follows:

$$\begin{bmatrix} \dot{x}_i \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_{i+3} \\ 0.1x_4 + \Sigma_{i=1,2} s_i(l)(x_1 - \alpha_i) + F_1 cos(x_5) - F_2 sin(x_5) \\ 0.1x_5 + \Sigma_{i=1,2} s_i(l)(x_2 - \beta_i) + F_1 sin(x_5) - F_2 cos(x_5) \\ -bF_1/I + aF_2/I \end{bmatrix} \tag{2.12}$$

where $i = 1, 2, 3$ and (x_1, x_2) is the positions along the x and y axis. The hybrid model consists of 3 locations: the first to third locations are specified using the sets $\{x \,|\, x_1 < 4\}$, $\{x \,|\, 4 \leq x_1 \leq 8\}$, and $\{x \,|\, x_1 > 8\}$, respectively. At the first, second and third locations, we have $s_1 = s_2 = 0$,$s_1 = -1, s_2 = 0$, and $s_1 = 0, s_2 = -2$, respectively. Also, (α_1, β_1) and (α_2, β_2) are the centers of the unsafe sets $\mathcal{U}_1$ and $\mathcal{U}_2$, respectively. We consider $a = b = I = 1$.

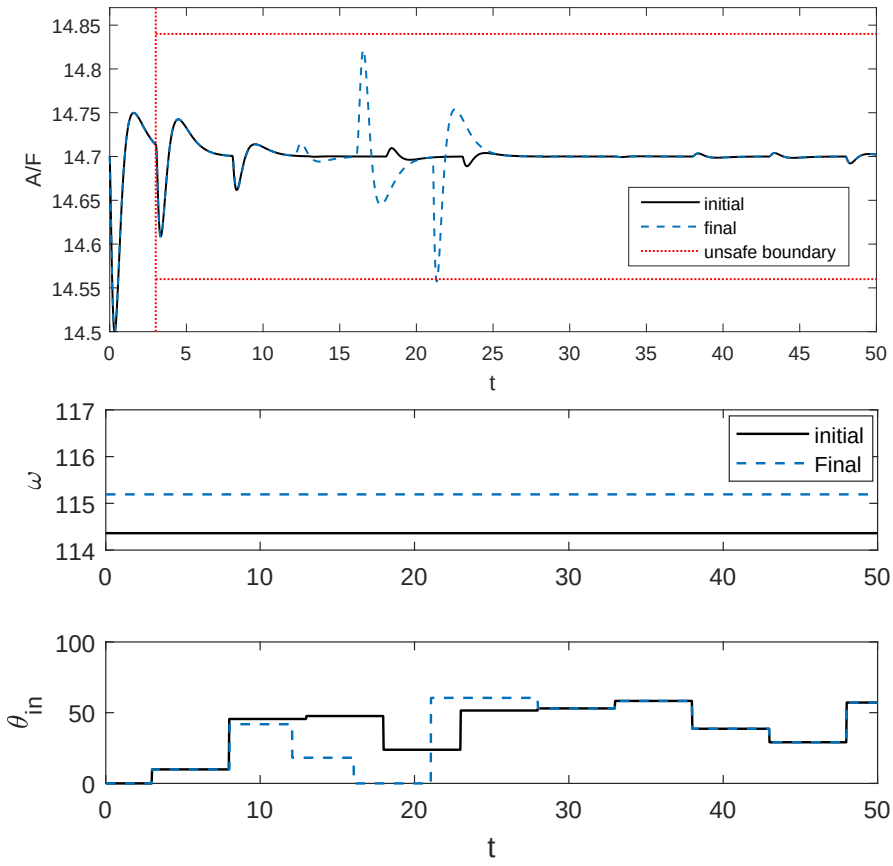


Figure 2.3: Top: GD Increases the Air/Flow Ratio over and Undershoots Causing It to Find a Falsifying Trajectory. Bottom Two: Initial and Falsifying Engine Speed and Throttle Input.

The system is required to satisfy this requirement: "Always avoid $\mathcal{U}_1 = [5.5, 6.5] \times [2.5, 3.5]$ and $\mathcal{U}_2 = [9.5, 10.5] \times [1.5, 3.5]$, and eventually reach the goal set $A = [11, 13] \times [3, 5]$ within T=10 seconds." The search is over the time and the amplitude of the piecewise constant inputs $F_1 \in [0, 0.3]^{[0,T]}, F_2 \in [-.2.2]^{[0,T]}$ and also initial conditions for $x_1 \in [0, 1]$ and $x_2 \in [0.4, 0.8]$. We parametrize F_1 and F_2 using 19 parameters each (10 parameters for signal values and 9 for switch times), so the search is over a 40 dimensional space. Using gradient descent on the negation of the requirement formula, we find inputs F_1 and F_2 that satisfy the requirement. The initial trajectory has robustness 3.3483 w.r.t the negation of the requirement (so it does not satisfy the requirement) and gradient descent decreases the

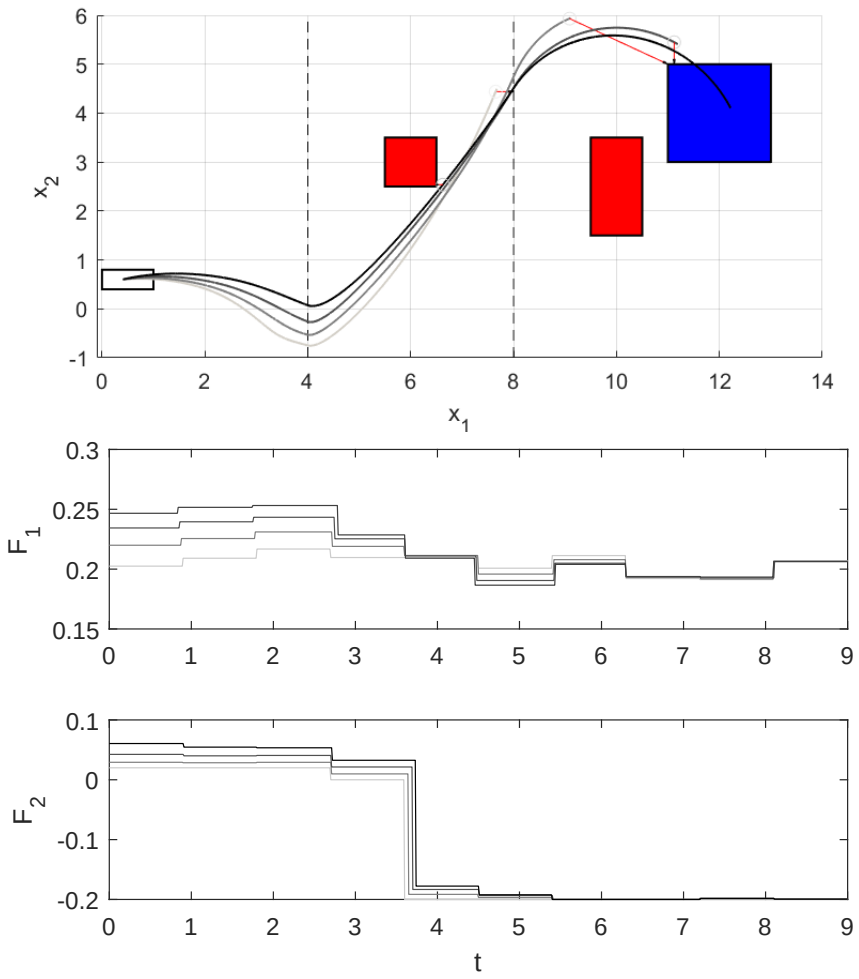


Figure 2.4: Top: Maneuvering Object Trajectories: GD Gradually Improves the Trajectories and Finally Finds a Satisfying Trajectory. Bottom Two: Maneuvering Object Control Inputs, the Shifts along the Time Axis Are the Result of Time Descent and the Shifts along the F-axis Are the Results of Space Descent.

robustness to -0.12462 in 4 iterations and finds a trajectory that satisfies the requirement. The system trajectories and their corresponding inputs in these 4 iterations are shown in Fig. 2.4. We show the signals with a darker marker as the iteration number increases.

Experimental results: To globally search for the minimizer of the robustness function, we use Alg. 1. We performed two statistical studies on the "Maneuvering object" and the "Powertrain" benchmarks to determine the effectiveness of applying GD. In both

Table 2.1: Experimental Results for the Space-Time Parametrization

	Maneuvering object		Power-train	
Optim. method	SA	SA+GD	SA	SA+GD
Num. of falsification	6/40	24/40	0/50	11/50
Avg. min robustness value	4.4435	3.2116	0.0725	0.0573
max min robustness value	10.1442	10.1301	0.1376	0.1382
min min robustness value	-0.5313	-0.6104	0.0162	-0.0436

experiments, we ran SA and SA+GD 50 times starting from the same initial conditions and with an equal total number of samples N=100. The results are presented in Table 2.1.

In both cases, the falsification rate has improved significantly with the help of GD. In the case of the "Maneuvering object" benchmark it achieves one order of magnitude improvement. In the case of the Powertrain benchmark, while SA+GD falsifies the requirement in 11 out of 50 runs, SA has not been able to find any falsifying input.

2.4.3 Falsification by Direct Search in the Function Spaces

While depending on the context, input parametrization may be useful and required to represent exogenous inputs of a given class, if the purpose is to find any exogenous inputs - constrained only by their magnitudes - that violate the system requirement, we need to search the infinite dimensional function spaces directly. In order to find $\delta w : t \to W$, that satisfies the condition in Problem 2.3.3, for a general (non-parametrized) exogenous input w, in [137], we used a method based on optimal control. Using the co-state method, we alter $J^{\mathbf{i}}$ as follows:

$$\bar{J}^{\mathbf{i}} = J^{\mathbf{i}} + \int_0^{t_*^{\mathbf{i}}} \lambda^\top \big(f(x,w) - \frac{dx}{dt}\big) dt$$

Forming the Hamiltonian as $H(x,w) = \lambda^\top f(x,w)$, $\bar{J}^{\mathbf{i}}$ can be written as:

$$\bar{J}^{\mathbf{i}} = \mu_*^{\mathbf{i}}(x(t_*^{\mathbf{i}})) + \lambda(0)^\top x(0) - \lambda(t_*^{\mathbf{i}})^\top x(t_*^{\mathbf{i}}) + \int_0^{t_*^{\mathbf{i}}} \big(H(x,w) + \frac{d\lambda}{dt}^\top x\big) dt$$

As a result, the gradient of the cost function $\bar{J}^{\mathbf{i}}$ is:

$$\delta\bar{J}^{\mathbf{i}} = \Big(\frac{d\mu_*^{\mathbf{i}}\big(x^{\mathbf{i}}(t_*^{\mathbf{i}})\big)}{dx} - \lambda^\top(t_*^{\mathbf{i}})\Big)\delta x(t_*^{\mathbf{i}}) + \lambda^\top(0)\delta x(0) + \int_0^{t_*^{\mathbf{i}}} \Big(\big(\frac{\partial H}{\partial x} + \dot{\lambda}^\top\big)\delta x + \frac{\partial H}{\partial w}\delta w\Big)dt$$

By updating the co-states λ backward in time with the following final value ODE,

$$\begin{aligned} \dot{\lambda}(t) &= -\frac{\partial f}{\partial x}\Big|^\top_{\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}}, t)}\lambda(t) \\ \lambda(t_*^{\mathbf{i}}) &= \frac{dJ^{\mathbf{i}}}{d\mathbf{x}}^\top = \partial\mu_*^{\mathbf{i}}\big|^\top_{\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}}, t_*^{\mathbf{i}})} \end{aligned} \tag{2.13}$$

$\delta\bar{J}^{\mathbf{i}}$ is reduced to $\delta\bar{J}^{\mathbf{i}} = \lambda^\top(0)\delta x(0) + \int_0^{t_*^{\mathbf{i}}} \frac{\partial H}{\partial w}\delta w\, dt$. Hence, the following choices of $\delta x_0^{\mathbf{i}}$ and $\delta w^{\mathbf{i}}$ with a small enough positive step size h will result in a negative $\delta\bar{J}^{\mathbf{i}}$ and as a result a decrease in $\bar{J}^{\mathbf{i}}$:

$$\begin{aligned} \delta x_0^{\mathbf{i}} &= -\lambda(0) \\ \delta w^{\mathbf{i}}(t) &= -\frac{\partial f}{\partial w}\Big|^\top_{\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}}, t)}\lambda(t) \end{aligned} \tag{2.14}$$

where λ is computed for the trajectory $\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}})$.

It has also been shown that in order to relax the need for a white-box model, the system's linearization information along the trajectory can be used for computing the co-states in Eq. 2.13 approximately, as it was used in [134] for approximating the sensitivity matrices. Algorithm 2 describes the process of finding inputs and initial conditions with reduced robustness value using the descent directions computed above. We can stop the algorithm based on different criteria. The algorithm can be stopped if:

- A maximum number of iterations is reached.
- The change in the robustness is less than a minimum value.
- The changes in the initial conditions and inputs are less than a minimum value.

To combine the local search using Alg. 2 with a "sampling method for coverage" or a "stochastic global optimization" approach, one can use the approach pictured in Fig. 2.5,

[2] $\delta x_p^i(0)$ is the non NN part of $\delta x^i(0)$

Algorithm 2 Optimal input and initial condition for falsification

Data: The system Σ in Eq. (2.1) or its Simulink model, TL formula φ, x_0^1, $w^1(t)$, X_0, W, step size h_0, and constant $c > 1$.

Initialize $r_* = \infty$, $(x_0', w') = (x_0^1, w^1)$

$(r_*, t_*) \leftarrow$ SIMULATE_FIND_ROBUSTVAL$(x_0, w, \Sigma, \varphi)$.

while *the stop condition is not active* **do**

$i \leftarrow 1$, $h \leftarrow h_0$

$\diamond$ **if** $r' < r_*$, **then**

$(r_*, t_*) \leftarrow r'$, $(x_0, w) \leftarrow (x_0', w')$

else

$i \leftarrow i + 1$

$h = h/2$

$(x_0', w') \leftarrow$FIT_IN_BOX$(x_0, w, \delta x_0, \delta w(t), X_0, W, h)$

$(r', t_*) \leftarrow$ SIMULATE_FIND_ROBUSTVAL$(x_0', w', \Sigma, \varphi)$

if $i < K$ **then**

return to $\diamond$

else

break

end

end

if $r < 0$ **then**

Break

end

Linearize the system around sample times taken in $[0, t_*]$ and evaluate δx_0 and $\delta w(t)$ using equations (2.14)

end

return local optimal initial condition x, local optimal input w, and r_*.

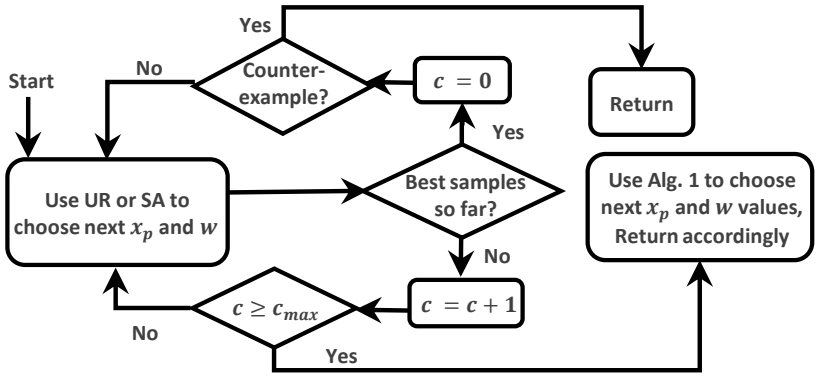


Figure 2.5: The Falsification Framework. UR Stands for Uniform Sampling.

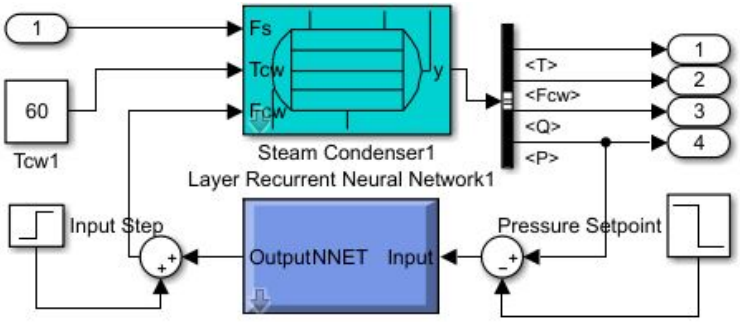


Figure 2.6: Simulink Model for Steam Condenser with Feedback RNN Controller

where $c = 0$ in the beginning and c_{max} is a design choice. Note that since the exogenous input will not conform with parametrization methods after it is perturbed along $\delta w(t)$, the local and global search cannot be intertwined as in previous approaches.

The method has been used for falsifying properties of Simulink models of a nonlinear system that includes feed-forward NN in the loop, in addition to a model of a Steam Condenser under a Recurrent NN controller which we will present next.

Steam Condenser with RNN Controller: We considered a dynamic model of a steam condenser with 5 continuous states based on energy balance and cooling water mass balance under an RNN controller with 6 discrete states and tangent-sigmoid activation functions. The Simulink model for the system is shown in Fig. 2.6. The steam flow rate $w(t)$ (Input 1 in Fig. 2.6) is allowed to vary in the set $[3.99, 4.01]$ and the system is tested for $T = 35$ seconds against the specification $\square_{[30,35]}\, p(t) \in [87, 87.5]$. Starting from

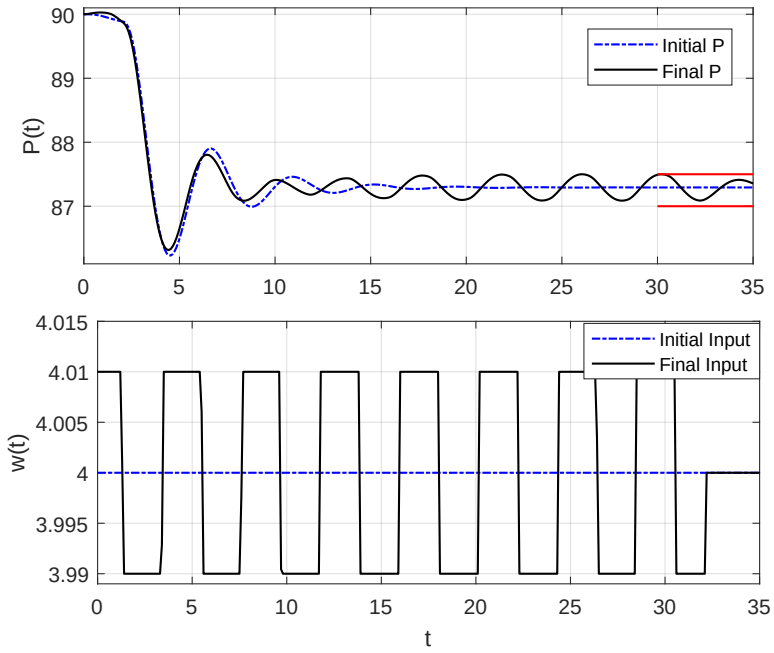


Figure 2.7: The System Robustness Is Reduced From .20633 Using a Constant Input $w(T) = 4$ To .00030222 Using the Local Optimal Input Shown in the Bottom Picture.

a constant valued signal $w(t) = 4$ that results in a robustness value equal to 0.20633, the above approach finds a falsifying trajectory with robustness 0.00030222. The initial and final trajectories and inputs are shown in Fig. (2.7). While the local search reduces the robustness values significantly, no falsifying behavior is found. The importance of combining this local search with a global sampler/optimizer becomes more clear in the following where the combination of the local search with uniform random sampling or Simulated Annealing method finds adversarial examples.

Note that, while the utilized NNs have a fairly small number of layers (since they were found to perform well enough during the training phase), the scalability of the proposed approach was tested on the systems of Sec. 5.1 and 5.2 including NN controllers with a larger number of layers (20 to 100) too. These experiments showed that the proposed approach scales well. Theoretically increasing the number of layers/neurons in FNNs or the number of non-recurrent layers (with no delay/memory) in RNNs will just increase the

Table 2.2: Falsification Results of Steam Condenser System with RNN Controller Using Different Search Methods.

	UR	SA	UR+GD	SA+GD
# falsifications	0/50	0/50	50/50	50/50
Avg min robustness	0.0843	0.0503	-0.0018	-0.0016
Avg execution time	> 60	> 60	15.7812	13.0688
Avg # simulations	600	600	87.48	62.26

number of blocks in the Simulink model linearly. Since MATLAB analytical linearization is computed block-by-block, increasing the number of these kinds of layers (l) increases the linearization complexity by $O(l \times r)$ where r is the maximum number of neurons in layers. However, increasing the size of state-space or the number of layers of the RNN with memory increases the linearization complexity faster. Specifically, the size of linearized matrices grows quadratically with the number of state-space plus RNN states. However, in practice, we observed a much less increase in the computation time of the overall algorithm when increasing the size of the NN states.

Experimental Results: We used Uniform the Random Sampling (UR) and Simulated Annealing (SA) implementations of S-TaLiRo unaided and aided by the optimal local search (UR+GD and SA+GD, respectively). For sampling using SA and UR, inputs were (initially[3]) considered to be piece-wise constant signals with 12 control points with varying sample times (total of 24 variables).

In the UR+GD implementation, local optimal search is performed when the sampler cannot find a sample with a less robustness value 50 times in a row, and in the SA+GD implementation, it is applied when the optimizer cannot find a less robust sample 30 times in a row. We run the experiments 50 times, and in each run, the maximum execution time is limited to 60 seconds[4]. The search is initialized with the same seed for all the

[3]when aided by the optimal local search, arbitrary perturbations were applied to inputs

[4]The next sample is taken only if the execution time so far is less than 60 seconds. The algorithm returns

experiments. The above search methods are compared against the number of falsifications found, average minimum robustness found, average execution time, and the average total number of simulations before returning. The improvement in the results from left to right in Table 2.2 is evident and it motivates the use of the proposed local search. While SA and UR were not able to find any counterexamples in 50 runs, their combination with gradient-based descent found an adversarial example in all the runs within a short amount of time and with less than 90 simulations on average. In fact, in a study that has been conducted in [47], multiple other falsification methods failed to find a falsifying behavior.

2.4.4 Falsification of Hybrid Systems

Sensitivity matrices and co-states can be computed using Eq. (2.7) and (2.13) only if we are dealing with a nonlinear system of the form (2.1) in which f is smooth and differentiable. So, to be able to solve Problem 2.3.3 for hybrid systems as defined in Eq. (2.2), the discussion has been extended in [135]. In order to solve Problem 2.3.3 by computation of sensitivity matrices and descent directions, we should impose further assumptions on our system stated below:

1. The system of Eq. (2.2) is observable, i.e., we have access to all the system states, or we have a state estimator which can estimate them.

2. In the local search stage, we are always able to find a neighboring tube around each trajectory such that none of the trajectories inside that tube hit the guard tangentially. This ensures that trajectories of the system $\mathcal{H}$ starting close enough to x_0 and under neighboring inputs of u undergo similar transitions/switches. This is guaranteed if we can find an auto-bisimulation function of a trajectory and the trajectories starting from its neighboring initial conditions and under neighboring inputs [130].

3. The system is deterministic and the transitions are taken as soon as possible. To have a deterministic system, if two transitions happen from the same location, their

faster in case of finding a counter/adversarial example.

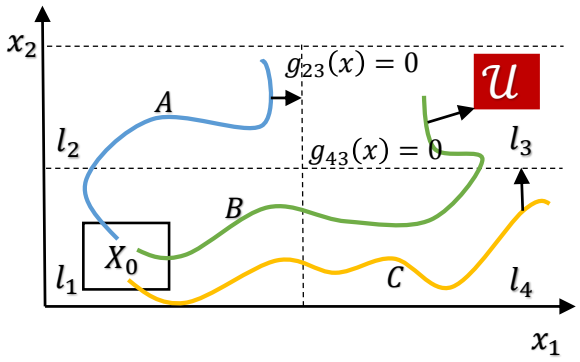


Figure 2.8: Trajectories B, A and C Improve Locally by Descending Toward the Unsafe Set, Guard g_{43} and Guard g_{23} Respectively.

Guards should be mutually exclusive.

4. *Guards* are of the form $g(x,t) = 0$ and *Reset* maps are functions of the form $x' = h(x)$, where g and h are C^1 functions. For all the states that satisfy a *Guard* condition the corresponding *Reset* map should satisfy $\frac{\partial h}{\partial x}\big|_x \neq 0$.

5. The trajectory $\eta(h_0, u(t), t)$ returned by the first stage, from which we descend, enters the location of the unsafe set.

The last assumption is made so that our problem be well-defined (note that the robustness will have finite value only if the trajectory enters an unsafe location). The task of finding such an initial condition h_0 is delegated to the higher-level stochastic search algorithm. If finding such a trajectory for the higher-level stochastic algorithm is hard, we can still improve our trajectories locally by descending toward the guards as pictured in Fig. 2.8.

Extending sensitivity analysis to the hybrid case is not straightforward and even in the case that there is no reset in transitions and the state stays continuous, a discontinuity often appears in the sensitivity function that needs to be evaluated [37].

Consider piece-wise constant exogenous inputs are applied to the system, and without loss of generality, assume that the hybrid trajectory undergoes a single transition from the location of the initial condition l_0 to the location of the critical predicate $l_{\mathcal{U}}$ at time τ.

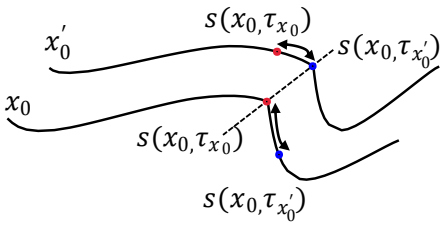


Figure 2.9: Assuming $\tau_{X_0} < \tau_{X_0'}$, Trajectories Are under Different Dynamics for All the Times $t \in [\tau_{X_0}, \tau_{X_0'}]$, Where τ_{X_0} and $\tau_{X_0'}$ Are Transition Times for $\tilde{x}_i(X_0, .)$ and $\tilde{x}_i(X_0', .)$ Respectively.

Assuming that $\mathbf{x}(x_0^{\mathbf{i}}, w^{\mathbf{i}})$ is the continuous part of the hybrid trajectory of the system, inside each of these locations, sensitivity matrices can be computed using the following ODEs:

$$\begin{aligned} \dot{s}_{x_0}(t) &= \partial_1 f(x(t), w(t)).s_{x_0}(t), \\ \dot{s}_\sigma^k(t) &= \partial_1 f(x(t), w(t)).s_\sigma^k(t) + \delta_k.\partial_2 f(x(t), w(t)), \quad t \in [\tau^{k-1}, T], \end{aligned} \tag{2.15}$$

where δ_k is equal to one if $t \in [\tau^{k-1}, \tau^k]$ and zero, otherwise. The initial and boundary conditions of the ODEs are as follows:

$$\begin{aligned} s_{x_0}(0) &= I_{n_x}, \qquad s_{x_0}(\tau^+) = r_{x_0} \\ s_\sigma^k(t) = 0, \quad t \in [0, \tau^{k-1}], &\qquad s_\sigma^k(\tau^+) = r_w^k, \text{ if } \tau^+ \geq \tau^{k-1} \end{aligned} \tag{2.16}$$

where τ^+ is the right hand side limit of the transition time τ that satisfies $(\mathbf{x}(x_0, w(\tau), \tau), \tau) \in Gu((l_0, l_\mathcal{U}))$. We will calculate r_{x_0} and r_w^k, shortly. Consider that even if there is no reset, this jump happens in the state triggered transitions since neighboring trajectories have different transition times and as a result they are under different dynamics during the time between their transition times (see Fig. 2.9).

Assume that $g(\mathbf{x}(x_0, w(t), t), t) = 0$ activates the guard condition between l_0 and l_u and causes a transition, and $Re(x, (l_1, l_2)) = h(x)$. Denote the transition time by $\tau(x_0, w)$.

Then, r_{x_0} and r_w^k can be computed as:

$$\begin{aligned} r_{x_0} &= \frac{\partial h}{\partial x}(s_{x_0}(\tau^-) + (\frac{\partial h}{\partial x}f^- - f^+)\partial_1\tau) \\ r_w^k &= \frac{\partial h}{\partial x}(s_\sigma^k(\tau^-) + (\frac{\partial h}{\partial x}f^- - f^+)\partial_2\tau^T) \end{aligned} \tag{2.17}$$

where $\frac{\partial h}{\partial x} = \frac{\partial h}{\partial x}\big|_{\mathbf{x}(x_0,w(\tau^-),\tau^-)}$, and f^- and f^+ are equal to $f_{l_0}(\mathbf{x}(x_0, w(\tau^-), \tau^-), w(\tau^-), \tau^-)$ and $f_{l_\mathcal{U}}(\mathbf{x}(x_0, w(\tau^+), \tau^+), w(\tau^+)), \tau^+)$ respectively, and:

$$\partial_1\tau = -\frac{\partial_1 g^T.s_{x_0}(\tau^-)}{\partial_1 g^T.f^- + \partial_2 g}, \qquad \partial_2\tau = -\frac{\partial_1 g^T.s_\sigma^k(\tau^-)}{\partial_1 g^T.f^- + \partial_2 g} \tag{2.18}$$

Hence, one can compute $\delta x_0^{\mathbf{i}}$ and $\delta\sigma^{k,\mathbf{i}}$ as:

$$\begin{aligned} \delta x_0^{\mathbf{i}} &= -\partial\mu_*^{\mathbf{i}}\big|^\top_{\mathbf{x}(x_0^{\mathbf{i}},w^{\mathbf{i}},t_*^{\mathbf{i}})} s_{x_0}(t_*^{\mathbf{i}}) \\ \delta\sigma^{k,\mathbf{i}} &= -\partial\mu_*^{\mathbf{i}}\big|^\top_{\mathbf{x}(x_0^{\mathbf{i}},w^{\mathbf{i}},t_*^{\mathbf{i}})} s_\sigma^k(t_*^{\mathbf{i}}) \end{aligned} \tag{2.19}$$

in which $s_{x_0}(t_*^{\mathbf{i}}), s_\sigma^k(t_*^{\mathbf{i}})$ are the values that sensitivity matrices of the hybrid system take at the critical time $t_*^{\mathbf{i}}$.

Consider the maneuvering object example modelled in Eq. (2.12) and its requirement. Using Eq. (2.19) or its negative we can reduce/increase $J^{\mathbf{i}}$ and consequently the robustness function until a falsifying/satisfying behavior is found (see the top figure in Fig. 2.10) or to get the trajectory closer to the location of the critical/unsafe predicate (see bottom figure in Fig. 2.10). Our search is over the initial positions in $[0, 1] \times [0.5, 1]$, and the input signals $F_1(t), F_2(t) \in [-1, 1]$; other states are zero initially. We used piece-wise constant inputs with 11 variables for each $F_1(t)$ and $F_2(t)$. So the overall search is over 24 dimensions. Starting from a trajectory that satisfies the requirement with the robustness value equal to 0.2950, our method improves the robustness value to 0.8599. The projection of the trajectories into the $x_1 - x_2$ plane is shown in Fig. 2.10, where light gray trajectories are refined to dark gray ones. In the bottom figure, one can see that even if the trajectory from which we want to descend does not enter the goal set location, we are still able to improve the trajectory by descending toward the adjacent guard with the least distance from that set.

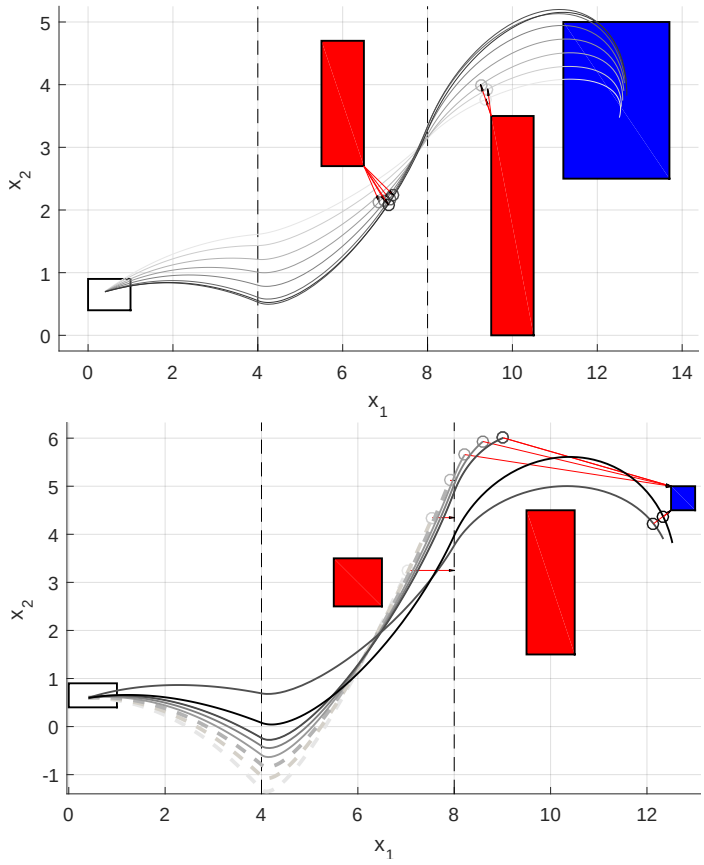


Figure 2.10: Shown in the Top: Improving the Robustness of the Trajectories with Respect to the Specification φ_2 From .2950 To .8599. Red Arrows Show the Steepest Ascent Direction. Shown in the Bottom: Trajectories That Do Not Enter the Goal Set Location (Dashed Trajectories Here) Can Still Improve by Descending Toward the Guard Set (Dashed Line at $x_1 = 8$).

To determine the effect of applying GD local search method to global search methods like Simulated Annealing (SA), we performed a statistical study in which we compare the combination of SA and GD (SA+GD) with SA only. To combine SA and GD, we apply GD algorithm whenever the samples picked by SA return a robustness value less than some threshold value r_T.h In our experiment we ran SA and SA+GD for 150 times with equal total number of samples $N = 100$ and $r_T = 2.5$ to automatically search for initial conditions and inputs that satisfy the specification with $\mathcal{U}_1 = [5.5, 6.5] \times [2.5, 3.5]$, $\mathcal{U}_2 = [9.5, 10.5] \times [1.5, 4.5]$, $G = [12.5, 13] \times [4.5, 5]$. In order to satisfy the requirement, we try to falsify its negation. The results are shown in Table (2.3). The improvement in finding falsifying trajectories is clear from the total number of falsifications in the first row. Also, since GD gets a chance to improve the performance only if SA finds a robustness value less than r_T, we added the second row which shows in how many percent of the cases falsification is achieved if SA finds a robustness value less than r_T. While the average of the best robustness value for all the tests is better for SA+GD algorithm, it is slightly better for SA if we only consider non-falsified cases. We can conclude that even if SA finds small robustness values, it is hardly able to further decrease it. As the constant budget in the comparison is "equal total number of simulations", we can claim that SA+GD can help improve the results if

Table 2.3: Comparing SA and SA+GD Results for the Maneuvering Object Example

Optim. method	SA	SA+GD
num. of total falsification	4/150	16/150
% of falsification if SA finds $r \leq r_T$	13.33%	39.02%
Avg. min-Rob. (all the cases)	9.1828	8.4818
Avg. min-Rob. (not falsified cases)	9.4278	9.4968
min. min-Rob. (not falsified cases)	0.0080	0.0059
max. min-Rob. (not falsified cases)	13.1424	13.0880

simulations/experiments are costly. Choosing different design parameters might lead to even better experimental results.

All the above methodologies for local-optimization-guided falsification have been implemented and are available as part of S-TaLiRo [114].

2.5 Conclusions and Future Work

In this section, we presented methods based on sensitivity analysis and optimal control that help to solve the optimization-based falsification problem under different assumptions about the system model and external inputs. The experimental results demonstrated that guided by a local search based on our proposed methods, we can find falsifications that previous tools cannot, furthermore, falsifications can be found faster.

Due to the complexity of the robustness function, optimization-based falsification can be further improved with more advanced optimization techniques. Hence, future work can include research on more efficient ways for optimizing the robustness function. Different stochastic and analytic methods for falsification can be explored. Momentum-based methods (including the conjugate method), as an example, compute the search direction as a weighted sum of the negative gradient direction and the search direction in previous steps. These methods have the potential to help the local search as they reduce the risk of getting stuck in a local minimum and speed up the convergence considerably in cases where the process would otherwise zig-zag heavily [108].

Future work can also include the convergence properties of the presented algorithms and the conditions under which the results based upon the local approximation of the robustness function in Eq. (2.6) can be extended to the robustness function itself.

Chapter 3

REQUIREMENT-BASED CONTROL SYNTHESIS

3.1 Introduction

Designing controllers for satisfying system properties specified in STL has been studied before [86]. In [109], a method for designing open-loop controllers for STL specifications based on MPC using Mixed Integer Linear Program (MILP) solvers is introduced. Authors design open-loop strategies that satisfy STL specifications using randomized tree search in [127]. The work in [105] introduces a smooth approximation to the robustness function of the STL formula, which allows using gradient-based optimizers to design robust controllers for the satisfaction of STL formulas. In general, open-loop controllers need to be designed and stored in advance, and MPC controllers need to be computed at the run-time. Computing MPC controllers for high-order systems include solving large optimization problems at each step [67], so in general, they do not scale to higher-order systems since their run-time computation is challenging. On the other hand, storing the results of the offline computation of open-loop or MPC controllers (e.g., as a lookup table) requires too much memory [78], so using Neural networks (NN) to approximate controllers [113, 77], learn optimal policies offline [61], or to improve baseline controllers [145] is very common. The application of NNs in high assurance systems has been studied in [118].

Model-based Reinforcement Learning methods are also another line of work close to our work [129, 143, 91]. Recently, Reinforcement Learning methods have been developed that make policy learning w.r.t temporal logic specifications possible, see e.g., [57, 94]. Despite the recent advances using Reinforcement Learning methods, providing guarantees for the behavior of systems trained using them is challenging. This is due to the complicated behavior of the Neural Networks which are used in these systems as function approximators. These systems sometimes have safety-critical roles, and as a result, the problem of testing

and system-level verification is of utmost importance.

The application of NN to control dynamical systems has a long history [71, 62, 118]. More recently, due to computational advances and available data, there has been a renewed interest in the utilization of NN in control systems. As discussed before, because of the limitations that verification approaches usually have in the application, testing methods are used to find falsifying system behaviors. For instance, the problem of testing autonomous vehicles equipped with NNs for perception, guided by system-level requirements is studied in [2, 125, 40]. The falsifying/adversarial samples that are found during the testing process can be used for retraining the NNs in order to improve their accuracy. Adversarial training is used in [41, 42] for improving the performance of the NNs used in an autonomous vehicle for perception.

The work in [44] is also related to our work wherein Neural network controllers are designed to satisfy reachability and region stability properties. In section 3.2, Using a smooth approximation of the STL robustness function, we will design controllers for general STL formulas. Another difference is that to guarantee safe worst-case performance, our neural networks are trained to maximize the worst-case robustness values over the space of possible initial conditions. Besides, in a post-training procedure, our algorithm searches for adversarial samples and retrain the NN using them. On the other hand, in section 3.3, we focus on training NN controllers for achieving reach-avoid specifications. The reason for focusing on a more limited class of requirements is that to achieve these types of requirements, at each state that the system visits there exist nonlinear/quadratic programs constrained by a set of inequalities linear in the control input that if solved can guarantee safety while guiding the system to the goal set [13]. Hence, we can use the solutions to these programs as expert demonstrations of good behavior to create data sets for training the NN. Consequently, the training can be achieved much more efficiently using imitation learning [111]. The constraints that enable safety in the aforementioned programs are related to the evolution of a function that creates a barrier around the unsafe set of states. This function

is called a Barrier Function. In this thesis, we extend the results based on the BFs to assure safety in presence of external inputs and disturbances.

The BF theory has been instrumental in developing safety-critical controllers for non-linear systems. Control Barrier Functions (CBF) have enabled the design of provable safe feedback controllers for several different systems such as adaptive cruise control [14], bipedal robot walking and long-term autonomy [13].

While the results presented in section 3.3 verify safety in the worst-case in the presence of uncertainty with hard bounds on its magnitude, when the disturbance has stochastic characteristics, we need to consider BFs in a stochastic setting. In [97], authors design BFs for nonholonomic systems in unknown environments modeled using stochastic semantic maps. Stochastic Barrier Functions (SBF) have been used in [107, 74] for verification of safety and temporal logic properties of stochastic systems. The authors in [116] use SBFs for finite-time stochastic system verification and feedback control design through solving sum-of-squares programs. However, finding such a closed-loop controller for more complicated systems and environments, and when uncertainty is higher, is not possible. To handle these cases, researchers consider the real-time computation of control inputs based on SBFs and using optimization methods. For instance, using QPs to design real-time control inputs for stochastic systems that maximize the probability of invariance of a set C has been studied in [28] using CBFs for complete and incomplete information and in [117] using high-relative degree SBFs. However, these works derive conditions that phase out the probability of eventually entering the unsafe set. Such a control input, if it exists, may be very conservative in many applications. Hence, in this section 3.4, we consider the real-time design of control inputs that bound the probability of a finite-time failure.

In motion planning, an Autonomous Mobile System (AMS) is required to move from a start location to a goal location while avoiding collisions with other agents, dynamic and static. In this work, we provide a method for control synthesis to solve this start-to-goal motion problem while bounding "the probability of a collision in finite-time (risk)". We

utilize a Barrier Function (BF) candidate whose level set of value one contains the unsafe set of the AMS and other agents' states [107]. The probabilistic nature of the behavior of agents is modeled using Stochastic Differential Equations (SDEs) [104]. Conditions on the BF candidate that bound its expected value over a finite-time horizon are derived based on the model of the AMS and the stochastic model of other agents. These conditions can be used to compute an upper bound on the risk [89, 116]. The upper bounds depend on the state of the system and the parameters used in the conditions that control the evolution of the expected value of the BF candidate. As a result, in a given state, we can bound the risk to the desired threshold by constraining the aforementioned parameters. We use these constraints to choose the values of the parameters and the control input. To lead the AMS to a goal location, our method unifies the conditions imposed by the BF for bounding the risk, with the conditions imposed by a Lyapunov function in a Quadratic Program (QP) which can be solved in real-time. The obtained sub-optimal control input will lead the AMS to the goal set while bounding the probability of entering the unsafe set in a finite time. We first develop the theoretical framework for the composition of conditions that control the finite-time growth of BFs with risk-bounds to derive sufficient conditions for risk-bounded control. The advantage of the risk-based formulation of the conditions is that it allows for a less conservative control design framework. Then we combine the aforementioned conditions with Lyapunov conditions in a QP whose on-the-fly solution solves the risk-bounded start-to-goal problem. Finally, we demonstrated the application on a lane-changing scenario on a highway with dense traffic.

3.2 Training Feedback Neural-network Controllers for TL Requirements

3.2.1 Preliminaries

We consider the discrete time nonlinear dynamical systems described below:

$$\Sigma : \begin{cases} x_{t+1} = f(x_t, u_t) \\ y_t = g(x_t) \end{cases} \tag{3.1}$$

where $x_t \in X \subset \mathbb{R}^{n_x}$ is the system state, $y_t \in \mathbb{R}^{n_y}$ is the system output, $u_t \in U \subset \mathbb{R}^{n_u}$ is the control input, $f : X \times U \to X$ is a differentiable function of its arguments, and $x_0 \in X_0 \subset X$ is the initial state. Given an initial state $x_0 \in X_0$ and a control sequence $\mathbf{u^N} = u_0, u_1, ...u_N$, s.t, $\forall t : u_t \in U$, the bounded time solution of the system is denoted as $\mathbf{x}(x_0, u^N) = x_0, x_1, ...x_N$, where $\forall t : x_t \in X$ and $x_{t+1} = f(x_t, u_t)$.

Smooth approximation of STL robustness function: The robustness function as introduced in section 2.2.2 is a non-differentiable function of its inputs as it consists of composition of max and min functions. As a result, in order to be able to use optimizers that require differentiability, we also introduce the smooth approximation [105] of the robustness function $\tilde{\rho}_\varphi$ which replaces the min and max functions with soft min ($\overline{\min}$) and soft max ($\overline{\max}$) functions:

$$\overline{\min}(x, \alpha) = -\frac{1}{\alpha} ln(\Sigma_{i=1}^{N} e^{-\alpha x_i}) \tag{3.2}$$

$$\overline{\max}(x, \alpha) = -\overline{\min}(-x, \alpha) \tag{3.3}$$

where x_i is the i^{th} argument of x. This function approaches the min function as $\alpha \to \infty$. It's been shown in [105] that an ϵ can be computed such that $|\rho_\varphi - \tilde{\rho}_\varphi| < \epsilon$. As a result, $\tilde{\rho}_\varphi > \epsilon$ guarantees satisfaction of φ.

Feedforward Neural Networks (FNN) are static or memory-less networks. The most general FNN is Multi-Layer Perceptron (MLP), which can approximate any nonlinear function. We can define FNNs using the number of their inputs n_I, outputs n_o, layers n_l, and weights $(W_i, b_i)_{i=1}^{n_l}$ which connect the i^{th} layer to the $i + 1^{th}$ layer or the outputs. The i^{th} layer applies the following function to its inputs $\mathcal{U}_i \in \mathbb{R}^{m_i}$:

$$\mathcal{U}_{i+1} = \phi_i(W_i^T \mathcal{U}_i + b_i) \quad i \in \{1, 2..., l\} \tag{3.4}$$

where ϕ_i is an activation function chosen to be one of the continuous nonlinear functions: ReLU, tanh, arctan, logistic or sigmoid.

The weight matrices W_i and the bias vectors b_i should be adjusted during NN training. The function $\mathcal{N}$ formed by the decomposition of the neurons in Eq. (3.4), calculates the

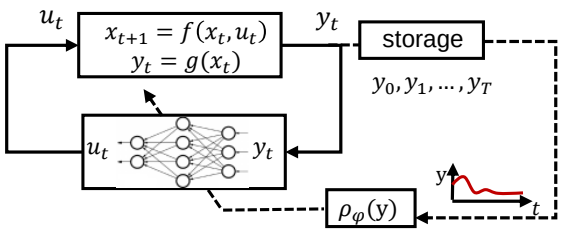


Figure 3.1: The Closed-loop System with Its NN Controller to Satisfy an STL Formula φ

final output of the FNN given the inputs and $(W_i, b_i)_{i=1}^{n_l}$. The weights W_i and the bias vectors b_i are collectively denoted as W.

3.2.2 Problem Formulation

Consider $h : \mathbb{R}^{n_y} \to U$ to be a differentiable function. The controller $u_t = h(y_t)$ should be designed in order to maximize the robustness value. Since the function spaces are infinite-dimensional, the function h needs to be parameterized. As neural networks are known to be universal function approximators [69], we consider h to be a Feedforward Neural Network (FNN) parameterized using its weights W. We write the output of the NN as $u_t = \mathcal{N}(y_t, W)$. As a result, the closed loop system (shown in Fig. 3.1) can be described as:

$$\begin{cases} x_{t+1} = f(x_t, \mathcal{N}(y_t, W)) \\ y_t = g(x_t) \end{cases} \tag{3.5}$$

Given an initial condition $x_0 \in X_0$, the bounded time solution of the closed loop system (3.5) is $\mathbf{x}(x_0, W) = x_0, x_1, ...x_N$, where $\forall t : x_t \in X$, $y_t = g(x_t)$, and $x_{t+1} = f(x_t, \mathcal{N}(y_t, W))$. In order for the NN to satisfy the constraints on the input, the NN output is saturated.

Consider the dynamical system $\mathcal{S} = (\Sigma, X_0, U)$, where Σ is described in Eq. (3.1), X_0 is a set of initial conditions, and U is the set of admissible control inputs. We would like to design a NN controller $\mathcal{N}$ for the system $\mathcal{S}$ that satisfies the input constraint such that the minimum robustness value over the set of initial conditions X_0 is maximized. The problem is formally defined as follows:

Problem 3.2.1. *Given the dynamical system $\mathcal{S}$, an STL formula φ whose horizon is N*

(see Def. 1), and a fixed architecture for $\mathcal{N}$, solve the following optimization problem for training the NN:

$$W^* = \underset{W}{argmax} \min_{x_0 \in X_0} \rho_\varphi(\mathbf{x}(x_0, u^N)) \tag{3.6}$$

$$s.t \begin{cases} x_{t+1} = f(x_t, u_t) \\ y_t = g(x_t) \\ u_t = \mathcal{N}(y_t, W) \\ \mathbf{x}(x_0, u^N) = x_0, x_1, ..., x_N \end{cases}$$

3.2.3 Solution Approach

To accomplish both exploration and exploitation in the space of NN parameters, in this work, we use an evolution-based method for optimization with a large enough population size called CMA-ES and combine it with the exploitation power of gradient-based methods. The NN weights W have a complex chained effect in the cost function of Prob. 3.2.1 through the sequence of states. Based on our experiments, direct policy search variants of reinforcement learning approaches based on evolution strategies [64], like covariance matrix adaptation evolution strategy (CMA-ES) algorithm, were not able to find a good set of parameters (W) without additional support. Nonetheless, their exploration power is usually able to bring the parameters close to their optimal value. On the other hand, with a good initialization, gradient-based approaches were found to be very effective in training neural networks using backpropagation approaches [22].

In this section, we use the best of two worlds: the exploration power of the evolution-based methods with a large enough population size, and the exploitation power of gradient-based methods. An overview of the solution approach is depicted in Fig. 3.2.

Neural Network Controller Architecture: We assume a given neural network architecture $\mathcal{N}$ for the controller with $n_I = n_y$, $n_o = n_u$, which satisfies the input constraint. In other words, $\mathcal{N}$ is a function with a range equal to U ($\mathcal{N} : \mathbb{R}^{n_y} \rightarrow U$). To satisfy the input constraints, the output layer can apply a scaled Hyperbolic tangent sigmoid transfer

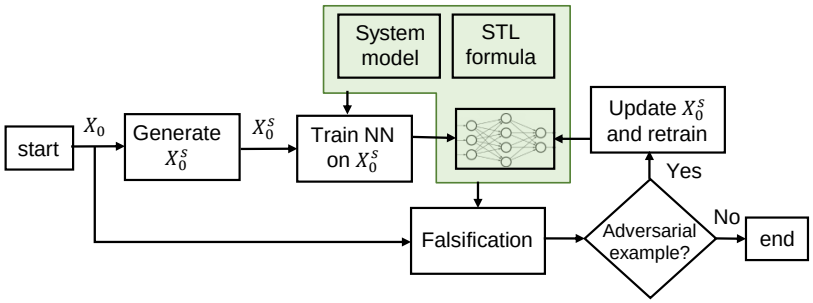


Figure 3.2: Training Framework

function (see Fig. 3.3) to its inputs. Assuming $\overline{u}$ and $\underline{u}$ to be the upper and lower bounds on U respectively, using the following activation function ensures the satisfaction of the input constraint.

$$\overline{tansig}(x, U) = \underline{u} + \frac{\overline{u} - \underline{u}}{2}(tansig(x) + 1) \tag{3.7}$$

Remark 2. *If an output feedback control law exists that satisfies the STL formula $\forall x_0 \in X_0$, then provided enough data, and using an appropriate neural network architecture, the optimal weights W can be found such that the system of Eq. (3.5) satisfies φ for all $x_0 \in X_0$. Note however that, in general, for satisfying the temporal and reactive properties of STL formulas, more features than systems' current states/outputs or a controller with memory may be required.*

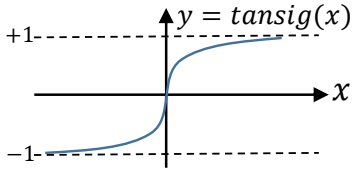


Figure 3.3: Tangent Sigmoid Function

NN Weights Initialization Using Global Optimization: In this step of the training, we look for a set of parameters that on average maximizes ρ_φ for a set of randomly selected initial states $x_0 \in X_0$ which is represented as X_0^s. We denote the number of samples in this set with $|X_0^s|$. Starting from a set of randomized weights and bias values W, our

global optimizer, minimizes the following loss (cost) function

$$C(W) = \sum_{x_0 \in X_0^s} c_\varphi(\mathbf{x}(x_0, W))$$

where $c_\varphi = -\rho_\varphi$. In this stage, the above cost function can be minimized using any method for global optimization. In this work, we use the CMA-ES algorithm [64], which is a derivative-free, evolution-based numerical optimization method. CMA-ES is primarily a local optimization approach, but it also has been reported to be reliable and highly competitive for global optimization when using larger population sizes [63]. In our experiments, we use the following relation[1] to choose the population size

$$\text{population size} = 10 \times (4 + round(3 \times ln(|W|))) \tag{3.8}$$

where ln is the natural logarithm function and $|W|$ is the number of NN weights and bias parameters.

Note that the minimizer of the above cost function W_G^* does not necessarily maximize the worst-case robustness value as required in Prob. 3.2.1. The reason is that, firstly, a finite number of randomly selected initial conditions are considered in the cost formulation, and secondly, W_G^* is trained on the average performance rather than the worst-case performance. Specifically, using $\mathcal{N}(y_t, W_G^*)$ as the controller:

1. There may exist $x_0 \in X_0 - X_0^s$ such that the solution to the closed-loop system (3.5) does not satisfy the specification, since the training is done using a finite set of initial states.

2. Or there may exist adversarial samples $x_0 \in X_0^s$, as the cost formulation is over the average robustness value.

As a result, in this work, we design a two-player game that uses adversarial examples to improve the weights of the neural network.

NN Weight Update Using Lagrange Multipliers for Incremental Robustness

[1] This is suggested in the Matlab's implementation of CMA-ES

In the model-based Reinforcement Learning, backpropagation of the desired change of the objective function to the weights of the NN is used for policy improvement [29].

In what follows, we derive a weight update law based on an optimal control approach to incrementally improve the robustness value. As gradient-based approaches require differentiability of the objective function, we alter the optimization problem of Eq. (3.6) in Prob. 3.2.1, as follows:

$$\max_{W} \ \overline{\min_{x_0 \in X_0^s}} \ \tilde{\rho}_{\varphi}(\mathbf{x}(x_0, u^N)) \tag{3.9}$$
$$s.t \quad \begin{cases} x_{t+1} = f(x_t, \mathcal{N}(g(x_t), W)) \\ \mathbf{x}(x_0, u^N) = x_0, x_1, ..., x_N \end{cases}$$

Consider the following objective function for one of the trajectories starting from $x_0 \in X_0^s$ and subject to the dynamical constraints:

$$\begin{cases} J_{x_0}(W) = \tilde{\rho}_{\varphi}(x_0, x_1, ..., x_N), \\ x_{t+1} = f(x_t, \mathcal{N}(g(x_t), W)) \end{cases}$$

using the Lagrange multipliers associated with the state equations which are called co-states, the dynamical constraints can be incorporated into the objective function, as follows:

$$\begin{aligned} \bar{J}_{x_0}(W) &= \tilde{\rho}_{\varphi}(x_0, x_1, ..., x_N) + \sum_{t=0}^{N-1} \lambda_{t+1}^{\top}\big(f(x_t, \mathcal{N}(g(x_t), W))) - x_{t+1}\big) \\ &= \tilde{\rho}_{\varphi}(x_0, x_1, ..., x_N) + H_0 - \lambda_N^{\top} x_N + \sum_{t=1}^{N-1} \big(H_t(x_t, W) - \lambda_t^{\top} x_t\big) \end{aligned} \tag{3.10}$$

where λ_ts are the co-states, and $H_t = \lambda_{t+1}^{\top} f(x_t, \mathcal{N}(g(x_t), W))$ is the Hamiltonian. Thus, we have:

$$\delta \bar{J}_{x_0}(W) \ = \ \sum_{t=1}^{N-1} \big(\frac{\partial H_t}{\partial x_t} \ + \ \frac{\partial \tilde{\rho}_{\varphi}}{\partial x_t} \ - \ \lambda_t^{\top}\big)\delta x_t \ + \ \big(\frac{\partial \tilde{\rho}_{\varphi}}{\partial x_N} \ - \ \lambda_N^{\top}\big)\delta x_N \ + \ \sum_{t=0}^{N-1} \frac{\partial H_t}{\partial W}\delta W \tag{3.11}$$

The following equations enforce $\delta \bar{J}_{x_0}(W) > 0$:

$$\lambda_N^\top = \frac{\partial \tilde{\rho}_\varphi}{\partial x_N}, \tag{3.12}$$

$$\lambda_t^\top = \frac{\partial H_t}{\partial x_t} + \frac{\partial \tilde{\rho}_\varphi}{\partial x_t} = \lambda_{t+1}^\top (\frac{\partial f}{\partial x_t} + \frac{\partial f}{\partial u_t}\frac{d\mathcal{N}}{dx_t}) + \frac{\partial \tilde{\rho}_\varphi}{\partial x_t}, \tag{3.13}$$

$$\delta W = \sum_{t=0}^{N-1} \frac{\partial H_t}{\partial W} = \sum_{t=0}^{N-1} \lambda_{t+1} \frac{\partial f}{\partial u_t}\frac{\partial \mathcal{N}}{\partial W} \tag{3.14}$$

where $\frac{\partial f}{\partial x_t} = \frac{\partial f}{\partial x_t}\big|_{(x_t,u_t)}$, is the Jacobian matrix of the open loop system, $\frac{\partial f}{\partial u_t} = \frac{\partial f}{\partial u_t}\big|_{(x_t,u_t)}$ is the derivative of the open loop system w.r.t its inputs, $\frac{\partial \tilde{\rho}_\varphi}{\partial x_t}$ is the derivative of the smooth robustness function to its t^{th} argument, $\frac{d\mathcal{N}}{dx_t} = \frac{\partial \mathcal{N}}{\partial y_t}\frac{\partial g}{\partial x_t}$ is the derivative of the NN to its inputs multiplied by their derivative to the states, and $\frac{\partial \mathcal{N}}{\partial W}$ is the derivative of the NN w.r.t its weights which is usually readily available. Equations (3.12) and (3.13) provide the terminal condition and the backward dynamics for the co-states, respectively. Equation (3.14) provides the desired change in the weights of the neural network in order to increase the objective function $J_{x_0}(W)$. As a result, changing the weights of the NN in the direction δW using a small enough step size will increase the robustness function ρ_φ for a single trajectory starting from x_0 -assuming α is large enough. To emphasize the dependency of the desired change in the weights on the initial condition x_0, we will denote it as δW_{x_0}. As a result, in order to improve the objective function in Eq. (3.9) which depends on all the samples in X_0^s rather than just one sample, the weights of the NN should be changed in the following direction:

$$\Delta W = \sum_{x_0 \in X_0^s} k_{x_0}(i)\delta W_{x_0} \tag{3.15}$$

where k_{x_0} is a vector of length $|X_0^s|$ whose i^{th} element is proportional to the effect of the smooth robustness value $\tilde{\rho}_\varphi(\mathbf{x}(x_0, W))$ in the value of the overall objective function $J = \overline{\min}_{x_0 \in X_0^s} \tilde{\rho}_\varphi(\mathbf{x}(x_0, u^N))$. We write these robustness values collectively as r_{x_0}, as a result:

$$k_{x_0} = \frac{\partial \overline{\min}(r_{x_0})}{\partial x} \tag{3.16}$$

where $\frac{\partial \overline{\min}}{\partial x}$ is the derivative of the soft min function in Eq. (3.2). Note that changing W in the direction $\delta W_{\underline{x}_0}$, where $\underline{x}_0$ corresponds to the minimum robustness value in X_0^s, may

result in a decrease in the worst-case robustness value. The reason is that the change in the robustness value for the other samples in X_0 are not considered and there could exist some $x_0 \in X_0, x_0 \neq \underline{x}_0$ which may worsen the minimum robustness value. Considering only the worst sample in the weight update law breaks the completeness of the method, and may cause oscillations between improving the policy for two $x_0 \in X_0^s$. However, using Eq. (3.15) for updating the weights ensures an improvement in the worst-case behavior in X_0^s under mild assumptions, as it considers the effect of the weight perturbation in all the samples. In practice, in order to reduce the computational complexity, one can remove the terms for which $k_{x_0}(i) << max(k_{x_0})$ in Eq. (3.15) and avoid calculating δW_{x_0} for them.

Proposition 1. *There exists a small enough step size $\bar{h}_w > 0$ such that for all $0 < h_w \leq \bar{h}_w$:*

$$\overline{\underset{x_0 \in X_0^s}{min}}\ \tilde{\rho}_\varphi(\mathbf{x}(x_0, W + h_w \Delta W) > \overline{\underset{x_0 \in X_0^s}{min}}\ \tilde{\rho}_\varphi(\mathbf{x}(x_0, W))$$

Proof. This is a direct result of the design choice since by design, ΔW is a descent direction of the cost function in Eq. (3.9). □

Choosing the step size Theoretically, the ideal step size for updating W is the upper-bound for all the $\bar{h}_w$ values that satisfy the condition in Prop. 1. In practice, however, finding such an ideal step size is not possible. Choosing a proper step size h_w is important for convergence of the gradient-based policy search methods. Choosing very large step sizes can result in passing better level sets and local extremes, and choosing very small step sizes will cause insignificant and slow improvements. As a result, in gradient-based optimization, the step size is usually changed adaptively.

In this work, we choose step sizes proportional to the norm of the weight matrix $|W|$, $h_w = k|W|$. At the first iteration, we pick $k = k_0 << 1$ and change it adaptively as described in Alg. 3 afterward. The value returned from Alg. 3 in one iteration is used as the initial step size in the next iteration.

Algorithm 3 step size adaptive update

Data: The dynamical system $\mathcal{S}$, W, ΔW, initial step size k, maximum number of iterations I_m, increase and decrease rates $\overline{\alpha}$ and $\underline{\alpha}$, and the specification φ

Result: The final step size k, and the improved weights W^* if W is not a local optima, and a local optima flag 'LO' o.w

$dec\_flag \leftarrow$ false, $inc\_flag \leftarrow$ false, $LO \leftarrow$ false

$h_w = k|W|$, $W' = W + h_w \Delta W$, $J^* = \overline{\min}_{x_0 \in X_0^s} \tilde{\rho}_\varphi(\mathbf{x}(x_0, W))$

for $i = 1, ..., I_m$ **do**

  $\Delta J = \overline{\min}_{x_0 \in X_0^s} \tilde{\rho}_\varphi(\mathbf{x}(x_0, W') - J^*$

  if $\Delta J > 0$ **then**

    $J^* = \overline{\min}_{x_0 \in X_0^s} \tilde{\rho}_\varphi(\mathbf{x}(x_0, W'))$

    if dec_flag **then**

      return k, W^*

      break

    end

    $k \leftarrow \overline{\alpha} k$, $W^* \leftarrow W'$, $h_w = k|W|$, $W' = W + h_w \Delta W$, $inc\_flag \leftarrow$ true

  else

    if inc_flag **then**

      return k, W^*

      break

    end

    $k \leftarrow \underline{\alpha} k$ $h_w = k|W|$, $W' = W + h_w \Delta W$, $dec\_flag \leftarrow$ true

  end

end

$LO \leftarrow$ true **return** k, W, LO

Training on X_0^s: To find a controller that maximizes the worst-case robustness value in the set of randomly selected initial conditions $X_0^s \subset X_0$, we use a global optimizer to explore the search space and find a set of weights W that optimize the average robustness value on X_0^S. Given the limited power of randomized search methods, the resulting weights from the

previous stage are used to initialize the gradient-based local search described in Sec. 3.2.3 which is used later to improve the worst-case performance in X_0^s[2]. This is described in Alg. 4. F If the purpose is to only satisfy the requirement, then the while loop can be exited as soon as the worst-case robustness value on X_0^s becomes positive.

Adversarial guided Training: As described earlier, finding a set of weights which maximally satisfies the specification φ on the set X_0^s, does not guarantee the worst case optimality for all $x_0 \in X_0$, nor does it guarantee the satisfaction of the formula in X_0. As a result, after training the weights on X_0^s, we use a falsification paradigm in order to find worst case adversarial samples for which $\rho_\varphi < 0$ and $|\rho_\varphi|$ is maximized. Given the NN controller and its weights W, a falsification method, aims to solve the following problem [137, 136]:

$$\begin{aligned} & \underset{x_0 \in X_0}{\operatorname{argmin}} \ \rho_\varphi(\mathbf{x}(x_0, W)) \\ s.t \quad & \begin{cases} x_{t+1} = f(x_t, \mathcal{N}(g(x_t), W)) \\ \mathbf{x}(x_0, W) = x_0, x_1, ..., x_N \end{cases} \end{aligned} \tag{3.17}$$

If an adversarial sample x_0^a was found, we will add the sample to X_0^s, and backpropagate the robustness ascent direction into the NN weights W until no more progress can be achieved (Alg. 4 without the initial global training), once a set of weights W was found that maximally satisfies φ on the updated set X_0^s, we look for another adversarial sample (See Alg. 5). If we were not able to satisfy φ for x_0^a the algorithm returns failure. The reason for a failure can be one of the followings:

- φ is not satisfiable on X_0.
- A state feedback controller is not enough for decision making. More information/features or dynamic controllers might be required.

[2]The solution to the Problem (3.9) can be negative: there may exist $x_0 \in X_0^s$ for which the specification is not satisfied even with the best set of weights, either since φ is hard or since the controller choice is not suitable

- The NN architecture is not suitable.
- There is no suitable policy neighboring W.

Figure 3.2 shows the overall learning framework.

If the falsification method provides testing coverage guarantees (e.g, see [5]), once no more adversarial samples were found, φ is satisfied on X_0 almost surely. One can also try to verify the property φ over the set X_0 using a verification method for dynamical systems including neural networks [44, 73, 43].

Algorithm 4 Training on X_0^s

Data: The dynamical system $\mathcal{S}$, the Specification φ, the randomly selected set of initial conditions X_0^s, and the NN architecture $\mathcal{N}$

Result: Set of the optimal parameters W

$W \leftarrow$ Use a global optimizer to find the set of weights W that optimizes the average robustness on X_0^S (Sec. 3.2.3)

$LO \leftarrow$ false

while $\neg LO$ **do**

  $\Delta W \leftarrow$ Use Eq. (3.15) to find the ascent direction for W

  $(W, LO) \leftarrow$ Use Alg. 3 to change W in the ΔW direction

end

return W

3.2.4 Experimental Results

The following experimental results were done in MATLAB 2018b. We use the Matlab implementation of CMA-ES [64] for global optimization. S-TaLiRo toolbox [16] is used to test the closed-loop system against the specification φ. The tool has different optimization algorithms. Specifically, in this work, we use the Simulated Annealing (SA) optimization method for finding the adversarial samples.

Vehicle Navigation: In this example, in order to evaluate the performance of the Lagrange multipliers approach, we use it without the help of the global search for updating

Algorithm 5 Retraining using adversarial samples

Data: The dynamical system $\mathcal{S}$, the Specification φ, and the randomly selected set of initial conditions X_0^s, the NN architecture $\mathcal{N}$ and its parameters W

Result: Set of the optimal parameters W

```
while true do
    x_0^a ← Use a falsification method to solve problem 3.17
    if ρ_φ(x(x_0^a, W)) < 0 then
        X_0^s ← X_0^s ∪ x_0^a   LO ← false
        while ¬LO do
            ΔW ← Use Eq. (3.15) to find the desired direction
            (W, LO) ← Use Alg. 3 to update W
        end
        if ρ_φ(x(x_0^a, W)) < 0 then
            return failure
            break
        end
    else
        return W
        break
    end
end
```

the weights of a NN which is used to guide a vehicle to accomplish a mission written in STL. The vehicle navigates in a 2D environment according to the following dynamics

$$\dot{x} = \begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} vcos(\theta) \\ vsin(\theta) \\ \frac{v}{L}tan(\gamma) \end{bmatrix} \tag{3.18}$$

where (p_x, p_y) is the x-y position of the center of the vehicle and θ is the angle that the vehicle's heading has with the x-axis. The inputs v, γ are the forward driving speed and the steering angle of the front wheels. The inputs are limited to the sets $[0, 5]$ and $[-\pi/4, \pi/4]$,

respectively. We assume that initially the car is at $(p_{x_0}, p_{y_0}) = (6, 8)$, but the heading angle can vary in the set $\theta_0 \in [-3\pi/4, \bar{\theta}]$. We choose $\bar{\theta} = -5\pi/8$ once and $\bar{\theta} = -\pi/2$ later. The system is simulated using a discretized step size $\Delta t = 0.05$ for 40 steps. The vehicle needs to visit goals 1 and then 2 in this specific order while avoiding an obstacle [94]. This can be expressed in STL using the following nested formula

$$\varphi = \Diamond_{[1,40]}\big(Goal_1 \wedge \Diamond Goal_2\big) \wedge \Box_{[1,40]}\neg \text{Unsafe}$$

where the Unsafe, $Goal_1$ and $Goal_2$ sets are 2D sets: $[1, 4]\times[2, 5]$, $[3, 4]\times[0, 1]$ and $[5, 6]\times[3, 4]$, respectively. These sets are shown in the left plot of Fig. 3.4. We design a NN with 3 inputs (system states), 2 hidden layers each with 5 neurons, and 2 outputs that are applied to the system as the control inputs v, γ. The NN has tangent Sigmoid activation functions and the input constraints are enforced by using scaled tangent Sigmoid functions in the outputs. Since the weight update law leads to local improvements in the weights, good weight initialization is important. In order to initialize the weights, we picked the weights randomly between -1 and 1 for 20 times and kept the one that maximizes the minimum robustness for samples $\theta_0 = -3\pi/4, -5\pi/8, -\pi/2$.

The initial set was chosen as $X_0^s = \{-3\pi/4, \bar{\theta}, (\bar{\theta} - 3\pi/4)/2\}$. The soft min and max function's constant was picked as $\alpha = 100$. The initial step size is set to $k = 0.05$. For $\bar{\theta} = -5\pi/8$, system trajectories starting from initial conditions $x_0 \in X_0^s$ using the initial NN weights are shown with dashed blue lines in Fig. 3.4 a). For $\bar{\theta} = -\pi/2$, they are shown with dashed blue lines as well, in Fig. 3.4 b). When $\bar{\theta} = -5\pi/8$, a set of weights that could satisfy φ in the worst case, was found using only 8 iterations of the while loop in Alg. 4. Interestingly, when the initial condition set grows larger by choosing $\bar{\theta} = -\pi/2$, the objective function (worst-case robustness value) increases in the ascent direction only with much smaller step sizes h_w, and the satisfying set of weights is found after 576 iterations of the while loop in Alg. 4. After training over the sample set X_0^s, using Alg. 5, no adversarial examples were found in the initial condition set for $\bar{\theta} = -5\pi/8$. However, for $\bar{\theta} = -\pi/2$, adversarial samples were found 2 times, and the weights were updated accordingly using

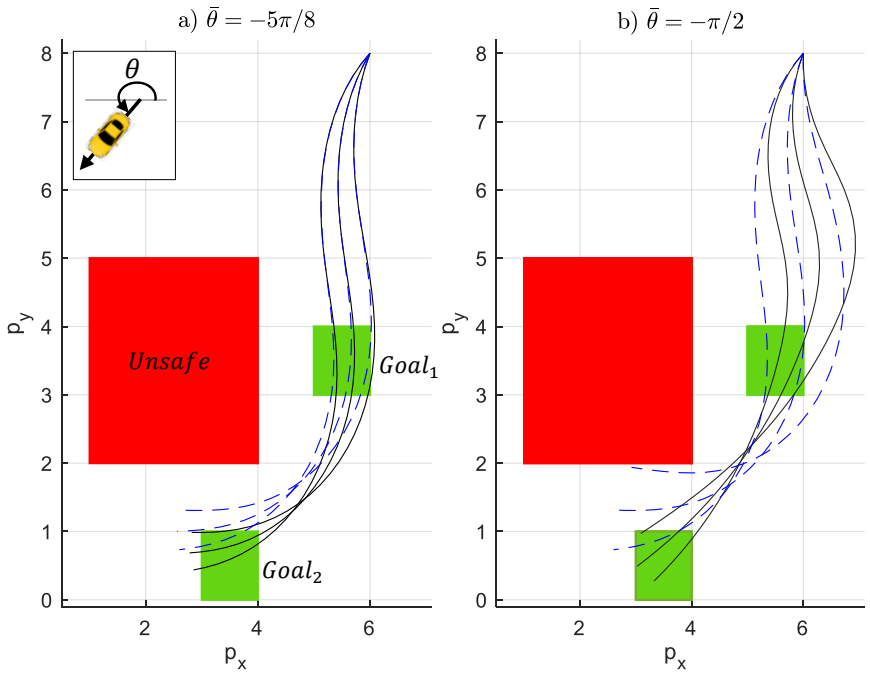


Figure 3.4: Vehicle Navigation: The Vehicle Should Visit Goals 1 and 2 in This Order While Avoiding the Unsafe Set Starting from a Set of Different Steering Angles.

Alg. 5. Satisfying trajectories starting from $x_0 \in X_0^s$ are shown with solid black lines in Fig. 3.4.

Quadrotor Mission In this case study, we consider a 6 dimensional model of a quadrotor which is affine in control, as follows [73]:

$$\dot{x} = \begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{p_z} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ g\tan(\theta) \\ -g\tan(\phi) \\ \tau - g \end{bmatrix} \tag{3.19}$$

where (p_x, p_y, p_z) and (v_x, v_y, v_z) are the quadrotor's position and velocity along x, y, z axis, θ, ϕ, τ are the control inputs (for pitch, roll and thrust), and $g = 9.81$ is the gravity. The inputs constraints are $\theta, \phi \in [-0.1, 0.1]$ and $\tau \in [7.81, 11.81]$. We assume that initially

the quadrotor is still, $(v_{x_0}, v_{y_0}, v_{z_0}) = (0, 0, 0)$, and in zero altitude $x_{z_0} = 0$. The initial x-y position of the quadrotor can vary in $(p_{x_0}, p_{y_0}) \in [0.02, 0.05] \times [0, 0.05]$. The system is simulated using a discretized step size $\Delta t = 0.05$ for N steps, where N is the STL formula horizon. The quadrotor should visit a Goal set which is blocked by a tall wall (the 'Unsafe' set) during a short amount of time while avoiding the wall. Formally, it needs to satisfy the following formula:

$$\varphi = \Box_{[1,35]} \neg \text{Unsafe} \wedge \Diamond_{[32,35]} Goal$$

where the time intervals correspond to the discrete time steps. The projection of the Unsafe and Goal sets into the quadrotor's position states are $[-\infty, 0.17] \times [0.2, 0.35] \times [0, 1.2]$ and $[0.05, 0.1] \times [0.5, 0.58] \times [0.5, 0.7]$, respectively. These sets are shown in Fig 3.5 with red and green boxes respectively. The set of initial positions is shown in grey.

We pick a NN with 6 inputs (the states of the quadrotor), 3 outputs (the control inputs of the NN), 3 hidden layers with $6, 10$, and 6 neurons. The NN has tangent Sigmoid activation functions and a scaled tangent Sigmoid in the output to enforce the control input constraints. Initially, we pick the corners and the center of the initial set to create X_0^s. In the initial training phase, the weights of the NN are picked to minimize a cost function that depends on the average distance of the quadrotor's trajectories from a motion plan that satisfies the specification. This distance is measured using a simplified dynamic time warping metric. The motion plan is a simple piece-wise constant trajectory that the quadrotor is not able to perfectly follow given its dynamics. This plan is shown in Fig. 3.5. A final cost based on whether the trajectory reaches to goal set in time or not is added to the cost function. 100 iterations of the CMA-ES with a population size of 200 were used for initial training of the neural network. This phase took about 3×10^4 seconds (about 8 hours) on our machine[3]. The trajectories of X_0^s after the initial training phase are shown in the top of Fig. 3.6. As it's clear from the figure, one of the trajectories (in purple) does not satisfy φ. As a result, we improve the set of NN weights W, using the Lagrange multiplier

[3]We did not use GPUs to reduce the training time.

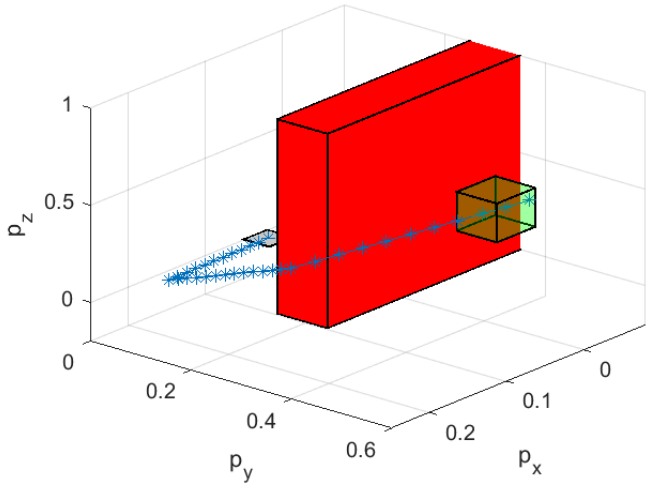


Figure 3.5: Quadrotor Needs to Avoid the Unsafe Set and Reach the Goal Set During a Short Interval of Time. The Mission Is Shown in Blue.

weight update law (see Alg. 4). We pick $\alpha = 1000$ and an initial step size of $k = 0.1$. The trajectories after this step are shown in Fig. 3.6(c). In the adversarial training phase (see Alg. 5), 3 adversarial examples were found using S-TaLiRo's "SA" optimization, they were added to the samples set X_0^s, and the NN weights were adjusted using our weight update law accordingly until no further adversary was found. The NN weight update using the Lagrange multipliers approach - on the initial set plus its integration with the adversarial samples - took about 10^3 seconds. An interesting observation was that while the NN was not designed to output discretized values, after the training phase, the weights were tuned such that the NN outputs were almost always equal to the minimum or maximum allowed values of the control inputs. In other words, the output layer activation function was always saturated. This is interesting since as described in [73], the optimal policy is a bang-bang strategy.

The results from the above Lagrange Multipliers (LM) approach were compared with the results using a pure CMA-ES method as described in the following:

1. The initial set X_0^s is selected as before.

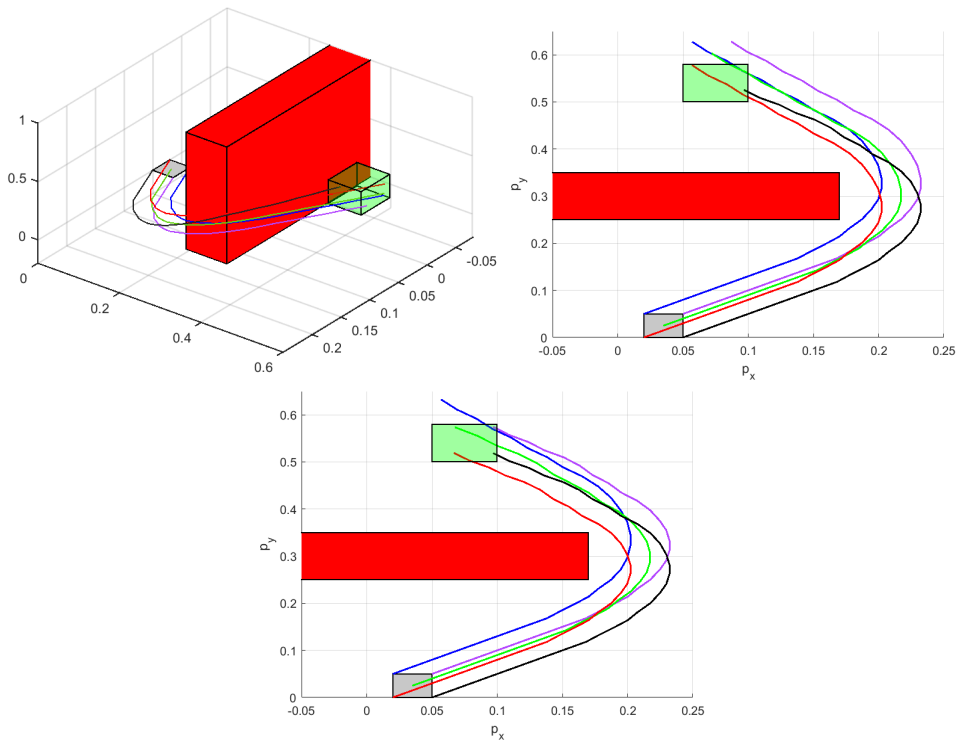


Figure 3.6: Sample Trajectories in x_0^s, Top Left: After the Global Training, Top Right: Projection of Top Left Plot into the x-y Plane Bottom: Trajectories after Improving the NN Weights Using Back-propagation

2. CMA-ES is used to maximize the minimum robustness value on X_0^s with a maximum of 100 iterations and it returns whenever the set of weights satisfies φ on X_0^s in the worst case.

3. A falsification approach is used. If an adversarial sample is found, it will be added to X_0^s, and the algorithm goes back to step (2), otherwise, the algorithm returns.

Table 3.1: Comparison of Training Using the Proposed Approach (LM) and CMA-ES

	LM	CMA-ES
time of training	$3 \times 10^4 + 10^3$	8.6×10^4
num. of falsifications found	3	6

Comparison results on the training time and the number of adversarial samples found before the algorithm returns can be found in Table 3.1. The training time for our approach is less than half the training time for the CMA-ES approach (training using the CMA-ES approach took about a day). In order to compare the quality of the resulting controller using these approaches, we finely grid the set of initial conditions and evaluated the corresponding robustness values. The resulting surfaces for the two approaches are shown in Fig. 3.7 which shows a better worst case performance using the LM approach. Note that, in both cases, a few samples with negative robustness values were found since the falsification approach that we have used here does not have coverage guarantees. These negative robustness values are much smaller in amplitude when using the controller designed by the LM approach while the training time was also much less. Based on the Goal set dimension, the upper bound to

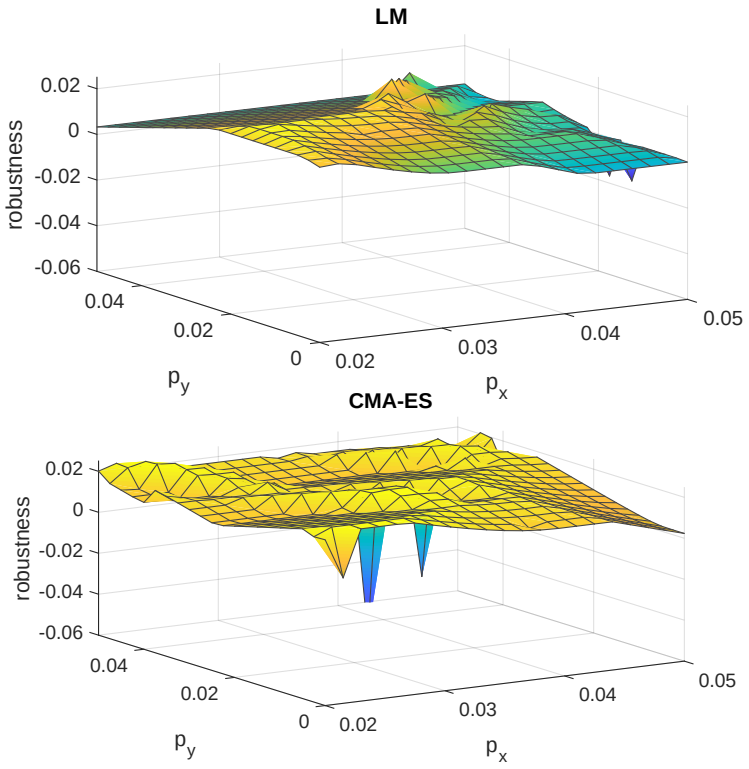


Figure 3.7: Robustness Surfaces Using the Controllers Trained with LM and CMA-ES Approach.

the robustness value is $\frac{0.1-0.05}{2} = 0.025$ which requires that the trajectory visits the center of the Goal set within the specified time interval.

3.3 Feedback Neural-network Controllers for Reach-avoid Specifications using CBFs

The results from the previous section showed training a NN controller based on increments of a loss function defined using the robustness of STL can improve the performance. However, it seems that while maximizing the STL robustness as the final reward helps to achieve the required system-level performance, it is difficult to train a robust closed-loop controller using this sole reward. Hence, in this section, we focus on a fragment of TL specification called '*Reach-Avoid*' specifications and propose using a more efficient training approach based on imitation learning. A reach-avoid specifications can be formalized as '$\Box$ safe $\wedge$ $\Diamond$ Goal' .

3.3.1 Preliminaries

Consider a nonlinear control system without disturbances and with affine control inputs:

$$\dot{x} = f(x) + g(x)u, \tag{3.20}$$

where $x \in X \subset \mathbb{R}^n$ is the system state, $u \in U \subset \mathbb{R}^l$ is the control input, and $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are locally Lipschitz functions. Given an initial condition $x(0)$, we denote the solution of the system at time t with $x(t)$. A function $\alpha : \mathbb{R} \to \mathbb{R}$ is said to be an extended class $\mathcal{K}$ function iff α is strictly increasing and $\alpha(0) = 0$ [13].

Definition 3.3.1 (Set Invariance [23])**.** *A set $C \subseteq \mathbb{R}^n$ is forward invariant w.r.t the system (3.20) iff for every $x(0) \in C$, its solution satisfies $x(t) \in C$ for all $t \geq 0$.*

Definition 3.3.2 (Barrier Function)**.** *Let $h : X \to \mathbb{R}$ be a continuously differentiable function, $C = \{x \in X | h(x) \geq 0\}$, and α be a locally Lipschitz extended class $\mathcal{K}$ function. h is a* barrier function *iff for all $x \in C$*

$$\dot{h}(x) \geq -\alpha(h(x)) \tag{3.21}$$

Lemma 3.3.1 ([60]). *If h is a barrier function for C, and α is as defined in Def. 3.3.2 then C is a forward invariant set.*

Definition 3.3.3 (**Control Barrier Function** [13]). *A continuous, differentiable function $h(x)$ is a Control Barrier Function (CBF) for the system (3.20), if there exist a class $\mathcal{K}$ function α such that for all $x \in C$:*

$$L_f h(x) + L_g h(x)u + \alpha(h(x)) \geq 0 \tag{3.22}$$

where $L_f h(x) = \frac{\partial h}{\partial x}^\top f(x), L_g h(x) = \frac{\partial h}{\partial x}^\top g(x)$ are the first order Lie derivatives of the system.

Any Lipschitz continuous controller $u \in K_{cbf}(x) = \{u \in U \mid L_f h(x) + L_g h(x)u + \alpha(h(x)) \geq 0\}$ results in a forward invariant set C for the system of Eq. (3.20).

Definition 3.3.4 (Control Input Relative Degree of a Function). *A continuously differentiable function h has a control input relative degree m w.r.t the system (3.20), if the first time that the control u appears in the derivatives of h along the system dynamics is in its m^{th} derivative.*

If the function h has a relative degree $m > 1$, $L_g h(x) = L_g^{m-1} h(x) = 0$. As a result Eq. (3.22) cannot be directly used for choosing safe controllers $u \in K_{cbf}(x)$. Motivated by the use of input-output linearization in Lyapunov functions [80], high Order Control Barrier Functions (HOCBF) were introduced in [103], and [131] to derive necessary conditions for guaranteeing set invariance in this case. Assuming that the function h has a relative degree m w.r.t the system (3.20), define the series of functions $\psi_i : \mathbb{R}^n \to \mathbb{R}, i = 0, 1, \cdots, m$ and the corresponding sets $C_1, \cdots, C_m$ as follows:

$$\begin{aligned}
&\psi_0(x) = h(x) \\
&\psi_1(x) = \dot{\psi}_0(x) + \alpha_1(\psi_0(x)) \qquad C_1 = \{x \mid \psi_0(x) \geq 0\} \\
&\psi_2(x) = \dot{\psi}_1(x) + \alpha_2(\psi_1(x)) \qquad C_2 = \{x \mid \psi_1(x) \geq 0\} \\
&\quad \vdots \qquad\qquad\qquad\qquad\qquad\quad \vdots \\
&\psi_m(x) = \dot{\psi}_{m\text{-}1}(x) + \alpha_m(\psi_{m\text{-}1}(x)) \;\; C_m = \{x \mid \psi_{m\text{-}1}(x) \geq 0\}
\end{aligned} \tag{3.23}$$

where $\alpha_1, \alpha_2 \cdots, \alpha_m$ are class $\mathcal{K}$ functions of their arguments.

Definition 3.3.5 (High Order Barrier Functions)**.** *A function $h : \mathbb{R}^n \to \mathbb{R}$ with a control input relative degree m is a High Order Barrier Function (HOBF) for system (3.20), if there exist differentiable class $\mathcal{K}$ functions $\alpha_1, \alpha_2 \cdots, \alpha_m$ such that for all $x \in C_1 \cap C_2 \cap \cdots \cap C_m$, we have: $\psi_m(x) \geq 0$. Under this condition, the set $C_1 \cap C_2 \cap \cdots \cap C_m$ is forward invariant.*

Definition 3.3.6 (High Order Control Barrier Functions [131])**.** *A function $h : \mathbb{R}^n \to \mathbb{R}$ with a relative degree m is a High Order Control Barrier Function (HOCBF) for system (3.20), if there exist differentiable class $\mathcal{K}$ functions $\alpha_1, \alpha_2 \cdots, \alpha_m$ such that for all $x \in C_1 \cap C_2 \cap \cdots \cap C_m$:*

$$\psi_m(x) = L_f^m h(x) + L_g L_f^{m-1} h(x) u + O(h(x)) + \alpha_m(\psi_{m-1}(x)) \geq 0 \tag{3.24}$$

where $O(.)$ denotes the remaining Lie derivatives along f with degree less than or equal to $m-1$.

Any Lipschitz controller $u \in K_{hocbf}(x) = \{u \in U \mid L_f^m h(x) + L_g L_f^{m-1} h(x) u + O(h(x)) + \alpha_m(\psi_{m-1}(x)) \geq 0\}$ renders the system safe, and the set $C_1 \cap C_2 \cap \cdots \cap C_m$ forward invariant.

3.3.2 Control Barrier Functions in presence of Disturbance

In this paper, the nonlinear control system (3.20) is considered in presence of disturbances as in the following:

$$\Sigma : \quad \dot{x} = f(x) + g(x)u + Mw, \quad x(0) \in X_0 \tag{3.25}$$

where x, u, f, g are defined as for the system of Eq. (3.20), X_0 is the set of initial conditions, and $w \in W \subset \mathbb{R}^l$ is the disturbance input. Each dimension of W which we denote by W_i defines an interval $[\underline{w_i}, \overline{w_i}]$ that the i^{th} element of w belongs to, M is a $n \times l$ zero-one matrix with at most one non-zero element in each row.

When a disturbance is present, in order to guarantee the forward invariance of the set C, which we call the safe set, the condition in inequality (3.21) needs to be satisfied for all $w \in W$, including its worst case where it minimizes the left-hand side of the inequality.

Definition 3.3.7. *The continuously differentiable function h with control input relative degree one is a CBF in presence of Disturbance (CBFD) for the system of Eq. (3.25), if there exist a class $\mathcal{K}$ function α such that for all $x \in C$ and $w \in W$, the following inequality is satisfied*

$$L_f h(x) + L_g h(x)u + L_M h(x)w + \alpha(h(x)) \geq 0 \tag{3.26}$$

where $L_M h(x) = \frac{\partial h}{\partial x}^\top M$. Equivalently for all $x \in C$ the following inequality needs to hold:

$$L_f h(x) + L_g h(x)u + \alpha(h(x)) \geq F^*_{L_M}(x) \tag{3.27}$$

where $F^*_{L_M}(x) = \max_{w \in W}(-L_M h(x)w)$

Definition 3.3.8 (Disturbance Relative Degree of a Function)**.** *A continuously differentiable function h has a disturbance relative degree q w.r.t the system (3.20), if the first time that the disturbance w appears in the derivatives of h along the system dynamics is in its q^{th} derivative.*

If the disturbance relative degree of h is greater than 1, $L_M h(x) = 0$ and a CBF is a CBFD too. Otherwise, since $L_M h(x)w$ is linear in w, and $w \in W$ imposes linear constraints on w, the program $\max_{w \in W}(-L_M h(x)w)$ is a linear program for each $x \in C$ whose solution can be found and replaced in inequality (3.27) to define the set of control values that satisfy the following inequality:

$$K_{cbfd}(x) = \{u \in U \mid L_f h(x) + L_g h(x)u + \alpha(h(x)) \geq F^*_{L_M}(x)\} \tag{3.28}$$

Theorem 1. *Given a CBFD h from Def. (3.3.7), any Lipschitz continuous controller $u \in K_{cbfd}(x)$ renders the set C forward invariant.*

Proof. The proof can be directly derived from Lemma 3.3.1. To be explicit, if for all $x \in C$ and $w \in W$, $\dot{h}(x) = L_f h(x) + L_g h(x)u + L_M h(x)w \geq -\alpha(h(x))$, then the solutions to system (3.25) with $x(0) \in C$, satisfy $h(x(t)) \geq 0$. So based on Def. 3.3.1, C is forward invariant. □

If the function h has a control input relative degree higher than one, the multiplier of u in Eq. (3.27), $L_g h(x)$ is equal to zero, so the choice of u will not affect the satisfaction of inequality (3.27). In the following section we will study HOCBFs in presence of disturbance.

3.3.3 High order Control Barrier Functions in Presence of Disturbance

Assume that the continously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ has control input relative degree m and consider the series of functions $\psi_i : \mathbb{R}^n \to \mathbb{R}, i = 0, ..., m$ and their corresponding sets $C_1, \ldots, C_m$ as defined in Eq. (3.23).

Definition 3.3.9. *The function h is a High Order Barrier Function in presence of disturbance (HOBFD) for system (3.25), if there exist differentiable class $\mathcal{K}$ functions $\alpha_1, \alpha_2, \ldots, \alpha_m$ that define the functions $\psi_1, \cdots, \psi_m$, such that for all $x \in C_1 \cap C_2 \cap \cdots \cap C_m$, we have: $\psi_m(x) \geq 0$*

Definition 3.3.10. *The function h is a High Order Control Barrier Function in presence of disturbance (HOCBFD) for system (3.25), if there exist differentiable class $\mathcal{K}$ functions $\alpha_1, ..., \alpha_m$ that define the functions $\psi_1, ..., \psi_m$, s.t for all $x \in C_1 \cap C_2 \cap ... \cap C_m$ and $w \in W$:*

$$\psi_m(x) = L_f^m h(x) + L_g L_f^{m-1} h(x) u + P(x, w) + O(h(x)) + \alpha_m(\psi_{m-1}(x)) \geq 0 \tag{3.29}$$

where $P(x, w)$ is a function of x and w that separates all the terms including w in $\psi_m(x)$ from the rest, $O(.)$ denotes the remaining Lie derivatives along f with degree less than or equal to $m-1$. Since equation (3.29) needs to be satisfied for all $w \in W$, we can equivalently write it as:

$$L_f^m h(x) + L_g L_f^{m-1} h(x) u + O(h(x)) + \alpha_m(\psi_{m-1}(x)) \geq F_P^*(x) \tag{3.30}$$

where $F_P^*(x) = \max_{w \in W} (-P(x, w))$

If the disturbance relative degree of h is greater than m, $P(x, w) = 0$ and any HOCBF is a HOCBFD, else if the disturbance relative degree is m, $P(x, w) = L_M L_f^{m-1} h(x) w$ and $\max_{w \in W} (-P(x, w))$ is a linear program. Otherwise $P(x, w)$ is a nonlinear function of w in

general, and the solution to the nonlinear program $F_P^*(x) = \max_{w \in W} (-P(x,w))$ can be used to find the set of control inputs that satisfy inequality (3.30):

$$K_{hocbfd}(x) = \{u \in U \mid L_f^m h(x) + L_g L_f^{m-1} h(x) u + O(h(x)) + \alpha_m(\psi_{m-1}(x)) \geq F_P^*(x)\}$$

Theorem 2. *Given a HOCBFD h from Def. (3.3.10), any Lipschitz continuous controller $u \in K_{hocbfd}(x)$ renders the set $C_1 \cap C_2 \cap \cdots \cap C_m$ forward invariant.*

Proof. Any controller $u \in K_{hocbfd}(x)$ enforces $\psi_m(x) \geq 0$ or equivalently $\dot{\psi}_{m-1}(x) \geq -\alpha_m(\psi_{m-1}(x))$ irrespective of the value of $w \in W$. Assuming that $x(0) \in C_1 \cap C_2 \cap \cdots \cap C_m$, and hence $x(0) \in C_m$, we have $\psi_{m-1}(x(0)) \geq 0$ which based on lemma 3.3.1, leads to $\psi_{m-1}(x) \geq 0$ $(x \in C_m)$ or equivalently $\dot{\psi}_{m-2}(x) \geq -\alpha_{m-1}(\psi_{m-2}(x))$, again since $x(0) \in C_{m-1}$ this results in $\psi_{m-2}(x) \geq 0$ $(x \in C_{m-1})$. Continuing this reasoning, we can prove that $C_1 \cap C_2 \cap \cdots \cap C_m$ is forward invariant. □

Remark 3. *The functions $F_{L_M}^*, F_P^*(x)$ are Lipschitz and hence, it is possible to find Lipschitz continuous controllers $u \in K_{cbfd}(x)$ or $u \in K_{hocbfd}(x)$. In the following we prove Lipschitz continuity of , $F_P^*(x)$. Lipschitz continuity of $F_{L_M}^*(x)$ will follow.*

$$\begin{aligned}
||F_P^*(x_2) - F_P^*(x_1)|| &= ||\max_w(-P(x_2,w)) - \max_w(-P(x_1,w))|| \\
&= ||\max_w(-P(x_2,w) + P(x_1,w) - P(x_1,w)) - \max_w(-P(x_1,w))|| \\
&\leq ||\max_w(P(x_1,w) - P(x_2,w)) + \max_w(-P(x_1,w)) - \max_w(-P(x_1,w))|| \\
&= ||\max_w(P(x_1,w) - P(x_2,w))|| \leq L_p||x_2 - x_1||
\end{aligned}$$

The first inequality is true since $\max(f+g)(w) \leq \max f(w) + \max g(w)$, and the second inequality is true since for all w including the one that maximizes $(P(x_1,w) - P(x_2,w))$, we have $||(P(x_2,w) - P(x_1,w))|| \leq L_p||x_2 - x_1||$ where L_p is the Lipschitz constant for P.

Remark 4. *In order to use HOCBFDs to prove that all the trajectories of the system 3.25 starting from X_0 will never exit C_1, the sets $C_1, C_2, \cdots, C_m$ should have a nonempty interior, and the set of initial conditions of the system, X_0, should be a subset of $C_1 \cap C_2 \cap \cdots \cap C_m$. Note that if $X_0 \subset C_1$ $(h(x(0)) \geq 0)$ except for special cases (see [131]) which we do not consider here, we can always choose $\alpha_1, \alpha_2 \cdots, \alpha_m$ such that $x_0 \in C_2 \cap \cdots \cap C_m$.*

Note that the problem $\max_{w \in W} (-P(x,w))$, is in general a nonlinear program and finding its optimal - or even suboptimal - solution can be time consuming. A special case of the problem is if we consider the linear class $\mathcal{K}$ functions $\alpha_1, \cdots, \alpha_{m-1}$ which will form Exponential Control Barrier Functions [13]. This makes $P(x,w)$ a polynomial function of degree at most m in w. In case of polynomial functions $\alpha_1, \cdots, \alpha_m$, $P(x,w)$ will be a polynomial function of w - potentially of higher degree than m.

A special case is when $m = 2$, and $\alpha_1, \cdots, \alpha_m$ are linear functions. In this case $P(x,w)$ is a quadratic function of w, and $\max_{w \in W} (-P(x,w))$ is a QP for each $x \in X$ that can be solved efficiently.

Example 1. *Consider the system $\dot{x}_1 = x_2 + w$, $\dot{x}_2 = u$ with $w \in [\underline{w}, \overline{w}]$. The control input should be designed such that the function $h(x) = x_1^2 - 1$ is a HOCBFD. We consider $\alpha_i(y) = y, i = 1, 2$, so we have $\alpha_i'(y) = \frac{\partial \alpha_i}{\partial y} = 1$, and as a result:*

$$\begin{aligned}\psi_2(x) &= \ddot{h}(x) + \alpha_1'(h(x))\dot{h}(x) + \alpha_2(\dot{h}(x) + \alpha_1(h(x))) \\ &= 2x_1 u + \underbrace{(4x_2 + 4x_1)w + 2w^2}_{P(x,w)} + 2x_2^2 + 4x_1 x_2 + x_1^2 - 1\end{aligned}$$

observing that $w_{opt} = \underset{\underline{w} < w < \overline{w}}{\arg\max} (-2w^2 - (4x_2 + 4x_1)w)$ is a quadratic program that can be solved at each x, any Lipschitz controller in the set $K_{hocbfd}(x) = \{2x_1 u + 2x_2^2 + 4x_1 x_2 + x_1^2 - 1 \geq -2w_{opt}^2 - (4x_2 + 4x_1)w_{opt}\}$ will make the set $C_1 \cap C_2 = \{x \mid h(x) \geq 0\} \cap \{x \mid \dot{h}(x) + h(x) \geq 0\}$ forward invariant, hence any trajectory starting from this set will never exit the set even in presence of the worst case disturbance. When $\underline{w} = -0.1, \overline{w} = 0.1$, this set is shown in Fig. 3.8.

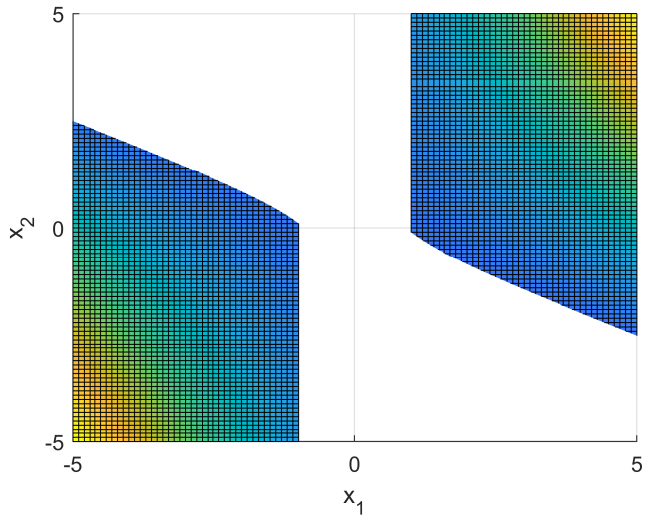


Figure 3.8: The Invariant Sets of Example 1 for the Worst-case Disturbance $w \in [-0.1, 0.1]$.

3.3.4 Control Optimization Problem with CBF constraints

In order to find safe sub-optimal controllers, many recent works [96, 131, 14, 142], formulate optimization problems with quadratic costs in the control input u subject to CLF and CBF constraints (each CBF constraints corresponds to an unsafe set) which are linear in u. These QPs are solved every time new information about the states x is received, and the resulting control value u is used in the time period before new information is received. In presence of disturbances, in order to formulate the QPs with constraints of type (3.27) or (3.30), w_{opt} should be computed as a prerequisite. To compute w_{opt} one need to solve $\max_{w \in W} (-L_M h(x) w)$ or $\max_{w \in W} (-P(x, w))$ - depending on the relative degree m - for each barrier function or unsafe set. After computing w_{opt} it can be used in the following QP to find the semi-optimal CBFD-based control input:

$$\min_{u \in U} u^T Q u$$

$$\text{s.t. Eq (8) if } m = 1 \text{ or Eq. (11) if } m > 1$$

As a result, formulating the quadratic program and solving it for evaluating the control input u may not be possible at run-time. In the following section, we present a paradigm

for training NN controllers that predict the value of the control input resulting from the quadratic programs.

3.3.5 Learning NN Controllers from Control Barrier Functions Using the DAGGER Algorithm

Imitation learning methods, which use expert demonstrations of good behavior to learn controllers, have proven to be very useful in practice [68, 4, 19, 110, 121]. While a typical method to imitation learning is to train a classifier/regressor to predict an expert's behavior given data from the encountered observations and expert's actions in them, it's been shown in [111] that using this framework, small errors made by the learner can lead to large errors over time. The reason is that in this scenario, the learner can encounter completely different observations than those it has been trained with, leading to error accumulation. Motivated by this, [111] presents an algorithm called DAGGER (Dataset Aggregation) that iteratively updates the training dataset with new observations encountered by the learner and their corresponding expert's actions and retrains the learner.

As described in Section 3.3.4, formulating and solving the required quadratic programs may not be feasible at run-time. As a result, we use an algorithm inspired by the DAGGER algorithm to train NN controllers that predict the outcome of the quadratic program. In this regard, the QP acts as an expert that a NN imitates. An NN controller that has been trained offline can be used in a feedback loop to produce the desired control values online. The NN training algorithm is described in Alg. 6 in which it is assumed that $\pi^*(x, \Sigma, U, W)$ is an expert that given the system Σ, W, and U performs the QP routine at x to output the desired control value.

Algorithm 6 Data set Aggregation for training NN using Quadratic Programs

Data: The dynamical system (3.25), the set of admissible control inputs U, the set of external inputs W, the set of initial conditions X_0, the constant $0 < p < 1$, maximum number of iterations N

Randomly choose the set X_0^s by sampling from X_0

Sample trajectories of the system (3.25) with initial conditions in X_0^s and input $\pi_0 = \pi^*(x, \Sigma, U, W)$

Initialize D with the pairs of visited states and corresponding control inputs: $D = (x, \pi^*(x, \Sigma, U, W))$

Train NN controller $\hat{\pi}_1$ on D

for $i = 1, ..., N$ **do**

$\beta = p^i$

Sample trajectories of the system (3.25) with $x(0) \in X_0^s$ and input $\pi_i = \beta\pi^*(x, \Sigma, U, W) + (1 - \beta)\hat{\pi}_i(x)$

Get dataset $D_i = (x, \pi^*(x, \Sigma, U, W))$ of visited states and corresponding control inputs

Aggregate datasets: $D \leftarrow D \cup D_i$

Train NN controller $\hat{\pi}_i$ on D

end

return the best $\hat{\pi}_i$ on validation

3.3.6 Reach Avoid Problem of a Water Vehicle Model

Consider the model of a surface water vehicle subject to wind gusts and water currents:

$$\dot{x} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} w, \qquad x(0) \in X_0 \tag{3.31}$$

where the state $x \in \mathbb{R}^3$ consists of vehicle location (x_1, x_2) and the heading angle θ. The control input $u \in \mathbb{R}$ is the vehicle's steering angle. The velocity v is assumed to be constant

($v = 1$) as it has a different relative degree from the steering angle u[4]. The external disturbance is $w \in [-0.1, 0.1]$. System trajectories starting from the set $X_0 = [8, 9] \times [5, 11] \times [-\pi, \pi]$ should avoid the unsafe sets $\mathcal{U}_i, i = 1, ..., 5$ and reach the goal set $\mathcal{G}$:

$$\mathcal{U}_i = \{x : (x_1 - p_i(1))^2 + (x_2 - p_i(2))^2 < r_i\},$$
$$\mathcal{G} = \{x : (x_1 - x_{g,1})^2 + (x_2 - x_{g,2}))^2 < 0.3\}$$

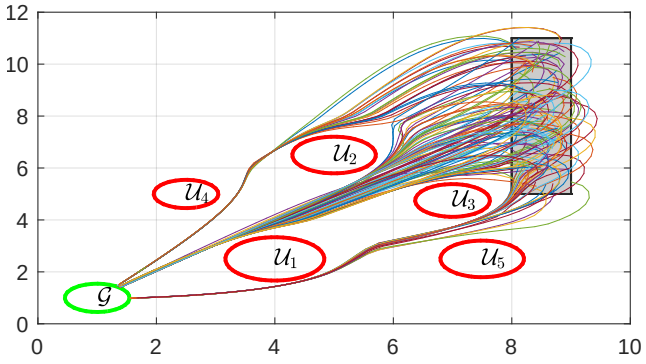
where $p_1 = (4, 2.5),\ r_1 = 0.7,\ p_2 = (5, 6.5),\ r_2 = 0.5,\ p_3 = (7, 4.75),\ r_3 = 0.4,\ p_4 = (2.5, 5),\ r_4 = 0.3,\ p_5 = (7.5, 2.5),\ r_5 = 0.5$, and $x_{g,1} = x_{g,2} = 1$.

In order to reach the goal set, instead of using CLF based constraints, we formulate the stabilizing condition in the objective function. The desired heading angle is $\theta_{ref}(x) = \arctan(\frac{x_{g,2}-x_2}{x_{g,1}-x_1})$, and the desired input u to force θ to follow θ_{ref} is $u_{ref}(x) = K(\theta_{ref}(x) - \theta)$ where K is a positive constant, here we choose $K = 1$. The barrier function corresponding to the unsafe set $\mathcal{U}_j$ is $h_j(x) = (x_1 - p_j(1))^2 + (x_2 - p_j(2))^2 - r_j$ which has relative degree 2 w.r.t to the steering angle u. We consider $\alpha_1(y) = \alpha_2(y) = 2y$. The function $\dot{\psi}_{2,j}$ corresponding to each h_j can be computed based on Eq. (3.23) using Matlab's Symbolic toolbox, for example:

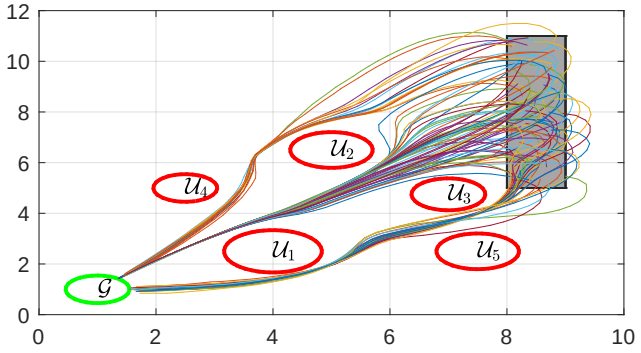
$$\begin{aligned}\dot{\psi}_{2,1} = &\underbrace{-(2\sin(\theta)(x_1 - 4) - 2\cos(\theta)(x_2 - 2.5))u}_{L_g L_f h_1(x) u} \\ &+ \underbrace{4w^2 + 4(\cos(\theta) + \sin(\theta) + 2((x_1 - 4) + (x_2 - 2.5)))w}_{P_1(x,w)} \\ &+ 4(x_1 - 4)^2 + 4(x_2 - 2.5)^2 + 4(\cos(\theta))(2x_1 - 8) + 4(\sin(\theta))(2x_2 - 5) - 0.8\end{aligned}$$

The functions $P_j(x, w)$ corresponding to each unsafe set are quadratic in w. Let's call the portion of $\dot{\psi}_{2,j}$ that only depends on x, Ψ_j. Note that $\Psi_j(x) = L_f^2 h_j(x) + O(h_j(x)) + \alpha_2(\psi_{1,j}(x))$. As a result, in order to reach the goal set while avoiding the unsafe sets, first

[4]Considering v as an input will make CBF constraints nonlinear in v, and the resulting problem will not be a quadratic program anymore. While this nonlinear program can be solved offline in this framework, in this paper we assume v is constant for simplicity.



(a)



(b)

Figure 3.9: Trajectories Initiated From x_0^s As Guided by (a) the Qps As Expert When $w = 0$ and (B) the Trained Nn Controller When Random Disturbance Is Applied to System

$F^*_{P,j}(x) = \max_{-0.1<w<0.1}(-P_j(x,w))$ needs to be computed and then the following quadratic program needs to be solved:

$$\min_{u} \quad (u - u_{ref}(x))^2 \tag{3.32}$$

$$s.t \quad L_g L_f h_j(x)u + \Psi_j(x) \geq F^*_{P,j}(x) \quad \forall j = 1, ..., 5$$

This QP is solved at each state visited by the vehicle under the controller π_i until reaching the goal set $\mathcal{G}$ as described in Alg. 6, to train NN controllers that can predict the expert's action online. Figure 3.9.(a) shows the trajectories of the system (3.31) guided by the solutions to QPs in Eq. (3.32) when $w = 0$. The NN controller successfully imitates the QPs at the 11^{th} iteration of the for loop in Alg. 6. Figure 3.9.(b) shows the system

trajectories guided by the trained NN controller when randomized disturbance is applied to the system. As it is clear from the figures the controller is robust to disturbances as it has been trained with controllers that are able to compensate for the disturbance in the worst-case. It is worth mentioning that the inputs to the NN are the location states (x_1, x_2) in addition to $(\sin(\theta), \cos(\theta))$ - instead of the state θ itself. This data processing helps remove the discontinuities that happen when mapping θ to $[-\pi, \pi]$ and helps NN understand that $-\pi$ and π are indeed equivalent. Also, even-though input constraints are not enforced in this example, they can be added to problem (3.32) as linear constraints and considered in the NN architecture by adding a saturation function in the output.

3.4 Stochastic Barrier Functions for risk bounding

In this section, we derive certificates for a stochastic system that bound the probability of a failure in a finite time and use these certificates for designing low-risk control inputs that achieve reach-avoid specifications.

3.4.1 Problem Formulation

Consider a deterministic nonlinear affine control system as described in the following ordinary differential equation

$$\dot{x}_r(t) = f_r(x_r(t)) + g_r(x_r(t))u(t), \tag{3.33}$$

where $x_r(t) \in X_r \subseteq \mathbb{R}^{n_r}$ is the system state, $u(t) \in U \subseteq \mathbb{R}^l$ is the control input, and $f_r : \mathbb{R}^{n_r} \to \mathbb{R}^{n_r}$ and $g_r : \mathbb{R}^{n_r} \to \mathbb{R}^{n_r \times l}$ are locally Lipschitz continuous functions.

Also, consider a probability space $(\Omega, \mathcal{F}, P)$, and a standard Wiener process $w(t)$ defined on this space. A stochastic system is defined using the following Stochastic Differential Equation (SDE):

$$dx_o(t) = f_o(x_o(t), t)dt + g_o(x_o(t), t)dw(t), \tag{3.34}$$

where $x_o(t) \in X_o \subseteq \mathbb{R}^{n_o}$ is a stochastic process, and f_o and g_o are locally Lipschitz continuous functions of appropriate dimensions. Since in general the process $x_o(t)$ is not guaranteed

to always lie inside the set X_o. the stopped process corresponding to $x_o(t)$ and X_o is defined as follows:

Definition 2 (Stopped Process [89]). *Assume that τ is the first time that $x_o(t)$ exits the interior of the set X_o. Then the stopped process $\tilde{x}_o(t)$ is defined as*

$$\tilde{x}_o(t) = \begin{cases} x_o(t) & \text{if} \quad t < \tau \\ x_o(\tau) & \text{if} \quad t \geq \tau \end{cases} \tag{3.35}$$

Remark 5. *The subscripts r, and o are used in the paper to indicate the quantities corresponding to the deterministic system, and the stochastic system, respectively.*

Assume that $\zeta : X_r \to \mathbb{R}_+$ is a function that defines the goal set of the system (2.1), as follows:

$$X_{\mathsf{g}} = \{x_r \in X_r \,|\zeta(x_r) \leq 0\}. \tag{3.36}$$

Also, let us define the stopped process $\tilde{x}(t) = [x_r(t), \tilde{x}_o(t)]^\top$ corresponding to the augmented state $x(t) = [x_r(t), x_o(t)]^\top$, and an unsafe region on the augmented space $X_r \times X_o$. We denote this unsafe region with $X_{\mathsf{u}} \subset X_r \times X_o$ and define it using a function $h : X_r \times X_o \to \mathbb{R}_+$ as follows:

$$X_{\mathsf{u}} =: \{\tilde{x} \in X_r \times X_o \mid h(\tilde{x}) \leq 0\}. \tag{3.37}$$

Given the state of the system at time t, $\tilde{x}(t) = [x_r(t), \tilde{x}_o(t)]^\top$, and a planning time horizon T, we define p_{u} as the probability that the process enters the unsafe set during this planning horizon, namely,

$$p_{\mathsf{u}} = P\{\tilde{x}(\tau) \in X_{\mathsf{u}} \text{ for some } t \leq \tau \leq t + T \mid \tilde{x}(t) \in X_r \times X_o\}. \tag{3.38}$$

Here, we use the term "risk" informally to refer to this event's probability (p_{u}) (see [98] for a more formal discussion about risk metrics).

A desired control input signal u steers the trajectory of the system (3.33) to X_g while bounding p_{u} for all $t \geq 0$ to a given desired threshold $\bar{p}$. Hence the problem we need to address is formalized as follows:

Problem 3.4.1. *Find a control input signal $u : \mathbb{R}_+ \to U$ for the system (3.33), s.t. 1) there exists some time $t_{\mathsf{g}} > 0$ for which $\zeta(x_r(t_{\mathsf{g}})) \leq 0$, and 2) at any time t which satisfies $0 \leq t \leq t_g$, given $x_r(t)$ of system (3.33) and $x_o(t)$ of system (3.34), the control input $u(t)$ bounds the risk p_{u} by the desired upper threshold $\bar{p}$, i.e., $p_{\mathsf{u}} \leq \bar{p}$.*

3.4.2 Stochastic Control Barrier Functions

In this section, we first review some background information about stochastic systems and processes, and then derive conditions on a BF candidate to bound the risk.

The evolution of a function of a deterministic system's state can be characterized using Lie derivatives. The stochastic analog of the Lie derivatives are infinitesimal generators that characterize the evolution of the expectation of functions of the stochastic system's state $x_o(t)$ [107]:

Definition 3 (Infinitesimal generator)**.** *The infinitesimal generator A of a stochastic process $x_o(t)$ on $\mathbb{R}^{n_o}$ is defined by*

$$AB(x_0) = \lim_{t \to 0} \frac{E[B(x_o(t)) \mid x_o(0)=x_0] - B(x_0)}{t},$$

for all the functions $B : \mathbb{R}^{n_o} \to \mathbb{R}$ for which the above limit exists for all x_0 [104].

Let $x_o(t)$ be a stochastic process satisfying Eq. (3.34). The generator A of a twice differentiable function $B : \mathbb{R}^{n_o} \to \mathbb{R}$ is given by [104]

$$AB(x_o) = \frac{\partial B}{\partial x_o} f_o(x_o, t) + \frac{1}{2} tr\Big(g_o(x_o, t)^\top \frac{\partial^2 B}{\partial {x_o}^2} g_o(x_o, t) \Big).$$

where $tr(.)$ computes the trace of a square matrix.

The stopped process $\tilde{x}_o(t)$ in Eq. (3.35) inherits the right continuity and strong Markovian property of $x_o(t)$. It also shares the same infinitesimal generator corresponding to $x_o(t)$ on X_o.

3.4.3 Bounded Risk Using Stochastic Control Barrier Functions

In the following, we derive the conditions on the control input s.t. the risk is bounded from above by the desired upper threshold. We build upon the idea of [107] to define a BF whose level set of value one contains X_{u} so that the evolution of the BF's expected value can be used to compute upper bounds on p_{u}. These bounds have been proved to exist in [89] and are collected in [116] for finite-time stochastic system verification and feedback control design. We use the computed upper bounds to establish conditions on the evolution of the BF s.t. p_{u} is bounded to $\bar{p}$, and choose control actions according to these conditions in real-time.

Definition 4. *A twice differentiable function $B : X_r \times X_o \to \mathbb{R}_+$ is a Barrier Function (BF) candidate w.r.t the sets X_r, X_o, X_{u}, if*

$$B(x) \geq 0 \quad \forall x \in X_r \times X_o, \text{ and} \tag{3.39}$$

$$B(x) \geq 1 \quad \forall x \in X_{\mathsf{u}}. \tag{3.40}$$

Example 2. *If h is a differentiable function, $B(x) = e^{-\gamma h(x)}$ is a BF candidate for $\gamma > 0$ w.r.t X_{u} as defined in (3.37).*

In what follows we assume that solutions to Eq. (3.33) are guaranteed to exist until at least t_g. As an example, a locally Lipschitz continuous state feedback control $u(t) = u(x)$ or a piecewise continuous time-varying control $u(t)$ can guarantee the existence of solutions to Eq (3.33) [80].

Definition 5. *Consider the system of Eq. (3.33) with a control input $u(t) = u(x)$, augmented with the stochastic system in Eq. (3.34). A BF candidate B is a Stochastic Barrier Function (SBF) for this augmented system, if there exist $a \geq 0, b \geq 0$ s.t. the following condition on the infinitesimal generator of B is satisfied $\forall x \in X_r \times X_o$,*

$$\frac{\partial B}{\partial x} F_{cl}(x) + \frac{1}{2} tr(g_o(x_o, t)^\top \tfrac{\partial^2 B}{\partial {x_o}^2} g_o(x_o, t)) \leq -aB(x) + b,$$

where $F_{cl}(x) = [f_r(x_r) + g_r(x_r)u(x)), f_o(x_o, t)]^\top$.

Given the current state of the defined augmented system $x(t)$, an SBF provides a bound on p_{u} (probability of entering the unsafe set during the planning horizon T):

Theorem 3. *Consider the stopped process $\tilde{x}(t)$ w.r.t the augmented system state $x(t)$, define $B_0 = B(x(t))$, and $p_B = P\{\sup_{t\leq\tau\leq t+T} B(\tilde{x}(\tau)) \geq 1 \mid \tilde{x}(t) \in X_r \times X_o\}$. Then:*

$$\textit{If } a = 0: \quad p_{\mathsf{u}} \leq p_B \leq B_0 + bT. \tag{3.41}$$

$$\textit{If } a > 0,\ b \leq a: \quad p_{\mathsf{u}} \leq p_B \leq 1 - (1 - B_0)e^{-bT}. \tag{3.42}$$

$$\textit{If } a > 0,\ a \leq b: \quad p_{\mathsf{u}} \leq p_B \leq \frac{B_0 + (e^{bT} - 1)\frac{b}{a}}{e^{bT}}. \tag{3.43}$$

Proof. The bounds are immediate corollaries of [89, Ch. 3, Thrm. 1, and Cor. 1-1]. □

Remark 6. *Note that for $B(\tilde{x}(t)) = B_0 \neq 0$, and $T = 0$, the right hand side of inequalities in Theorem. 3 do not reduce to zero. The reason is that the method of proof for finding the bounds in [89] does not distinguish between the fixed/deterministic initial conditions for $B(\tilde{x})$ and initial conditions which are random variables with expected value $B(\tilde{x}(t))$. Hence, the results remain valid for the case of nonanticipative initial conditions with mean $B(\tilde{x}(t))$.*

The definition of an SBF is more suitable for verifying stochastic safety properties of a system with a closed-form state feedback control $u(x)$. When solving a control synthesis problem, additional conditions on a BF candidate should depend on the choice of the control input:

Definition 6. *A BF candidate B is a Stochastic Control Barrier Function (SCBF) for the augmented system of Eq. (3.33), and (3.34), if there exist a control input $u \in U$ s.t. for all $x \in X_r \times X_o$ the following condition is satisfied for some $a \geq 0, b \geq 0$.*

$$\frac{\partial B}{\partial x}(F_{ol}(x) + e_r g_r(x_r)u) + \frac{1}{2}tr(g_o(x_o,t)^\top \frac{\partial^2 B}{\partial {x_o}^2} g_o(x_o,t)) \leq -aB(x) + b, \tag{3.44}$$

where $F_{ol}(x) = [f_r(x_r), f_o(x_o,t)]^\top$, and $e_r = [I_{n_r}, 0_{n_r \times n_o}]^\top$, in which I_n is an $n \times n$ identity matrix, and $0_{n\times m}$ is an $n \times m$ zero matrix.

While Theorem. 3 provides us with bounds on the risk as functions of a, b, B_0, T, we still need to provide conditions on a, b (B_0 given $\tilde{x}(t)$, and T are fixed) and the control input u

to guarantee that the risk is always bounded by $\bar{p}$. We derive these conditions using SCBFs below:

Theorem 4. *Suppose that there exists a SCBF B for the augmented system of Eq. (3.33), and (3.34). If at each visited state $\tilde{x}(t)$, the control input $u \in U$ satisfies the conditions of Def. 6 for some $a \geq 0, b \geq 0$ s.t. one of the conditions*

$$a = 0, b \leq (\bar{p} - B_0)/T, \tag{3.45}$$

$$a > 0, b \leq \min(a, -\tfrac{1}{T} \ln \tfrac{1-\bar{p}}{1-B_0}), \text{ or} \tag{3.46}$$

$$a > 0, \tfrac{b(e^{bT}-1)}{\bar{p}e^{bT}-B_0} \leq a \leq b \tag{3.47}$$

hold, then for all $t \geq 0$, $p_{\mathsf{u}} \leq \bar{p}$ holds.

Proof. Based on the assumptions, the function B becomes a SBF for the system in Eq. (3.33) in closed loop with a control input u that satisfies the conditions of Thrm. (4), hence the bounds in (3.41)-(3.43) are valid. Since the extra conditions on a, b in inequalities (3.45)-(3.46) based on which the control is chosen bound the right hand sides of (3.41)-(3.43) to $\bar{p}$, we have $p_{\mathsf{u}} \leq \bar{p}$, and the proof is complete. □

3.4.4 Risk Bounded Optimization-Based Control Design

In this section, we use the properties of SCBFs for synthesizing risk-based control inputs. We use the constraint in Eq. (3.44) combined with the constraints on a, b in Eq. (3.41), (3.42), or (3.43) in an optimization problem with a quadratic cost in u to find an optimal control input that bounds the risk. Such an optimization problem has an objective function:

$$J(u) = (u - u_d)^T Q (u - u_d), \tag{3.48}$$

where u_d is a desired value which is set to zero if input minimization is desired, and Q is a diagonal matrix with non-negative elements. Hence, given $\bar{p}$, the following optimization problem can be solved each time new information about the states x_r, x_o is received, and the obtained control value u^* can be used to bound the risk until new information is received

and a new control value is computed.

$$\min_{u \in U, a, b} J(u) \tag{3.49}$$
$$\text{s.t.} \begin{cases} \text{Ineq. (3.44)} \\ \\ \text{Ineq. (3.45), or (3.46), or (3.47)} \end{cases}.$$

Note that in the above program the objective function and the first constraint are respectively quadratic and linear in the search parameters u, a, b. However, the second constraint imposed by Eq. (3.46), or (3.47) are nonlinear in either a or b. Hence, to transform the program (3.49) to a quadratic program (QP) for which efficient solvers exist, one can fix a in (3.45) or b in (3.46) to positive values, and find the other parameter to satisfy the second constraint along with an optimal control u that bounds the risk to $\bar{p}$. Also, in order to minimize the risk when possible, parameters a and b can be included in the objective function with negative and positive multipliers respectively (note the inverse relationship of the upper bound in Eq. (3.43) with a and the direct relationship of the upper bounds in equations (3.41)-(3.43) with b).

3.4.5 Goal Set Reachability

Recall that a solution to Prob. 3.4.1 should lead the states of the system (3.33) to a goal set while bounding the risk. An advantage of using BF methods for safety is that they can be combined with methods that seek other objectives like reachability. In this section, we derive the conditions under which the control input of the system (3.33) lead the states x_r to a goal set $X_{\mathbf{g}}$ as in Eq. (3.36).

Definition 7 (Control Lyapunov like Function (CLF))**.** *A differentiable function* $V : X_r \rightarrow$

$\mathbb{R}$ *is a Control Lyapunov like*[5] *function (CLF), if it satisfies the following conditions*

$$V(x_r) > 0 \quad \forall x_r \in X_r / X_g, \tag{3.50}$$

$$V(x_r) \leq 0 \quad \forall x_r \in X_g, \tag{3.51}$$

$$\forall x_r \in X_r,\ \exists u \in U \ \ s.t.\ \ \tfrac{\partial V}{\partial x_r}(f_r(x_r) + g_r(x_r)u) \leq 0. \tag{3.52}$$

If the control input u satisfies Ineq. (3.52) for all $x_r \in X_r$, then $V(x)$ decreases in value until eventually $V(x) \leq 0$ and hence the goal set X_{g} is reached. Hence, the reachability objective can be unified with safety objectives by considering the program (3.49) with an additional constraint imposed by a CLF $V(x)$ as defined in Def. 7, as follows

$$\min_{u \in U, a, b, \delta} J(u) + k\delta \tag{3.53}$$

$$\text{s.t.} \begin{cases} \text{Ineq. (3.44)} \\ \text{Ineq. (3.45), or (3.46), or (3.47)} \\ \frac{\partial V}{\partial x_r}(f_r(x_r) + g_r(x_r)u) \leq \delta \end{cases} .$$

where k is a positive constant. A candidate for the function $V(x)$ when ζ is a differentiable function is $V(x) = \zeta(x)$ (see the definition of X_{g} in Eq. (3.36)).

As in [14], in the above program, the CLF constraint (3.52) is relaxed through δ. By adding δ to the objective function, we allow for control inputs that minimally violate the Lyapunov constraint (3.52) when instantaneous improvement toward the goal set contradicts safety conditions. Since δ is considered in the objective function when safety and reachability constraints do not conflict, they will be satisfied at the same time and Prob. 3.4.1 can be solved by iteratively solving the program (3.53).

Program (3.53) can also be transformed into a QP by fixing either a or b. Also, these parameters can be included in the objective function to scale down the risk when possible, and to increase the chance of finding an admissible control u that satisfies the constraints at a later time. Whereas, if the inequality constraints are always satisfied with equality, the

[5]Despite conventional Lyapunov functions the positive definiteness of $V(x)$ is not neccessary since reachability (and not stability) is the objective.

chance of not finding an effective reaction to the stochastic process x_o through an admissible control $u \in U$ increases in the next iterations. But also note that the hard constraint on a or b prevents choosing riskier actions that possibly decrease the total objective function by an immediate movement toward the goal set or by reducing the control cost $J(u)$.

3.4.6 Control Design in the Presence of Multiple Unsafe regions

When designing a real-time control input for the system (3.33) by iteratively solving an optimization problem, it may be required to consider safety w.r.t a varying number of stochastic processes at each iteration. Furthermore, different bounds may be needed on their associated risks. For instance, the system (3.33) can describe the model of a vehicle that needs to be driven/controlled in presence of a varying number of other vehicles with stochastic characteristics, and the risk bounds corresponding to larger, or emergency agent vehicles may need to be set to smaller values too. In this case, the control action should satisfy a varying number of safety constraints related to the agents.

Assume that in an specific iteration, M stochastic processes $x_{o,i}, i = 1, 2, ..., M$ defined using SDEs of the form (3.34) - with $f_o = f_{o,i}$, $g_o = g_{o,i}$, $w = w_i$ - need to be considered in the control design. We denote the corresponding unsafe sets with $X_{\mathsf{u},i}$s. Each of these sets is defined on the space of the augmented state $\tilde{x}_i = [x_r, \tilde{x}_{o,i}]^\top$ using a relation over a function $h_i(\tilde{x}_i)$, like in Eq. (3.37). In order to bound $p_{\mathsf{u},1}, ..., p_{\mathsf{u},M}$ (the probabilities of entering the unsafe sets $X_{\mathsf{u},i}$ within a given time-horizon) to $\bar{p}_1, ..., \bar{p}_M$, we need to consider M BF candidates B_i for each unsafe set $X_{\mathsf{u},i}$ based on the Def. 4. Note that one can consider a smaller value for $\bar{p}_i$, if entering the unsafe set $X_{\mathsf{u},i}$ has a more severe impact on the system. The BFs then can be used to find a series of M conditions formed based on Thrm. 4. These conditions can be added to program (3.53) to find a sub-optimal control input that bounds the risks $p_{\mathsf{u},i}$ while reaching the goal set. Note that even with our framework that allows for designing less conservative controllers, when multiple safety constraints are present or $U \neq \mathbb{R}^l$, the feasibility of the program (3.53) cannot be assured. Such a framework needs to be considered as part of a larger architecture wherein a backup controller is implemented if

no feasible solution to the program (3.53) is found.

3.4.7 Application to Nonholonomic Systems

The deterministic system (3.33) can describe a unicycle model that can be considered as a simplified model of an AMS, i.e $x_r = [\mathsf{p}_r^x, \mathsf{p}_r^y, \theta_r]^\top$, $u = [u_1, u_2]^\top$, where $\mathsf{p}_r^x, \mathsf{p}_r^y, \theta_r$ describe the x and y position of the robot and its heading angle respectively, and u_1, u_2 are the linear and angular velocities of the robot. Also $f_r = [0, 0, 0]^\top$, and

$$\dot{x}_r(t) = g_r(x_r(t))u(t) = \begin{bmatrix} \cos(\theta_r) & 0 \\ \sin(\theta_r) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{3.54}$$

In this case, the projection of X_r into its first and second dimensions represents the AMS's workspace, i.e, the environment in which it is moving, and its projection into its third dimension is $[-\pi, \pi]$. The goal set of the AMS can describe a set of position states in $\mathbb{R}^2$:

$$X_{\mathsf{g}} =: \{x_r \in X_r \mid ([\mathsf{p}_r^x, \mathsf{p}_r^y] - x_g)^2 - r_g^2 \leq 0\}, \tag{3.55}$$

where x_g is the center and r_g is radius of the goal set.

There are M agents around the AMS whose stochastic behavior can be modelled using SDEs of the form (3.34). Assuming that each agent $i \in 1, ..., M$ is moving in direct line with slope γ_i, it can be modelled as

$$dx_{o,i}(t) = [v_{c,i} \quad \gamma_i v_{c,i}]^\top dt + c_i [1 \quad \gamma_i]^\top dw_i, \tag{3.56}$$

where c_i is a constant, $x_{o,i} = [\mathsf{p}_{o,i}^x, \mathsf{p}_{o,i}^y]^\top$ is the agent i's position, $v_i = v_{c,i} + w_i$ is its velocity where $v_{c,i}$ is a constant value and w_i is a stochastic Wiener process representing the stochastic changes in agent i's velocity.

AMS's collisions with moving agents are undesirable, hence one can define the unsafe sets as

$$X_{u,i} =: \{\tilde{x}_i \in X_r \times X_{o,i} \mid ([\mathsf{p}_r^x, \mathsf{p}_r^y] - [\mathsf{p}_{o,i}^x, \mathsf{p}_{o,i}^y])^2 - r_i^2 \leq 0\}, \tag{3.57}$$

where $\tilde{x}_i = [x_r, \tilde{x}_{o,i}]$ is the stopped process corresponding to the augmentation of the AMS's state and the obstacle i's state, $x_i = [x_r, x_{o,i}]$, and r_i depends on the width/length/radius

of the AMS and the agent i.

To find a control input u that leads the AMS to the goal set while bounding the probabilities of collisions with agents $1, ..., M$ in finite-time T to $\bar{p}_1, ..., \bar{p}_M$, program (3.53) with conditions based on BFs w.r.t $X_{u,i}$s can be solved.

3.4.8 SCBFs and CLFs for Nonholonomic Systems

For the non-holonomic system of Eq. (3.54), with an unsafe set of the form (3.57) which represents a set of position states of the system, the control inputs u_1 and u_2 (the linear and angular velocities) have different relative degrees w.r.t the BF candidate $B_i(x_i) = e^{-\gamma_i h_i(x_i)}$. The consequence is that while u_1 appears on the right-hand side of the Ineq. (3.44), u_2 does not. Hence, u_2 cannot be derived accordingly to help render the system risk-bounded. So the constraint in Thrm. 4 may not be satisfied if $\frac{\partial B_i}{\partial x_i} e_r g_r(x_r) u = \frac{\partial B_i}{\partial \mathsf{p}_r^x} cos(\theta) + \frac{\partial B_i}{\partial \mathsf{p}_r^y} sin(\theta)$ is a zero vector, or if no admissible velocity u_1 in the corresponding bounds in U can satisfy the constraint in Thrm. 4. As in [97], to avoid involved control design methods, we use a near-identity diffeomorphism to solve the problem for a closely related system. Consider $\bar{x}_r = [\bar{\mathsf{p}}_r^x, \bar{\mathsf{p}}_r^y, \bar{\theta}_r]$, and the transformation $\bar{x}_r := x_r + l[R(\theta_r)e_1, 0]^\top$, where $l > 0$ is a small constant that allows for approximating x_r with $\bar{x}_r$ with the needed precision, $R(\theta_r) = \begin{bmatrix} \cos(\theta_r) & -l\sin(\theta_r) \\ \sin(\theta_r) & l\cos(\theta_r) \end{bmatrix}$, and $e_1 = [1, 0]^\top$. Hence:

$$\dot{\bar{x}}_r(t) = \begin{bmatrix} \cos(\theta_r) & -l\sin(\theta_r) \\ \sin(\theta_r) & l\cos(\theta_r) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{3.58}$$

In the full rank system of Eq (3.58) both u_1, u_2 appear in Ineq. (3.44), and they can both contribute to satisfaction of the condition. Note that the maximum distance of x_r from $\bar{x}_r$ is l. Hence, defining $\bar{x}_i = (\bar{x}_r, \tilde{x}_{o,i})$, the unsafe sets can be expanded to account for the introduced error as

$$\bar{X}_{u,i} =: \{\bar{x}_i \mid ([\bar{\mathsf{p}}_r^x, \bar{\mathsf{p}}_r^y] - [\mathsf{p}_{o,i}^x, \mathsf{p}_{o,i}^y])^2 - (r_i + l)^2 \leq 0\}. \tag{3.59}$$

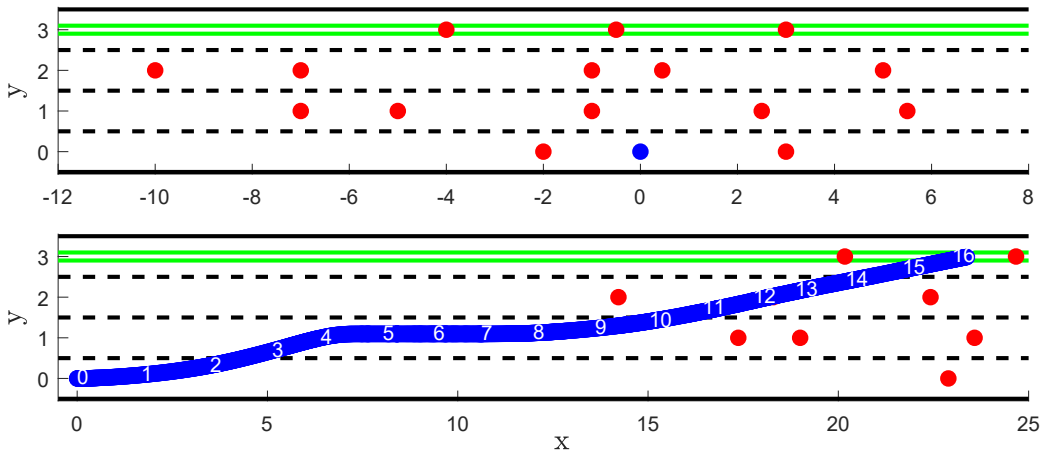


Figure 3.10: Top: Initial Positions of the Ego Vehicle and Traffic Participants. Bottom: Final Position of the Traffic Participants Alongside With the Time-stamped Trajectory of the Ego Vehicle From Start to Finish. Simulation Video Can Be Found At https://youtu.be/hqGe8h1erzA.

3.4.9 Experimental Results

In this section, we illustrate our method on a reach-avoid problem in a highway scenario. An ego vehicle in the rightmost lane needs to reach the left-most lane while avoiding collisions with other traffic participants. The ego vehicle is modelled using the unicycle model of Eq. (3.54) with the initial condition $x_r(0) = [0, 0, 0]^\top$. The linear and angular velocities of the vehicle are considered to be in the set $u \in U = \{0 \leq u_1 \leq 2,\ -\pi/6 \leq u_2 \leq \pi/6\}$. It is assumed that the traffic participants move in their lanes with stochastic velocities close to the highway's desired speed, and, hence, they are modelled using the SDE in (3.56) with $v_{c,i} = 1.5, c_i = 0.2$, and $\gamma_i = 0$. We consider a scenario with 15 traffic participants with different initial states $x_{o,i}(0)$. The top subplot in Fig. 3.10 shows the position of the ego vehicle (in blue) and the traffic participants (in red). The goal set (shown in green) is $X_{\mathsf{g}} = \{x_r | (\mathsf{p}_r^y - 3)^2 \leq 0.1^2\}$.

We define the sets $X_{\mathsf{u},i},\ i = 1, ..., 15$ that include the augmented state of the ego car

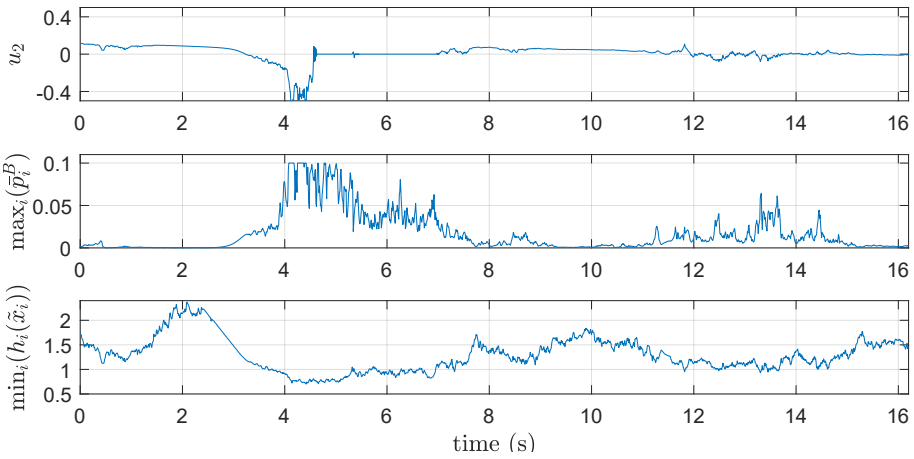


Figure 3.11: Figures Show Control Input Signals, Maximum Upper Bound to p_i^b over All the Traffic Participants, Minimum $h_i(X)$ over All the Traffic Participants Respectively.

and the traffic participant i, $\tilde{x}_i$, as in Eq. (3.57) with $r_i = 0.5$, i.e:

$$X_{\mathsf{u},i} = \{\tilde{x}_i \in X_r \times X_{o,i} \mid ([\mathsf{p}_r^x, \mathsf{p}_r^y] - [\mathsf{p}_{o,i}^x, \mathsf{p}_{o,i}^y])^2 - 0.5^2 \leq 0\}.$$

At time t, the control input $u(t)$ should be designed to bound the risks of entering the sets $X_{\mathsf{u},i}$ within a 1-second time horizon ($T = 1$) by $\bar{p}_i = 0.1$ for all $i = 1, ..., 15$. In order to find such a control input using BFs of the form $B_i(x_i) = e^{-\gamma_i h_i(x_i)}$, we transform the model of the ego vehicle using Eq. (3.58) with $l = 0.01$. In order to compensate for the transformation error, we define new unsafe sets as in Eq. (3.59). Hence, we define $B_i(\bar{x}_i) = e^{-\gamma_i h_i(\bar{x}_i)}$ with $\gamma_i = 5$ and $h_i(\bar{x}_i) = ([\bar{\mathsf{p}}_r^x, \bar{\mathsf{p}}_r^y] - [\mathsf{p}_{o,i}^x, \mathsf{p}_{o,i}^y])^2 - (0.5 + l)^2$.

At each state $\bar{x}_r$ in order to find a sub-optimal control input u that guides the ego vehicle to reach the goal set in the top lane, while bounding the risk to $\bar{p}_i = 0.1$, we formulate the quadratic program (3.53), by fixing $a_i = 1$, and using constraints from Ineq. (3.46), and $V(\bar{x}_r) = (\bar{\mathsf{p}}_r^y - 3)^2 - (0.1 + l)^2$. To guide the program to minimize the risk when possible and avoid a risky action when it is not necessary, we will add the variables b_i to the objective function. Also, at each state $\bar{x}_r$, to improve the efficiency, we only consider the SCBF constraints related to the traffic participants that are at a distance of 3 or less of the ego vehicle. Finally, we add an additional soft constraint to the QP to encourage smaller input changes from iteration to iteration. The bottom subplot in Fig. 3.10, shows

the resulting - time-stamped - trajectory of the ego vehicle and the final positions of the traffic participants. The computed control inputs u_1 and u_2 are admissible and lie in the corresponding bounds of U. The control input related to the angular velocity (u_2) is shown in the top subplot in Fig. 3.11. The middle subplot shows the maximum $\bar{p}_i^B$ over all the traffic participants, where $\bar{p}_i^B = 1-(1-B_0)e^{-b_i T}$ is the upper bound to the risk p_i^u (see Eq. (3.42)). The bottom subplot shows the minimum $h_i(\bar{x}_i)$ over all the traffic participants. From the figures, we can see that $\bar{p}_i^B$, and, hence the chance of an upcoming collision in the next 1 second is bounded to 0.1. The correlation between the bottom two subplots shows that as expected, the risk increases as the distance between the ego vehicle and some traffic participant ($\min_i(h_i(\bar{x}_i))$) is about to decrease. Another observation is that when risk increases and approaching the goal set conflicts with risk-related constraints, the computed angular velocity (u_2) steers the ego vehicle away from the traffic participants in the opposite direction of the goal set. So, when necessary reachability objective is postponed until it can be satisfied with the safety constraints at the same time. Note that our experiments show that the task cannot be completed if the risk is not tolerated and the input u is expected to make the probability of the undesired event zero as required in previous works like [28, Cor. 1].

3.5 Conclusion and Future Work

In this chapter, we proposed different methods for control synthesis w.r.t requirements of interest. The proposed methods consider the need for the implementation of efficient controllers for real-time computations, and non-conservative low-risk decisions. We started by focusing on general TL formulas, and later focused on the fragment of reach-avoid specifications that can be specified as '$\Box$ safe $\wedge$ $\Diamond$ Goal' that arise in many applications to enable more efficient design strategies:

In Section 3.2, given a system model, we proposed a systematic approach to training neural network controllers that satisfy system properties given in STL. The loss functions for training the NNs are inspired by the robustness of the STL formula which is defined

over temporal sequences corresponding to the closed-loop system response. We provided a formulation for gradient-based training which solves this issue. Furthermore, after training the NN with random samples, we iteratively search for adversarial samples to the STL property, add them to the sample set and retrain the NN. We demonstrate our approach on a 6 dimensional model of a quadrotor that needs to accomplish a mission specified in an STL formula.

While we used state feedback neural network controllers, a controller might require more information than current states to satisfy general STL properties. Specifying features that are required for satisfying different STL properties will be investigated in future work. Dynamical NN controllers for satisfying STL formulas will also be studied.

In Section 3.3, we studied Control Barrier Functions (CBF) in presence of disturbances. These functions define constraints on the control input that can be used in an optimization problem to find safe sub-optimal control inputs that enforce reach-avoid specifications. As solving these optimization problems might not be possible in real-time, we presented a framework to train NN controllers that can be used online to predict the outcome of the optimization problems. Future work will use methods like [45] to establish the safety of the learned controller and counter-example generation methods as in [138] to speed up training. Control barrier functions that enforce STL satisfaction have also been explored recently in [96]. Hence, future work can also include training NNs using imitation learning based on QPs that enforce STL satisfaction using these CBFs.

While in sections 3.2, and 3.3 the NNs are trained to satisfy the TL or reach-avoid specification, there is no guarantee that they satisfy these requirements. Hence, future work can also work on methods for designing verified NN controllers. One possible approach is to consider CBF constraints during the training procedure and enforce their satisfaction when choosing the weights of the NN.

Section 3.4 presented the conditions under which the system's probability of failure in a finite time becomes bounded to desired thresholds. These conditions depend on BF candi-

dates that contain unsafe operating conditions and constrain the growth of their expected value to bound the probability of failure. These constraints combined with constraints based on Lyapunov functions are used in a QP to design less conservative low-risk control inputs that stabilize the system or lead the system to a set of goal states, online. Our case study uses the proposed constrained QP to successfully drive a vehicle that needs to change lanes on a crowded highway while bounding the risk of collisions. In the future, we consider using the work in [75] to modify our proposed QPs to achieve a guaranteed asymptotic convergence rate. A comparison between our approach with other notions of risk, like the Conditional Value at Risk (CVaR) [6] for planning can be beneficial, too. Furthermore, we may be able to improve the bounds derived in our work using the results of [123] for exponential BFs. Finally, motivated by their application, another line of future work can include designing risk-bounded controllers based on stochastic barrier functions for stochastic systems whose states need to be estimated based on noisy measurements.

REFERENCES

[1] Houssam Abbas and Georgios Fainekos. Computing descent direction of mtl robustness for non-linear systems. In *2013 American Control Conference*, pages 4405–4410. IEEE, 2013.

[2] Houssam Abbas, Matthew O'Kelly, Alena Rodionova, and Rahul Mangharam. Safe at any speed: A simulation-based test harness for autonomous vehicles. 2017.

[3] Houssam Abbas, Andrew Winn, Georgios Fainekos, and A Agung Julius. Functional gradient descent method for metric temporal logic specifications. In *American Control Conference (ACC), 2014*, pages 2312–2317. IEEE, 2014.

[4] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[5] Arvind Adimoolam, Thao Dang, Alexandre Donzé, James Kapinski, and Xiaoqing Jin. Classification and coverage-based falsification for embedded control systems. In *International Conference on Computer Aided Verification*, pages 483–503. Springer, 2017.

[6] Mohamadreza Ahmadi, Masahiro Ono, Michel D Ingham, Richard M Murray, and Aaron D Ames. Risk-averse planning under uncertainty. In *2020 American Control Conference (ACC)*, pages 3305–3312. IEEE, 2020.

[7] Takumi Akazaki, Shuang Liu, Yoriyuki Yamagata, Yihai Duan, and Jianye Hao. Falsification of cyber-physical systems using deep reinforcement learning. In *International Symposium on Formal Methods*, pages 456–465. Springer, 2018.

[8] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[9] Matthias Althoff, Olaf Stursberg, and Martin Buss. Stochastic reachable sets of interacting traffic participants. In *2008 IEEE Intelligent Vehicles Symposium*, pages 1086–1092. IEEE, 2008.

[10] Rajeev Alur. *Principles of Cyber-Physical Systems*. MIT Press, 2015.

[11] Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*, pages 209–229. Springer, 1993.

[12] Rajeev Alur, Thomas A Henzinger, and Eduardo D Sontag. *Hybrid systems III: verification and control*, volume 3. Springer Science & Business Media, 1996.

[13] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. *European Control Conference (ECC)*, 2019.

[14] Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, pages 6271–6278. IEEE, 2014.

[15] Yashwanth Singh Rahul Annapureddy and Georgios E Fainekos. Ant colonies for temporal logic falsification of hybrid systems. In *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*, pages 91–96. IEEE, 2010.

[16] Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer, 2011.

[17] Panos J Antsaklis, James A Stiver, and Michael Lemmon. Hybrid system modeling and autonomous control systems. In *Hybrid systems*, pages 366–392. Springer, 1992.

[18] Saeed Asadi Bagloee, Madjid Tavana, Mohsen Asadi, and Tracey Oliver. Autonomous vehicles: challenges, opportunities, and future implications for transportation policies. *Journal of modern transportation*, 24(4):284–303, 2016.

[19] JA Bagnell, Joel Chestnutt, David M Bradley, and Nathan D Ratliff. Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems*, pages 1153–1160, 2007.

[20] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Ničković, and Sriram Sankaranarayanan. Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In *Lectures on Runtime Verification*, pages 135–175. Springer, 2018.

[21] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donze, Georgios Fainekos, Oded Maler, Dejan Nivckovic, and Sriram Sankaranarayanan. Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In *Lectures on Runtime Verification*, pages 135–175. Springer, 2018.

[22] Dimitri P Bertsekas. *REINFORCEMENT LEARNING AND OPTIMAL CONTROL*. Athena Scientific, 2019.

[23] Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[24] T Bourke and M Zélus Pouzet. a synchronous language with odes. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control (HSCC 2013), Philadelphia, PA, USA*, pages 8–11, 2013.

[25] Robert N Charette. This car runs on code. *IEEE spectrum*, 46(3):3, 2009.

[26] Steven Chen, Kelsey Saulnier, Nikolay Atanasov, Daniel D Lee, Vijay Kumar, George J Pappas, and Manfred Morari. Approximating explicit model predictive control using constrained neural networks. In *2018 Annual American Control Conference (ACC)*, pages 1520–1527. IEEE, 2018.

[27] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*, pages 258–263. Springer, 2013.

[28] Andrew Clark. Control barrier functions for complete and incomplete information stochastic systems. In *2019 American Control Conference (ACC)*, pages 2928–2935. IEEE, 2019.

[29] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[30] Jyotirmoy Deshmukh, Xiaoqing Jin, James Kapinski, and Oded Maler. Stochastic local search for falsification of hybrid systems. In *International Symposium on Automated Technology for Verification and Analysis*, pages 500–517. Springer, 2015.

[31] Simulink Documentation. Simulation and model-based design, 2020.

[32] Adel Dokhanchi, Bardh Hoxha, and Georgios Fainekos. Formal requirement debugging for testing and verification of cyber-physical systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 17(2):1–26, 2017.

[33] Adel Dokhanchi, Shakiba Yaghoubi, Bardh Hoxha, and Georgios Fainekos. Arch-comp17 category report: Preliminary results on the falsification benchmarks. In Goran Frehse and Matthias Althoff, editors, *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 48 of *EPiC Series in Computing*, pages 170–174. EasyChair, 2017.

[34] Adel Dokhanchi, Shakiba Yaghoubi, Bardh Hoxha, and Georgios Fainekos. Vacuity aware falsification for mtl request-response specifications. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1332–1337. IEEE, 2017.

[35] Adel Dokhanchi, Shakiba Yaghoubi, Bardh Hoxha, Georgios Fainekos, Gidon Ernst, Zhenya Zhang, Paolo Arcaini, Ichiro Hasuo, and Sean Sedwards. Arch-comp18 category report: Results on the falsification benchmarks. In *ARCH@ ADHS*, pages 104–109, 2018.

[36] Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*, pages 167–170. Springer, 2010.

[37] Alexandre Donzé and Oded Maler. Systematic simulation using sensitivity analysis. *Hybrid Systems: Computation and Control*, pages 174–189, 2007.

[38] Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 92–106. Springer, 2010.

[39] Tommaso Dreossi, Thao Dang, Alexandre Donzé, James Kapinski, Xiaoqing Jin, and Jyotirmoy V Deshmukh. Efficient guiding strategies for testing of temporal properties of hybrid systems. In *NASA Formal Methods Symposium*, pages 127–142. Springer, 2015.

[40] Tommaso Dreossi, Alexandre Donzé, and Sanjit A Seshia. Compositional falsification of cyber-physical systems with machine learning components. In *NASA Formal Methods Symposium*, pages 357–372. Springer, 2017.

[41] Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Kurt Keutzer, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Counterexample-guided data augmentation. *arXiv preprint arXiv:1805.06962*, 2018.

[42] Tommaso Dreossi, Somesh Jha, and Sanjit A Seshia. Semantic adversarial deep learning. In *International Conference on Computer Aided Verification*, pages 3–26. Springer, 2018.

[43] Souradeep Dutta, Xin Chen, and Sriram Sankaranarayanan. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *International Conference on Hybrid Systems: Computation and Control (HSCC)*, 2019.

[44] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Learning and verification of feedback control systems using feedforward neural networks. *IFAC-PapersOnLine*, 51(16):151–156, 2018.

[45] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Learning and verification of feedback control systems using feedforward neural networks. In *Analysis and Design of Hybrid Systems*, 2018.

[46] Gidon Ernst, Paolo Arcaini, Ismail Bennani, Alexandre Donze, Georgios Fainekos, Goran Frehse, Logan Mathesen, Claudio Menghi, Giulia Pedrinelli, Marc Pouzet, et al. Arch-comp 2020 category report: Falsification. *EPiC Series in Computing*, 2020.

[47] Gidon Ernst, Paolo Arcaini, Alexandre Donze, Georgios Fainekos, Logan Mathesen, Giulia Pedrielli, Shakiba Yaghoubi, Yoriyuki Yamagata, and Zhenya Zhang. Arch-comp 2019 category report: Falsification. *EPiC Series in Computing*, 61:129–140, 2019.

[48] Gidon Ernst, Sean Sedwards, Zhenya Zhang, and Ichiro Hasuo. Fast falsification of hybrid systems using probabilistically adaptive input. In *International Conference on Quantitative Evaluation of Systems*, pages 165–181. Springer, 2019.

[49] Peyman Mohajerin Esfahani, Debasish Chatterjee, and John Lygeros. The stochastic reach-avoid problem and set characterization for diffusions. *Automatica*, 70:43–56, 2016.

[50] Georgios E Fainekos and George J Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[51] Chuchu Fan. *Formal methods for safe autonomy: Data-driven verification, synthesis, and applications*. PhD thesis, University of Illinois at Urbana-Champaign, 2019.

[52] Chuchu Fan, Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. Progress on powertrain verification challenge with c2e2. In *ARCH@ CPSWeek*, pages 207–212, 2015.

[53] Chuchu Fan, James Kapinski, Xiaoqing Jin, and Sayan Mitra. Locally optimal reach set over-approximation for nonlinear systems. In *2016 International Conference on Embedded Software (EMSOFT)*, pages 1–10. IEEE, 2016.

[54] Francesca M Favarò, Nazanin Nader, Sky O Eurich, Michelle Tripp, and Naresh Varadaraju. Examining accident reports involving autonomous vehicles in california. *PLoS one*, 12(9):e0184952, 2017.

[55] Jaime F Fisac, Mo Chen, Claire J Tomlin, and S Shankar Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *Proceedings of the 18th international conference on hybrid systems: computation and control*, pages 11–20, 2015.

[56] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*, pages 379–395. Springer, 2011.

[57] Qitong Gao, Davood Hajinezhad, Yan Zhang, Yiannis Kantaros, and Michael M Zavlanos. Reduced variance deep reinforcement learning with temporal logic specifications. *ACM/IEEE International Conference on Cyber-Physical Systems*, 2019.

[58] Mark A Geistfeld. A roadmap for autonomous vehicles: State tort liability, automobile insurance, and federal safety regulation. *Calif. L. Rev.*, 105:1611, 2017.

[59] Luca Geretti, Julien Alexandre Dit Sandretto, Matthias Althoff, Luis Benet, Alexandre Chapoutot, Xin Chen, Pieter Collins, Marcelo Forets, Daniel Freire, Fabian Immler, et al. Arch-comp20 category report: Continuous and hybrid systems with nonlinear dynamics. *EPiC Series in Computing*, 74:49–75, 2020.

[60] Paul Glotfelter, Jorge Cortés, and Magnus Egerstedt. Nonsmooth barrier functions with applications to multi-robot systems. *IEEE control systems letters*, 1(2):310–315, 2017.

[61] Martin T Hagan, Howard B Demuth, and Orlando De Jesús. An introduction to the use of neural networks in control systems. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 12(11):959–985, 2002.

[62] Martin T. Hagan, Howard B. Demuth, and Orlando De Jesus. An introduction to the use of neural networks in control systems. *International Journal of Robust and Nonlinear Control*, 12(11):959–985, 2002.

[63] Nikolaus Hansen and Stefan Kern. Evaluating the cma evolution strategy on multimodal test functions. In *International Conference on Parallel Problem Solving from Nature*, pages 282–291. Springer, 2004.

[64] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[65] Mohammad Hekmatnejad, Shakiba Yaghoubi, Adel Dokhanchi, Heni Ben Amor, Aviral Shrivastava, Lina Karam, and Georgios Fainekos. Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic. In *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design*, page 6. ACM, 2019.

[66] Thomas A Henzinger, Peter W Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of computer and system sciences*, 57(1):94–124, 1998.

[67] Michael Hertneck, Johannes Köhler, Sebastian Trimpe, and Frank Allgöwer. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3):543–548, 2018.

[68] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[69] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[70] Zhenqi Huang, Yu Wang, Sayan Mitra, Geir E Dullerud, and Swarat Chaudhuri. Controller synthesis with inductive proofs for piecewise linear systems: An smt-based algorithm. In *2015 54th IEEE conference on decision and control (CDC)*, pages 7434–7439. IEEE, 2015.

[71] Kenneth J. Hunt, Daniel G. Sbarbaro, Rafat Zbikowski, and Peter Gawthrop. Neural networks for control systems - a survey. *Automatica*, 28:1083–1112, 1992.

[72] Fabian Immler, Matthias Althoff, Xin Chen, Chuchu Fan, Goran Frehse, Niklas Kochdumper, Yangge Li, Sayan Mitra, Mahendra Singh Tomar, and Majid Zamani. Arch-comp18 category report: Continuous and hybrid systems with nonlinear dynamics. In *Proc. of the 5th International Workshop on Applied Verification for Continuous and Hybrid Systems*, 2018.

[73] Radoslav Ivanov, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. pages 169–178, 2019.

[74] Pushpak Jagtap, Sadegh Soudjani, and Majid Zamani. Temporal logic verification of stochastic systems using barrier certificates. In *International Symposium on Automated Technology for Verification and Analysis*, pages 177–193. Springer, 2018.

[75] Mrdjan Jankovic. Robust control barrier functions for constrained stabilization of nonlinear systems. *Automatica*, 96:359–367, 2018.

[76] Xiaoqing Jin, Jyotirmoy V Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. Powertrain control verification benchmark. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 253–262. ACM, 2014.

[77] Kyle D Julian and Mykel J Kochenderfer. Neural network guidance for uavs. In *AIAA Guidance, Navigation, and Control Conference*, page 1743, 2017.

[78] Kyle D Julian, Jessica Lopez, Jeffrey S Brush, Michael P Owen, and Mykel J Kochenderfer. Policy compression for aircraft collision avoidance systems. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2016.

[79] James Kapinski, Jyotirmoy V Deshmukh, Xiaoqing Jin, Hisahiro Ito, and Ken Butts. Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques. *IEEE Control Systems Magazine*, 36(6):45–64, 2016.

[80] Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

[81] Niklas Kochdumper, Bastian Schürmann, and Matthias Althoff. Utilizing dependencies to obtain subsets of reachable sets. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–10, 2020.

[82] Shishir Kolathaya and Aaron D Ames. Input-to-state safety with control barrier functions. *IEEE control systems letters*, 3(1):108–113, 2018.

[83] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. dreach: δ-reachability analysis for hybrid systems. In *International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems*, pages 200–205. Springer, 2015.

[84] Philip Koopman and Michael Wagner. Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine*, 9(1):90–96, 2017.

[85] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299, 1990.

[86] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[87] Vidyadhar G Kulkarni. *Modeling and analysis of stochastic systems*. Crc Press, 2016.

[88] Jan Kuřátko and Stefan Ratschan. Combined global and local search for the falsification of hybrid systems. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 146–160. Springer, 2014.

[89] Harold J Kushner. Stochastic stability and control. Technical report, Brown Univ Providence RI, 1967.

[90] Nancy G Leveson and Clark S Turner. An investigation of the therac-25 accidents. *Computer*, 26(7):18–41, 1993.

[91] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pages 1071–1079, 2014.

[92] Nan Li, Anouck Girard, and Ilya Kolmanovsky. Optimal control based falsification of unknown systems with time delays: a gasoline engine a/f ratio control case study. *IFAC-PapersOnLine*, 51(31):252–257, 2018.

[93] Xiao Li and Calin Belta. Temporal logic guided safe reinforcement learning using control barrier functions. *arXiv preprint arXiv:1903.09885*, 2019.

[94] Xiao Li, Yao Ma, and Calin Belta. A policy search method for temporal logic specified reinforcement learning tasks. In *2018 Annual American Control Conference (ACC)*, pages 240–245. IEEE, 2018.

[95] Daniel Liberzon. *Switching in systems and control*. Springer Science & Business Media, 2003.

[96] Lars Lindemann and Dimos V Dimarogonas. Control barrier functions for multi-agent systems under conflicting local signal temporal logic tasks. *IEEE control systems letters*, 2019.

[97] Lars Lindemann, George J Pappas, and Dimos V Dimarogonas. Control barrier functions for nonholonomic systems under risk signal temporal logic specifications. *arXiv preprint arXiv:2004.02111*, 2020.

[98] Anirudha Majumdar and Marco Pavone. How should a robot assess risk? towards an axiomatic theory of risk in robotics. In *Robotics Research*, pages 75–84. Springer, 2020.

[99] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[100] Nick Malone et al. Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field. *IEEE Transactions on Robotics*, 33(5):1124–1138, 2017.

[101] Logan Mathesen, Shakiba Yaghoubi, Giulia Pedrielli, and Georgios Fainekos. Falsification of cyber-physical systems with robustness uncertainty quantification through stochastic optimization with adaptive restart. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 991–997. IEEE, 2019.

[102] Truong Nghiem, Sriram Sankaranarayanan, Georgios Fainekos, Franjo Ivancić, Aarti Gupta, and George J Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pages 211–220, 2010.

[103] Quan Nguyen and Koushil Sreenath. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. In *2016 American Control Conference (ACC)*, pages 322–328. IEEE, 2016.

[104] Bernt Øksendal. Stochastic differential equations. In *Stochastic differential equations*, pages 65–84. Springer, 2003.

[105] Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. Smooth operator: Control using the smooth robustness of temporal logic. In *Control Technology and Applications (CCTA), 2017 IEEE Conference on*, pages 1235–1240. IEEE, 2017.

[106] Giordano Pola, Manuela L Bujorianu, John Lygeros, and Maria Domenica Di Benedetto. Stochastic hybrid models: An overview. *IFAC Proceedings Volumes*, 36(6):45–50, 2003.

[107] S. Prajna, A. Jadbabaie, and G. J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428, 2007.

[108] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[109] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 81–87. IEEE, 2014.

[110] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via regularized behavioral cloning. *arXiv preprint arXiv:1905.11108*, 2019.

[111] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[112] Royal Academy of Engineering. Innovation in autonomous systems, 2015. [Online].

[113] Vicenc Rubies Royo, David Fridovich-Keil, Sylvia Herbert, and Claire J Tomlin. Classification-based approximate reachability with guarantees applied to safe trajectory tracking. *arXiv preprint arXiv:1803.03237*, 2018.

[114] S-TaLiRo: Temporal Logic Falsification Of Cyber-Physical Systems. `https://sites.google.com/a/asu.edu/s-taliro/s-taliro`, 2013. [Online; accessed April-2014].

[115] Abbas Saadat. Defect information report (nhtsa recall 14v-053). *Source: http://www. odinhtsa. dot. gov/acms/cs/jaxrs/download/doc/UCM*, 450071, 2014.

[116] Cesar Santoyo, Maxence Dutreix, and Samuel Coogan. A barrier function approach to finite-time stochastic system verification and control. *arXiv preprint arXiv:1909.05109*, 2019.

[117] Meenakshi Sarkar, Debasish Ghose, and Evangelos A Theodorou. High-relative degree stochastic control lyapunov and barrier functions. *arXiv preprint arXiv:2004.03856*, 2020.

[118] Johann Schumann and Yan Liu. *Applications of Neural Networks in High Assurance Systems*, volume 268 of *SCI*. Springer, 2010.

[119] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv:1708.06374v6*, 2018.

[120] Simone Silvetti, Alberto Policriti, and Luca Bortolussi. An active learning approach to the falsification of black box cyber-physical systems. In *International Conference on Integrated Formal Methods*, pages 3–17. Springer, 2017.

[121] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 7461–7472, 2018.

[122] Mohit Srinivasan, Samuel Coogan, and Magnus Egerstedt. Control of multi-agent systems with finite time control barrier certificates and temporal logic. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1991–1996. IEEE, 2018.

[123] Jacob Steinhardt and Russ Tedrake. Finite-time regional verification of stochastic non-linear systems. *The International Journal of Robotics Research*, 31(7):901–923, 2012.

[124] Paulo Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

[125] Cumhur Erkan Tuncali, Georgios Fainekos, Hisahiro Ito, and James Kapinski. Simulation-based adversarial test generation for autonomous vehicles with machine learning components. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1555–1562. IEEE, 2018.

[126] Cumhur Erkan Tuncali, Shakiba Yaghoubi, Theodore P Pavlic, and Georgios Fainekos. Functional gradient descent optimization for automatic test case generation for vehicle controllers. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1059–1064. IEEE, 2017.

[127] Cristian-Ioan Vasile, Vasumathi Raman, and Sertac Karaman. Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3840–3847. IEEE, 2017.

[128] Graham Wild, Kellie Gavin, John Murray, Jose Silva, and Glenn Baxter. A post-accident analysis of civil remotely-piloted aircraft system accidents and incidents. *Journal of Aerospace Technology and Management*, 9(2):157–168, 2017.

[129] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[130] Andrew Winn and A Agung Julius. Safety controller synthesis using human generated trajectories. *IEEE Transactions on Automatic Control*, 60(6):1597–1610, 2015.

[131] Wei Xiao and Calin Belta. Control barrier functions for systems with high relative degree. *arXiv preprint arXiv:1903.04706*, 2019.

[132] Wei Xiao, Calin Belta, and Christos G Cassandras. Decentralized merging control in traffic networks: A control barrier function approach. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 270–279, 2019.

[133] Xiangru Xu, Paulo Tabuada, Jessy W Grizzle, and Aaron D Ames. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27):54–61, 2015.

[134] Shakiba Yaghoubi and Georgios Fainekos. Hybrid approximate gradient and stochastic descent for falsification of nonlinear systems. In *American Control Conference (ACC), 2017*, pages 529–534. IEEE, 2017.

[135] Shakiba Yaghoubi and Georgios Fainekos. Local descent for temporal logic falsification of cyber-physical systems. In *Seventh Workshop on Design, Modeling and Evaluation of Cyber Physical Systems*, 2017.

[136] Shakiba Yaghoubi and Georgios Fainekos. Falsification of temporal logic requirements using gradient based local search in space and time. *IFAC-PapersOnLine*, 51(16):103–108, 2018.

[137] Shakiba Yaghoubi and Georgios Fainekos. Gray-box adversarial testing for control systems with machine learning components. In *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, pages 179–184, New York, NY, USA, 2019. ACM.

[138] Shakiba Yaghoubi and Georgios Fainekos. Gray-box adversarial testing for control systems with machine learning components. In *ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, 2019.

[139] Shakiba Yaghoubi and Georgios Fainekos. Worst-case satisfaction of stl specifications using feedforward neural network controllers: a lagrange multipliers approach. *ACM Transactions on Embedded Computing Systems (TECS)*, 18, 2019.

[140] Shakiba Yaghoubi, Georgios Fainekos, and Sriram Sankaranarayanan. Training neural network controllers using control barrier functions in the presence of disturbances. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[141] Shakiba Yaghoubi, Keyvan Majd, Georgios Fainekos, Tomoya Yamaguchi, Danil Prokhorov, and Bardh Hoxha. Risk-bounded control using stochastic barrier functions. *IEEE Control Systems Letters*, 2020.

[142] Guang Yang, Bee Vang, Zachary Serlin, Calin Belta, and Roberto Tron. Sampling-based motion planning via control barrier functions. In *Proceedings of the 2019 3rd International Conference on Automation, Control and Robots*, pages 22–29. ACM, 2019.

[143] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 528–535. IEEE, 2016.

[144] Xiaojing Zhang, Monimoy Bujarbaruah, and Francesco Borrelli. Near-optimal rapid mpc using neural networks: A primal-dual policy learning framework. *arXiv preprint arXiv:1912.04744*, 2019.

[145] Siqi Zhou, Mohamed K Helwa, and Angela P Schoellig. Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5201–5207. IEEE, 2017.

[146] Aditya Zutshi, Jyotirmoy V Deshmukh, Sriram Sankaranarayanan, and James Kapinski. Multiple shooting, cegar-based falsification for hybrid systems. In *Proceedings of the 14th International Conference on Embedded Software*. ACM, 2014.