Contract-Based Safety Verification for Autonomous Driving

by

Lucas Liebenwein

B.Sc., Swiss Federal Institute of Technology Zurich (2015)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2018

Author .....
Department of Electrical Engineering and Computer Science
August 28, 2018

Certified by .....
Daniela Rus
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Contract-Based Safety Verification for Autonomous Driving

by

Lucas Liebenwein

Submitted to the Department of Electrical Engineering and Computer Science
on August 28, 2018, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

The safe, successful deployment of autonomous systems under real-world conditions, in part, hinges upon providing rigorous performance and safety guarantees. This thesis considers the problem of establishing and verifying the safety of autonomous systems. To this end, we present a novel framework for the synthesis of safety constraints for autonomous systems, so-called *safety contracts*, that can be applied to and used by a wide set of real-world systems by acting as a design requirement for the controller implementation of the system. The contracts consider a large variety of road models, guarantee that the controlled system will remain safe with respect to probabilistic models of traffic behavior, and ensure that it will follow the rules of the road. We generate contracts using reachability analysis in a reach-avoid problem under consideration of dynamic obstacles, i.e., other traffic participants. Contracts are then derived directly from the reachable sets. By decomposing large road networks into local road geometries and defining assume-guarantee contracts between local geometries, we enable computational tractability over large spatial domains. To efficiently account for the behavior of other traffic participants, we iteratively alternate between falsification to generate new traffic scenarios that violate the safety contract and reachable set computation to update the safety contract. These counterexamples to collision-free behavior are found by solving a gradient-based trajectory optimization problem. We demonstrate the practical effectiveness of the proposed methods in a set of experiments involving the Manhattan road network as well as interacting multi-car traffic scenarios.

Thesis Supervisor: Daniela Rus
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to start by thanking my advisor, Daniela Rus, who has been a constant source of inspiration and motivation throughout the last two years at the Distributed Robotics Lab. Her counseling and guidance have helped me over and over again, and allowed me to step into the world of research. It has been an invaluable experience and I look forward to the coming years.

I would also like to thank Sertac Karaman who has provided me with crucial insights during our many meetings and discussions. Cristian-Ioan Vasile and Jonathan DeCastro also deserve particular thanks for all their hard work and enlightening moments. They introduced me to formal methods and verification, and have taught me a great deal about these techniques. Many of the ideas in this thesis originated from our discussions.

During my undergraduate research, I was lucky enough to be advised by Max Kriegleder and Raffaello D'Andrea. It was my first exposure to robotics, and I have been in the field since then. I will always be grateful to them for guiding me through this time and teaching me so much about science. When I first visited MIT, Emilio Frazzoli took it upon him to help me choose an advisor and introduced me to Daniela. His counseling during these days was a tremendous support to help me settle into graduate school.

My time here would have not been the same without the members of the Distributed Robotics Lab, many of which have become my dear friends. I would like to thank Felix Naser, with whom I shared an office for a year. Our countless crazy moments are a more than welcoming distraction to keep a fresh mind. I will always remember the endless hours and nights Cenk Baykal and I have spent solving problem sets and working on research. Cenk's passion for theoretical computer science has inspired me to strike a new path in my research. I am fortunate to be able to call him my close collaborator, office neighbor, and friend. I would also like to acknowledge my other friends and members of the lab, including Brandon Araki, Thomas Balch, Igor Gilitschenski, Robert Katzschmann, Teddy Ort, Wilko Schwarting, Alex Wallar,

and many more. It is an honor to be part of such a unique group of people.

I would have never come this far if it were not for my family. Their unconditional love and support gave me strength and courage to pursue my dreams. Among my many family members, I would like to thank my mother, Jutta, for all the moments of counseling and advice in every aspect of my life; my father, Karl, for being there when it matters the most; my sister, Leonie, for being the best sister one could imagine; my two brothers, Constantin and Nicolaus, for being such sweet and joyful boys; my aunt and uncle, Karin and Wolfgang, for always having an open ear for my sorrows; and my grandmother, Berta, for being such an inspiration and role model to look up to.

Finally, I would like to thank my wife and love of my life, Pia. She has been there for me at every imaginable moment to guide me, support me through my struggles, and give me the strength when I needed it the most. I could not be here without her. Her passion, joy, and wonderful heart are unparalleled and made me into the person I am today.

I dedicate this thesis to you, Pia.

Contents

1 Introduction **17**

1.1 Contributions 21

1.2 Outline of Thesis 21

2 Related Work **23**

2.1 Local Motion Planning 23

2.2 Reachability Analysis 24

2.3 Verification and Synthesis 25

2.4 Falsification 27

3 Contract Synthesis with Known, Dynamic Obstacles **29**

3.1 Problem Definition 29

3.1.1 Ego-car 30

3.1.2 Road Network 30

3.1.3 Traffic 30

3.1.4 Controllers and Driving Behaviors 31

3.1.5 Safety 31

3.2 Methods 32

3.2.1 Library of Parameterized Models 34

3.2.2 Reachability Analysis 35

3.2.3 Verification of Controller Contracts 36

3.2.4 Road Network Verification 39

3.3 Analysis 40

3.4 Implementation . . . 41
3.4.1 Reachability Tool . . . 41
3.4.2 Set Representation . . . 42
3.4.3 Set Pruning . . . 42
3.4.4 Reduction of Complexity . . . 42
3.5 Results . . . 45
3.5.1 Dynamic Motion Model and Dynamic Constraints . . . 45
3.5.2 Other Vehicles . . . 46
3.5.3 Drivable Space . . . 47
3.5.4 Rules of the Road . . . 48
3.5.5 Admissible Configurations . . . 48
3.5.6 Road Library and Network . . . 49
3.5.7 Traffic Model . . . 49
3.5.8 Experiments . . . 50

4 Contract Synthesis with Counterexample-Guided Obstacles 55
4.1 Problem Definition . . . 55
4.1.1 Stochastic Models of the Traffic System . . . 56
4.1.2 Problem Formulation . . . 57
4.2 Methods . . . 58
4.2.1 Overview . . . 58
4.2.2 Gradient-Based Probabilistic Falsification . . . 60
4.2.3 Collision-Free Safety Conditions . . . 62
4.2.4 Reachability with Contracts . . . 63
4.2.5 Computation of Contract . . . 64
4.2.6 Rules of the Road . . . 65
4.3 Implementation . . . 66
4.4 Results . . . 69

5 Conclusion 73
5.1 Conclusion . . . 73

5.2 Future Work 74
5.3 Lessons Learned 75
5.4 Funding 76

List of Figures

1-1 The network shows the Manhattan street grid. We verified a part of Manhattan consisting of ca. 130 blocks with 180 intersections and 330 straight road segments (marked black) using a library of 22 verified segments. This underlines the gain in efficiency of the compositional-based approach compared to that of a direct naive approach. 18

3-1 Consider the library $\mathcal{M} = \{m^1, m^2\}$ composed of a straight road m^1 and a four-way intersection m^2. (a) The four-way intersection m^2 is shown. (b) The entry set is propagated forward, and, (c), concurrently pruned of unsafe states induced by other cars. (d) The safe exit set is the intersection of the safe reachable set at time step H, the verification horizon, and the exit set. (e) The safe entry set is computed by backward propagating the safe exit set. (f) Lastly, the composition of the models using the associated assume-guarantee contracts enables us to certify road networks. 34

3-2 (a) The exact Minkowski swept volume between the ego-car and another vehicle, which represents configurations in collision, is shown. (b) Drivable space in a four-way intersection left-turn scenario obtained by taking the Minkowski sum between the rectangular ego-car and the road over Δq, i.e., $\mathfrak{D}(\mathcal{Z})$. This is used to trim the admissible state space. Because of symmetry and for brevity, θ is only shown for $[0, \pi]$. Approximations to 10 θ-slices are used in the implementation to reduce computational complexity. 47

3-3 The dynamic obstacles (black) and the drivable space (grey) are shown for time k on the left. Set subtraction yields the admissible configuration volume $\mathfrak{A}_k$ (right). 49

3-4 The forward propagation of the reachable set is shown for a left turning maneuver. Blue sets indicate the safely reachable configuration set $g_q(\widetilde{Z_k})$ (top row), and the position set $g_p(\widetilde{Z_k})$ (bottom row) of the ego-car for various times k. Black sets mark the swept volume $\mathfrak{C}_k^i(\mathcal{Z})$ (top row) and the footprint $\mathcal{B}^i(x_k^i;\mathcal{R})$ (bottom row) of other traffic participants. The entry and exit sets are shown in green and red, respectively. Note how the ego-car maintains a safe distance to the other cars and the road boundaries at all times. 51

3-5 Backward propagation on the four-way intersection m for various times k. At $t = 6.0s$, we start out at $\widehat{Z_T} = exit(m, 1, 4) = \mathfrak{O}_4$ and compute the backward reachable set $\widehat{Z_k}$ (marked blue) for each time k to obtain the safe entry set $entry(m, 1, 4)$. 52

3-6 A selection of verified road models, comprised of various intersections and straight roads, is shown together with the reachable set (blue) and other traffic participants (black) at the indicated timestep. The initial and final set are marked green and red, respectively. 53

3-7 The box plot indicates computation times for various parts of one iteration with a fixed timestep of $h = 0.05s$ averaged over all conducted experiments. 53

4-1 Two iterations of the overall approach. 60

4-2 Different iterations of the approach. Within each iteration, the upper two plots indicate a counterexample trajectory of the traffic system that falsifies collision-free behavior under the proposed contracts. The lower plot illustrates a new contract that guards against the counterexample. 70

4-3 The contract for timestep $t = 4.8s$ at iteration 4 for each set of parameters. 71
4-4 The log-likelihood for each test case across all iterations. The red $\times$ marks iterations where the contract terminated with an empty set, and the green dashed line indicates the chance constraint α. 71

List of Tables

3.1 Symbols table. 33

4.1 Extended symbols table. 59
4.2 Rules of the road for highway scenarios. 67
4.3 Parameters used to model driver behaviors for the traffic cars. 69

Chapter 1

Introduction

The way we use and think about mobility and transportation has changed significantly in the last years due to many recent developments in autonomous driving. With the increasing space of potential applications, the required safety certificates for deployment are becoming increasingly difficult to manage both in terms of computational complexity for simulations and the hours spent on road testing. Moreover, safety guarantees are not only required for the safe deployment of systems but can also act as a key stepping stone in gaining customers' trust in autonomous systems.

To this end, we consider the problem of providing rigorous safety guarantees for autonomous car controllers through formal verification methods with respect to vehicle, environment, and traffic models. Formal verification can fill the gap of certification by providing a platform to assess safety with clear assumptions and guarantees. While simulation and testing are undoubtedly essential tools for deployment of complex systems, they lack the completeness, and therefore the guarantees, of verification, potentially missing out on rare and hard-to-characterize events.

Recent studies [36] have indicated that the requirement to demonstrate safety for an autonomous car is *hundreds of millions* of miles of testing taking possibly *tens of years* to complete. To meet these proof-of-safety demands, testing and simulation, which provide very detailed insights for specific events, can be supplemented with verification frameworks, which provide insights for an entire set of events, though often less detailed. In such a sense, verification provides a way to check over an

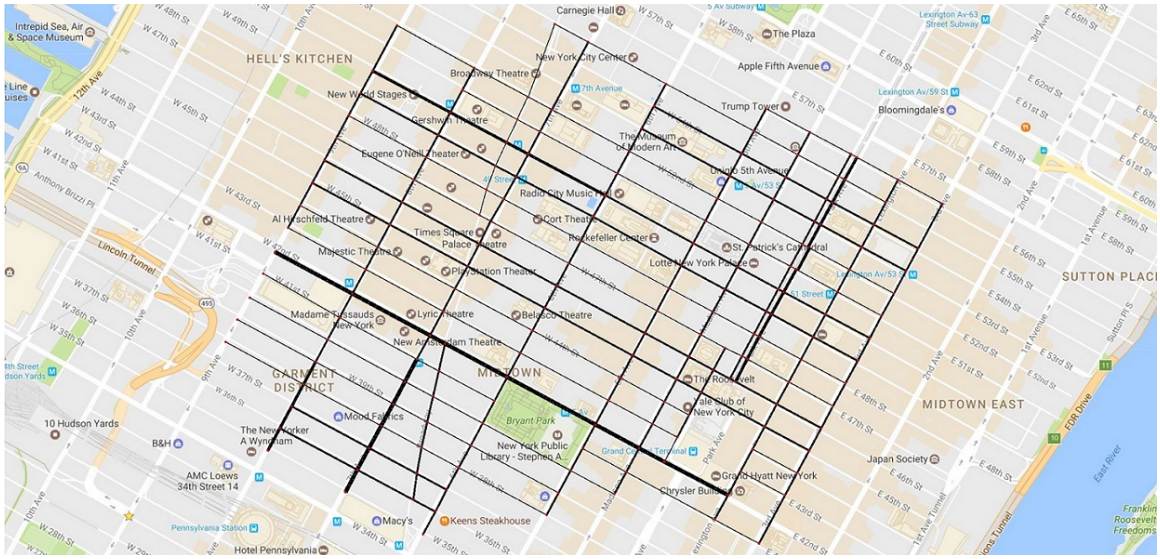


Figure 1-1: The network shows the Manhattan street grid. We verified a part of Manhattan consisting of ca. 130 blocks with 180 intersections and 330 straight road segments (marked black) using a library of 22 verified segments. This underlines the gain in efficiency of the compositional-based approach compared to that of a direct naive approach.

infinite number of simulated trajectories as opposed to straightforward case-based simulation at the cost of reduced model complexity.

Despite its appeal, verification can quickly become computationally intractable. For any realistic scenario involving a large number of interacting agents, a large road network, and complex autonomous system implementations, obtaining correctness guarantees becomes prohibitively expensive. Since verification usually requires reasoning about the entire set of possible outcomes, which is potentially of infinite cardinality, analytic solutions to verification do not exist, and instead we require efficient approximation algorithms. Moreover, the verification process typically must be repeated whenever any part of the autonomous systems is modified, thus heavily reducing the practicability and reusability of such guarantees.

Current state-of-the-art work, including [3, 25, 45, 47] among others, in verification has considered the synthesis of safety guarantees for a variety of autonomous systems, such as aerial systems and autonomous vehicles. Previous work has either leaned towards simplifying the system to be verified [25, 47] or towards considering more conservative, coarse approximations of complex systems [3, 45] to cope with

the issue of computational tractability. As a consequence, the question of combining large-scale safety guarantees with complex control systems has not been adequately addressed. In this regard, our work aims at closing the research gap between the vast, mostly independent prior work in verifying large-scale systems and in verifying complex systems by proposing methods to synthesize safety guarantees that are simultaneously applicable to large-scale *and* complex systems.

In particular, in this thesis we present a novel framework to concurrently verify and synthesize compositional safety contracts for autonomous systems that are embedded in probabilistic traffic scenarios within large road networks and consider rules of the road. By means of generating contracts – instead of verifying the system itself – we may reuse the obtained safety contract, which constitutes a set of constraints in the autonomous system's state space, enabling scalable verification across any autonomous system that implements the safety contract. By means of decomposing the verification tasks into local tasks – instead of verifying the entire system at once – that consider particular traffic scenarios and road geometries, we achieve computational tractability over large domains.

The local verification task and contract synthesis method is based on reachability analysis, which computes the safe (collision-free) subset of the state space for each timestep. Within these local tasks, we verify local road models, such as intersections, and synthesize the contract that allows an autonomous system to safely traverse the local road model. We then compose guarantees for larger road networks through appropriate safe entry and exit sets, i.e., assume-guarantee contracts, which are part of the safety contract. Moreover, we employ falsification to search for relevant traffic counterexamples from the rich probabilistic behavior models of other traffic participants. These counterexamples are used to populate the local verification tasks with traffic. That way, we avoid the need of verifying over the entire set of possible behaviors and overcome the tractability issues arising from verifying such a complex system. The falsification method to find traffic counterexamples is based on a gradient-based trajectory optimization problem with chance constraints. Both methods, reachability analysis and falsification, are combined in an iterative fash-

ion to verify local road models where during each iteration we first find a proposal contract using reachability analysis and then try to falsify the proposed contract by searching for a counterexample.

In this sense, we provide safety guarantees for controllers and planners of autonomous systems that can operate under consideration of the safety contract, which can be readily implemented via state-space constraints. The focus hereby lies on obtaining safety guarantees for road networks that are known a priori, such that we can obtain the necessary safety guarantees in an offline procedure before deployment. Moreover, as we consider a priori known road networks, the probabilistic traffic model can be validated and augmented using real-world traffic data from the considered road network to ensure its accuracy. While we do not actively consider perception as part of the verification procedure, we note that the modular, decomposable approach of our method can be leveraged to also introduce assume-guarantee contracts between the control system and the perception system, such that (potentially probabilistic) safety guarantees for the perception system can be derived independently.

The proposed framework enables a variety of novel capabilities. Using compositionality, we can verify the safety properties over domains, such as the road network shown in Figure 1-1, previously considered to be too large to be tractable using traditional verification techniques. We also argue that the framework results in explainable verification since the found traffic counterexamples from falsification quantify the type of behavior that the contract can and/or cannot guard against. Moreover, through probabilistic modelling of traffic agents, we can evaluate the particular behavior of certain traffic agents in terms of the amount of risk a certain counterexample imposes. The contract synthesis method itself can account for rules of the road while ensuring safety. We can therefore evaluate the performance of the system, not only in terms of safety but also in terms of the desired behavior, and limit the type of behavior that is allowed through an explainable set of rules.

1.1 Contributions

This thesis contributes the following:

1. A safety verification problem formulation that entails providing safety guarantees for a large set of autonomous systems and introduces the notion of safety contracts as verification guarantee.

2. A contract synthesis procedure based on reachability analysis that considers rules of the road, local road geometries, road networks, and probabilistic traffic behavior.

3. A gradient-based falsification approach that enables efficient generation of a wide variety of probabilistic traffic scenarios with tunable behavior via chance constraints.

4. Domain-specific methods and implementations to overcome tractability issues, including efficient methods for reachability analysis and set-based operations.

5. Empirical results demonstrating the broad applicability and practical effectiveness on real-world inspired traffic scenarios, including simulation results on the Manhattan road network and highway overtaking maneuvers.

Preliminary versions of these results have appeared in [21, 42].

1.2 Outline of Thesis

This thesis is organized as follows. In Chapter 2, we present related work in safety verification and discuss the scope and limitation of previous work. In Chapter 3, we formally introduce the safety verification problem and describe the verification framework used to verify and synthesize safety contracts. We show how reachability analysis can be leveraged to synthesize safety contracts in dynamic, known traffic scenarios and present a compositional approach to compute safety guarantees over

large spatial domains. We also present a case study on the Manhattan road network. In Chapter 4, we relax the assumption of known, dynamic traffic scenarios via falsification. We show how we use a gradient-based trajectory optimization method to identify probabilistic traffic counterexamples, which can be used to update the contract in an iterative manner. Empirical results on a highway overtaking scenario are also presented. In Chapter 5, we conclude with a discussion on the presented approaches and mention potential directions for future work.

Chapter 2

Related Work

Our work leverages and hinges upon prior work in local motion planning, reachability analysis, formal verification and synthesis, and falsification.

2.1 Local Motion Planning

Methods to compute safe trajectories for autonomous vehicles in dynamic environments have been proposed in various contexts [50]. Using input space discretization, motion planning can be performed on a grid with graph algorithms [26, 64]. Sampling-based methods, such as rapidly exploring random trees [38], sample the state space for points and aim at generating an optimal time-discretized trajectory from the sampled points. Receding horizon control [24, 59], or model-predictive control, leverages nonlinear optimization tools to directly optimize a trajectory over a cost map under consideration of constraints, such as dynamics. This can also be applied for shared-control of highly automated vehicles [57] where the authors consider the human input as additional term in the cost function to minimize the deviation from the human input. These methods work well in practice but usually compute valid and safe trajectories only up to a pre-defined time horizon with no global and long term guarantees. In this work, we aim to obtain guarantees via safety contracts, which are employed by the local planner.

Moreover, robust motion planning aims to generate a proposal trajectory while

accounting for disturbances, such as uncertainty in the dynamics and the state of the robot. This yields to approaches such as model predictive control with tubes [46], where the tubes represent a safety margin around the nominal, desired trajectory. Other approaches include planning using motion primitives [29] based on Hamilton-Jacobi reachable sets and a sampling-based planner that can reason about uncertainty [41]. Moreover, the concept of funnels [45, 62], where a library of funnel-based motion primitives is built up to account for uncertainty in the state, has been shown to be applied to robust motion planning. In contrast to some of the prior work (such as [24, 46]), our work explicitly accounts for reachable sets based on the underlying dynamics of the robot system. Compared to approaches such as [29, 45, 62], where dynamics are also accounted for through the appropriate reachable sets, our work aims at building a library of safety constraints, i.e., a safety contract, that can account for a wide variety of dynamic obstacles while simultaneously being applicable to a large variety of motion planners. In other words, the safety contract may be implemented by any desired motion planner, either sampling-based or optimization-based for example.

2.2 Reachability Analysis

A large body of literature has been devoted to the formal analysis of reachability of systems. For finite systems efficient search mechanisms over the state-transition graph can be used to check the reachable states, e.g., Clarke et al. consider binary decision diagrams to this end [15]. For continuous systems, the authors of [9] and [13] propose to use sampling-based methods to compute reachable sets. In [35], the reachable set of continuous systems is computed using a combination of Runge-Kutta methods and affine arithmetic. The reachable sets of hybrid systems have also been extensively investigated in various contexts [4, 5, 12, 47].

Accurate reachability analysis necessitates the computation of the reachable states from an (uncountable) set of states, which is computationally intractable in practice [60]. Therefore, a vast collection of prior work has focused on developing approx-

imation algorithms, such as finite abstractions, for the computation of approximate reachable sets. To this end, an approach to computing overapproximations of reachable sets using zonotopes is presented in [2] and implemented as the CORA toolbox. The authors propose to approximate nonlinear systems polynomial systems and to account for approximation errors by appropriately inflating the underlying zonotopes. Taylor flow tubes are used to compute overapproximations of reachable sets in the work of [11]. Tools such as HyTech [32] and [14] consider only linear dynamics. In [25, 47] the task of computing reachable sets is cast as Hamilton-Jacobi Partial Differential Equations (PDEs) and standard tools for solving PDEs are used. Virtually all of these tools, however, compute over-approximations and cast the generally (highly) non-linear system dynamics as polynomials or even linear functions, which results in potentially unbounded error terms. Moreover, they are highly sensitive to the dimensionality of the input space and suffer to a great extent from the curse of dimensionality [47]. More recently, the work in [43] considers a sampling-based approximation method to underapproximation of reachable sets with provable approximation guarantees.

In this thesis, we consider the use of reachability analysis to verify the safety over local road models, which is used to synthesize safety contracts as well as guarantees over larger domains. This way, we avoid the computational tractability issues over large state space domains while allowing to synthesize contracts and guarantees over large domains.

2.3 Verification and Synthesis

Applications of reachability analysis range from ensuring the safety of mobile robots in human environments to flight maneuver verification. In particular, the work in [10, 23, 27, 52, 58, 67] investigates applications of reachability analysis to verifying grasping and manipulation tasks. Other applications include aerial robotics [30] and mobile robotics [44].

Verification has been employed in a variety of other safety-critical domains as well,

such as aerospace [7] and automotive [48, 49, 66]. In the automotive field, prior work has focused on time-bounded behaviors and simple motion primitives rather than the verification of controlled systems over complex, structured environments. In [1] and [3], the authors consider the use of reachability analysis for online safe motion planning in an autonomous vehicle. Accounting for model and sensing uncertainties through reachability analysis, the proposed method can reason about all possible future scenarios under consideration.

The contractual approach to verification has been proposed for complex system designs in many different domains [55] and has found numerous applications in recent verification and synthesis techniques, which exploit compositional approaches [6, 18, 39, 53] and assume-guarantee contracts [33, 54]. These approaches enable the separation of the controller design from the verification procedure and enable verification over larger, otherwise computationally intractable, domains. Our work is similar to [39, 40], where a compositional approach to synthesizing contracts is presented for traffic networks that must adhere to global specifications.

The idea of creating constraints based on a behavioral model may be viewed as an instance of robust explicit model-predictive control. The approaches in [19, 65] aim to synthesize controllers that satisfy a high-level task specification while in [20] the goal is to synthesize simple partitions of the state-space based on reachable sets.

In contrast to our work, the aforementioned approaches assume simple, non-probabilistic environment behaviors and are limited in the scope of the system and/or model considered. On the other hand, the contracts considered in this thesis are represented as simple state-space constraints on the car's behaviors, which are shown to be valid for highway-type scenarios, including lane keeping, lane changing, and merging, as well as city-type scenarios, including various turning maneuvers in intersections. To this end, we leverage compositional and contract-based verification to simultaneously achieve scalability and computational tractability in the model (large road networks), the system (control system of car), as well as the considered traffic scenarios (allowed behaviors of dynamic obstacles).

2.4 Falsification

Falsification [9, 13, 51] aims to find counterexamples that violate a given property and enables the analysis of more complex systems than verification at the cost of completeness. The problem – as opposed to verification – becomes one of finding failures, rare events within distributions capturing realistic driving behaviors, which can be difficult to solve.

Bhatia and Frazzoli propose to use sampling-based algorithms, such as rapidly exploring random trees, to capture failure events by sampling points until either a counterexample is found or the search space is exhausted. Counterexample guided abstraction refinement methods, such as the work presented in [16, 37], are based on the idea to initially start with an abstract, simple model that may produce counterexamples. Through an iterative approach, the model is then refined in subsequent steps to produce more relevant counterexamples. The authors of [56] propose to use the cross-entropy method to efficiently guide the sampling of random scenarios to find relevant counterexamples.

In contrast to sampling-based methods, such as [9, 56], we leverage a gradient-based probabilistic optimization to falsification of systems involving a large number of agents in short scenarios with few discrete decisions. This approach allows us to quickly converge on solutions by optimizing their utility at each iteration step and thus identify highly-relevant traffic scenarios, i.e., a collection of trajectories of dynamic obstacles. Moreover, the falsification-based approach to identify relevant traffic scenarios enables computational tractability since we are not required to verify the system over the entire domain of possible traffic scenarios. We also note that – similar to related work – the highly nonlinear nature of the problem prevents a globally-optimal solution, i.e., conclude that the added certificates are a formal proof of safety. However, our approach seeks to iteratively find locally-optimal counterexamples at each step of the contract-generation process and hence targets important failure cases that sampling-based approaches may miss.

Chapter 3

Contract Synthesis with Known, Dynamic Obstacles

In this chapter[1], we define the safety verification problem and present a framework to concurrently synthesize and verify safety contracts, i.e., a set of state space constraints for the autonomous system (ego-car). The safety contract is computed using reachability analysis and can be implemented by any control strategy that can account for state-space constraints, such as receding horizon and sampling-based controllers. We show how we can synthesize safety contracts for a large variety of local road geometries and how we can subsequently use these local contracts to synthesize safety contracts for large road networks. For the sake of simplicity, traffic, i.e., dynamic obstacles, are assumed to be known. This assumption is further relaxed in Chapter 4.

3.1 Problem Definition

In this section, we introduce the safety verification problem of controllers for autonomous cars. We define models for the ego-car, the road network, traffic, controllers, and safety that form the context of the verification problem. An overview of

[1]This chapter is based on Lucas Liebenwein, Wilko Schwarting, Cristian-Ioan Vasile, Jonathan DeCastro, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Compositional and contract-based verification for autonomous driving on road networks. In *International Symposium on Robotics Research*. International Foundation of Robotics Research, 2017.

the notation used in this chapter is given in Table 3.1.

3.1.1 Ego-car

The ego-car is defined as a dynamical system $V = (\mathcal{Z}, \mathcal{R}, \mathcal{U}, f_{ego}, g_p, g_q)$ evolving according to $z_{k+1} = f_{ego}(z_k, u_k)$, where $\mathcal{Z} \subseteq \mathbb{R}^{n_{ego}}$ is the state space, $\mathcal{R} \subset \mathcal{Z}$ the workspace, $\mathcal{U} \subseteq \mathbb{R}^{n_u}$ the control space, $\mathcal{Q} \subset SE(2)$ the configuration space (pose) of the car, and z_k the state of the car at time k. Further, let $f_{ego} : \mathcal{Z} \times \mathcal{U} \to \mathcal{Z}$, $g_p : \mathcal{Z} \to \mathbb{R}^2$, and $g_q : \mathcal{Z} \to \mathcal{Q}$ be the Lipschitz continuous (invertible) dynamics, observation function, and configuration space submersion, respectively. Then, $p_k = g_p(z_k)$ and $q_k = g_q(z_k)$ denote the location and pose of the car at time k, respectively. Moreover, let $\mathcal{B}(z_k) \subset \mathcal{Z}$ be the ego-car's footprint at time k, and let

$$\mathcal{B}(Z_k) = \bigcup_{z_k \in Z_k} \mathcal{B}(z_k)$$

denote the ego-car's footprint for a set of states $Z_k \subseteq \mathcal{Z}$. Finally, let $t_k = hk$ denote the time at k, where h is the time step.

3.1.2 Road Network

The workspace with its associated rules $\mathcal{R}$ contains the roadway of the road network the car operates in, which is a planar compact connected region, and the rules of the road associated with the roadway. We assume that $\mathcal{R} \subseteq \mathcal{Z}$, i.e., that we can express the road, which follows naturally, and its rules in terms of state space constraints on the ego-car. In Chapter 4, we will elaborate in details on rules we consider and how they may be implemented.

3.1.3 Traffic

The road network associated with the ego-car is also populated by other traffic participants, e.g., pedestrians, bikes, and cars. For brevity, we only consider other cars. We denote the state of car i present on the road $\mathcal{R}$ at time k by x_k^i, $i \in \{1, \ldots, N(k)\}$,

where $N(k)$ is the number of cars in $\mathcal{R}$ at time k. We consider a traffic model $\mathcal{T} = (\mathcal{V}(0), \mathfrak{I}, \mathfrak{O}, S)$, where $\mathcal{V}(0) = \{V^i\}_{i=1}^{N(0)}$ is the set of vehicles present in $\mathcal{R}$ at initial time $k = 0$, $\mathfrak{I} \subset \mathcal{Z}$ and $\mathfrak{O} \subset \mathcal{Z}$ are sets of states for entering and leaving the road network, and S is a scheduler that generates cars at $\mathfrak{I}$ and destroys them at $\mathfrak{O}$, see Section 3.5 for an example in the form of a hybrid system. Similar to the ego-car, we denote by $\mathcal{B}^i(x_k^i; \mathcal{R}) \subset \mathcal{Z}$ the footprint of traffic car i at time k according to the rule set $\mathcal{R}$. We note that making the footprint dependent on the rule set allows for including rules, such as varying safety margins, in the footprints of the traffic cars.

3.1.4 Controllers and Driving Behaviors

The behaviors of all the cars are defined by controllers (feedback or open-loop). Formally, a controller for car i is a map from all the car states $x_k = [z_k, x_k^{1:N}] \in \mathcal{X} \subseteq \mathbb{R}^n$ to a control value u_k^i, i.e., $C^i : \mathcal{X} \to \mathcal{U}$ such that $x_{k+1}^i = f_i(x_k^i, C^i(x_k))$, $\forall i \in \{1, \ldots, N\}$, where f_i defines the dynamics of car i. Similarly for the ego-car, we let $C : \mathcal{X} \to \mathcal{U}$ denote the controller of the ego-car such that $z_{k+1} = f_{ego}(z_k, C(x_k))$. $\mathcal{X}$ hereby denotes the joint state-space of all the cars and x_k^i denotes the state of car i at time k, as before. Throughout the paper, we will tacitly assume that the other cars' models are given together with the controllers that define their behavior and are known a priori for verification.

3.1.5 Safety

The controller for the ego-car is said to be *safe* at time k if it does not collide with environment obstacles, the road boundary, or other vehicles, and if it respects the rules of the road $\mathcal{R}$. To this end, let $\neg\mathcal{R}$ denote the complement of the road with its associated rules, and let $\mathcal{B}^i(x_k^i; \mathcal{R}) \oplus \mathcal{B}(Z_k)$ denote the Minkowski sum ("$\oplus$") between the footprint $\mathcal{B}^i(x_k^i; \mathcal{R})$ of car i and the ego-car's footprint $\mathcal{B}(Z_k)$ at time k. Moreover, we note that a desired safety margin, i.e., minimum distance between the ego-car and other vehicles, can be considered to be part of the rules $\mathcal{R}$ and thus appropriately included for in the footprint $\mathcal{B}^i(x_k^i; \mathcal{R})$ of the other cars. Then, the ego-car is said to

be safe at time k for a set $Z_k \subseteq \mathcal{Z}$ of potential states if

$$\inf_{z \in Z_k, o \in \mathcal{O}_k} \|z - o\| > 0,$$

where

$$\mathcal{O}_k = \neg\mathcal{R} \cup \bigcup_{i=1}^{N} \mathcal{B}^i(x_k^i; \mathcal{R}) \oplus \mathcal{B}(Z_k)$$

denotes the set of unsafe states. Similarly, the ego-car is *safe* on $\{0, \ldots, T\}$, $T \in \mathbb{N} \cup \{0, \infty\}$, if it is safe for all times $k \in \{0, \ldots, T\}$. The set of constraints representing safety are hereby said to be the *controller contract* $\mathcal{C}$, which is synthesized during the verification procedure as explained in Section 3.2.3.

The problem that we address in this paper is checking the *safety* of executing a controller C on the ego-car with respect to given car, road network, and traffic models. The controller C is hereby represented by the controller contract $\mathcal{C}$, which it is supposed to enforce and which is synthesized during the verification procedure. This not only allows for an abstract representation of specific controllers but also enables the concurrent verification of a broad class of controllers.

Problem 1 (Controller safety)**.** *Given an ego-car model V operating in road network $\mathcal{R}$, using controller C that abides by the controller contract $\mathcal{C}$, under the assumption of a traffic model $\mathcal{T}$, determine whether the ego-car is* safe *under control of C in the time interval $\{0, \ldots, T\}$ starting from some subset of the initial states $\mathfrak{I} \subseteq \mathcal{Z}$.*

3.2 Methods

In this section, we propose a verification framework based on the decomposition of the problem into smaller verification tasks corresponding to topological features of road networks, mainly road segments and intersections. Verification over entire networks is achieved by composition of models using synthesized assume-guarantee contracts.

The framework has two steps:

Table 3.1: Symbols table.

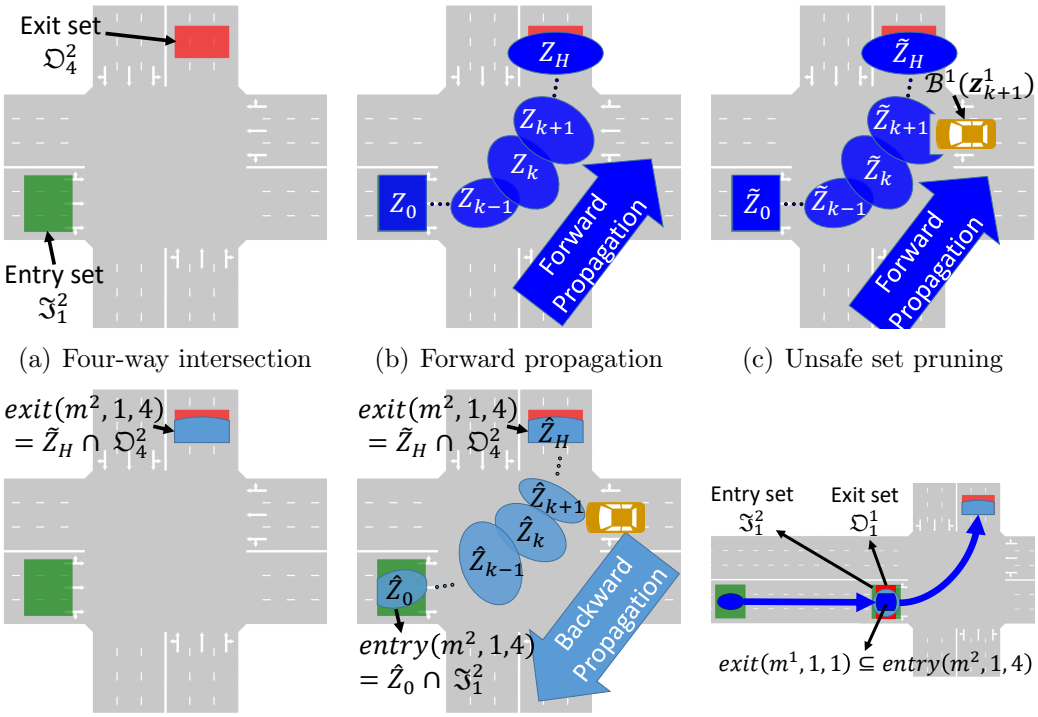
V	vehicle model
$\mathcal{Z}$, $\mathcal{U}$, $\mathcal{Q}$	state space, control space, configuration space
$\mathcal{R}$	road segment and associated rules
$f_{ego}(\cdot)$	ego-car dynamics
$g_p(\cdot)$, $g_q(\cdot)$	observation, configuration map
z_k, Z_k	state and set of states, respectively, of the ego-car at time k
x_k^i	state of car i at time k
p_k, q_k	position and pose, respectively, of the ego-car at time k
t_k, h, T	time at k, timestep, final time
$\mathcal{B}(z_k)$, $\mathcal{B}(Z_k)$	footprint of the ego-car for state z_k and the set Z_k, respectively
$\mathcal{B}^i(x_k^i; \mathcal{R})$	footprint of car i at state x_k^i
$\mathcal{O}_k$	combined footprint of static and dynamic obstacles at time k
$\mathcal{V}(k)$, $N(k)$	set and number of cars, respectively, present in $\mathcal{R}$ at time k
$\mathcal{T}$, S	traffic model, traffic scheduler
$\mathfrak{I}, \mathfrak{O}$	regions where cars may enter and exit a road model
C^i	controller of car i
$F(t_k; z_0)$	reachable set of the ego-car at time t_k starting from state z_0
$F(t_k; Z_0)$	reachable set of the ego-car at time t_k starting from the set Z_0
$F^\dagger(t_k; z_0)$	backward reachable set at time $T - t_k$ starting from state z_T
$F^\dagger(t_k; Z_0)$	backward reachable set at time $T - t_k$ starting from the set Z_T
$\widetilde{Z}_k$, $\widehat{Z}_k$	safe forward reachable of ego-car at time k
$\widehat{Z}_k$	safe backward reachable set of ego-car at time k
$G = (I, R)$	topological graph of the road network
$\mathcal{M}$	set of verified road models
m, A	road model and associated parameters
$\mathcal{C}, (entry, exit)$	controller contract, pair of assume-guarantee contracts

1. local verification and synthesis of the controller contract $\mathcal{C}$ with additional synthesis of assume-guarantee contracts, and

2. fitting of local models and composition with the assume-guarantee contracts.

In the first part, the parameterized local road models are verified and $\mathcal{C}$ is synthesized. At the same time, a pair of *safe* entry and exit states is synthesized for each model, which forms the associated *assume-guarantee contract.*These tasks may be performed off-line in parallel. The collection of local models is called a *library.*

In the second part, given a road network, locally verified models are fitted to the roads and intersections of the network. The assume-guarantee contracts are used to check the composition of the models based on the topology of the road network.



(a) Four-way intersection (b) Forward propagation (c) Unsafe set pruning

(d) Safe exit set (e) Backward propagation (f) Composition

Figure 3-1: Consider the library $\mathcal{M} = \{m^1, m^2\}$ composed of a straight road m^1 and a four-way intersection m^2. (a) The four-way intersection m^2 is shown. (b) The entry set is propagated forward, and, (c), concurrently pruned of unsafe states induced by other cars. (d) The safe exit set is the intersection of the safe reachable set at time step H, the verification horizon, and the exit set. (e) The safe entry set is computed by backward propagating the safe exit set. (f) Lastly, the composition of the models using the associated assume-guarantee contracts enables us to certify road networks.

3.2.1 Library of Parameterized Models

The verification process is decomposed into smaller local problems to enable computational tractability as well as facilitate parallel and distributed solutions. Moreover, we want to reuse the local computations both within and across road networks. Thus, we propose constructing a library of parameterized models that can be verified a priori and used in any road network to decide the safety of a controller. Local verification of models is valid for *any* controller that implements the controller contract $\mathcal{C}$.

We let $2^{\mathcal{Z}}$ denote the power set of $\mathcal{Z}$. Formally, each road element model is a tuple $m = (\mathcal{R}, \mathfrak{I}, \mathfrak{O}, S, A)$, where $\mathcal{R}$ is the road with its associated rules as before, $\mathfrak{I} = \{\mathfrak{I}_j\}_{j=1}^{n_I} \subset 2^{\mathcal{Z}}$ is the set of n_I possible entry regions, $\mathfrak{O} = \{\mathfrak{O}_j\}_{j=1}^{n_O} \subset 2^{\mathcal{Z}}$ is the

set of n_O possible exit regions, S is the traffic scheduler that dictates when cars are generated into $\mathcal{R}$, and A is the set of parameters associated with the model. The parameters of a road model can be related to its geometry, such as the width of the lanes, the angles of an intersection's branches, and the pose of the model within a global road network. Note that the safety guarantees do not always hold for all possible values of a parameter (e.g. lane width), which implies that the verification procedure has to be conducted for a (finite) set of potential parameter values. The *library* of all available road element models is denoted by $\mathcal{M} = \{m^i\}_i$, where upper indices are used to distinguish between multiple road models if necessary. An example of a four-way intersection is shown in Figure 3-1(a). Composition of road models is done such that the exit region of the current model overlaps with the entry region of the next one, see Figure 3-1(f) for an example.

3.2.2 Reachability Analysis

Consider the dynamics model for the ego-car, $z_{k+1} = f_{ego}(z_k, u)$, where $z \in \mathcal{Z}$ is the state, and $u \in \mathcal{U}$ is the control input, as previously described. Then, let $z_k(z_0, u(\cdot)) \in \mathbb{R}^{n_{ego}}$ explicitly denote its state at time k when starting from the initial state z_0 evolving under control $u : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}^{n_u}$. Then, the reachable set $F(t_k; z_0) \subseteq \mathcal{Z}$ at time t_k under all possible controls $u \in \mathcal{U}$ starting from the state z_0 is defined as

$$F(t_k; z_0) = \{z_k(z_0, u(\cdot)) \mid u(t_{k'}) \in \mathcal{U}, \forall k' \in \{0, \ldots, k\}\}.$$

Since we are interested in the reachable set from a set of initial conditions, rather than a fixed initial condition, we define the reachable set $F(t_k; Z_0)$ at time t_k from a set of initial conditions $Z_0 \subseteq \mathcal{Z}$ as follows:

$$F(t_k; Z_0) = \bigcup_{z_0 \in Z_0} F(t_k; z_0).$$

We note that in practice we usually cannot compute the reachable set $F(t_k; Z_0)$ exactly. Thus, $F(t_k; Z_0)$ subsequently refers to an overapproximation of the true reach-

able set.

In addition to the forward reachable set $F(t_k; Z_0)$, we also consider the backward reachable set for the purpose of safety verification. To this end, let $f^{\dagger}_{ego}$ denote the inverse dynamics of the ego-car, such that $z_{k-1} = f^{\dagger}_{ego}(z_k, u_{k-1})$. Then, let $z_{-k}(z_T, u(\cdot)) \in \mathbb{R}^{n_{ego}}$ denote the state at time $T - hk = T - t_k$ when starting from the (final) state z_T evolving under control $u : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}^{n_u}$. Similar to above, let the backwards reachable set $F^{\dagger}(t_k; z_T) \subseteq \mathcal{Z}$ at time $T - t_k$ under all possible controls $u \in \mathcal{U}$ starting from the state z_T be as follows:

$$F^{\dagger}(t_k; z_T) = \{z_{-k}(z_T, u(\cdot)) \mid u(t_{k'}) \in \mathcal{U}, \forall k' \in \{k, \ldots, T\}\}.$$

Finally, we define the backwards reachable set $F^{\dagger}(t_k; Z_T)$ from a set $Z_T \subseteq \mathcal{Z}$ as

$$F^{\dagger}(t_k; Z_T) = \bigcup_{z_T \in Z_T} F^{\dagger}(t_k; z_T).$$

We leverage reachability analysis to compute the set of safe reachable states $\widehat{Z}_k$ at time k as explained below.

3.2.3 Verification of Controller Contracts

Verification of the local road models in the *library* is based on reachability analysis, which enables the concurrent synthesis of the controller contract $\mathcal{C}$ and the assume-guarantee contracts, which are used to obtain safety guarantees over road networks.

An assume-guarantee contract of a road model m, traversed from entry region j to exit region j', is a pair of safe entry and exit sets $(entry(m, j, j'), exit(m, j, j'))$, where $entry(m, j, j') \subseteq \mathfrak{I}_j \subset \mathcal{Z}$ and $exit(m, j, j') \subseteq \mathfrak{O}_{j'} \subset \mathcal{Z}$. The contract is interpreted as follows: if the system starts in $entry(m, j, j')$, then it is guaranteed that the controller can drive the ego-car safely to $exit(m, j, j')$ within the exit region $\mathfrak{O}_{j'}$. The assume-guarantee contract may be inferred from the controller contract $\mathcal{C}$, as described in Algorithm 1. The controller contract $\mathcal{C}$ takes the form of state space constraints, where $\mathcal{C}_k$ denotes the set of constraints at time k, see below for more details.

Algorithm 1 VERIFYROADMODELS

Input: $m = (\mathcal{Z}, \mathfrak{I}, \mathfrak{O}, S, A)$: road model, $\mathfrak{I}_j \in \mathfrak{I}$, $\mathfrak{O}_{j'} \in \mathfrak{O}$: pair of entry/exit states, T: verification horizon

Output: $\mathcal{C}$: controller contract, $entry(m, j, j')$, $exit(m, j, j')$: pair of safe entry/exit regions (assume-guarantee contract)

1: $\widetilde{Z}_0 \leftarrow \mathfrak{I}_j$
2: $\{x_0^i\}_{i=1}^{N(0)} \leftarrow S.init()$
3: **for all** $k \in \{0, \ldots, T-1\}$ **do** ▷ forward propagation
4: $\quad x_{k+1}^i \leftarrow f_i(x_k^i, C^i(\widetilde{Z}_k, x_k^{1:N(k)}))$, $\forall i \in \{1, \ldots, N(k)\}$
5: $\quad N(k+1) \leftarrow S.update()$
6: $\quad Z_{k+1} \leftarrow F(h; \widetilde{Z}_k)$
7: $\quad \mathcal{O}_{k+1} = \neg\mathcal{R} \cup \bigcup_{i=1}^{N} \mathcal{B}^i(x_{k+1}^i; \mathcal{R}) \oplus \mathcal{B}(Z_{k+1})$
8: $\quad \widetilde{Z}_{k+1} \leftarrow Z_{k+1} \setminus \mathcal{O}_{k+1}$ ▷ enforcing safe behavior
9: $\quad \widetilde{Z}_{k+1} \leftarrow$ REDUCESETCOMPLEXITY$(\widetilde{Z}_{k+1})$ ▷ simplify the set representation
10: $exit(m, j, j') \leftarrow \widetilde{Z}_T \cap \mathfrak{O}_{j'}$
11:
12: $\widehat{Z}_T \leftarrow exit(m, j, j')$
13: **for all** $k \in \{T, \ldots, 1\}$ **do** ▷ backward propagation
14: $\quad \widehat{Z}_{k-1} \leftarrow F^{\dagger}(h; \widehat{Z}_k) \cap \widetilde{Z}_{k-1}$
15: $\quad \widehat{Z}_{k-1} \leftarrow$ REDUCESETCOMPLEXITY$(\widehat{Z}_{k-1})$ ▷ simplify the set representation
16: $\mathcal{C} \leftarrow \{\widehat{Z}_k\}_{k=0}^{T}$ ▷ assign the safe reachable sets to the contract
17: $entry(m, j, j') \leftarrow \widehat{Z}_0 \cap \mathfrak{I}_j$
18:
19: **return** $\mathcal{C}$, $entry(m, j, j')$, $exit(m, j, j')$

The verification procedure for a road model m with given entry and exit regions is shown in Algorithm 1 and can be considered as a type of reach-avoid problem. The objective here is to find the controller contract $\mathcal{C}$ by computing the safe reachable set $\widehat{Z}_k$ for the ego-car at all time steps such that it preserves the rules associated with $\mathcal{R}$ and is collision-free. Concurrently, we are able to synthesize the assume-guarantee contracts for finding the safe entry/exit pairs. Overall, all entry-exit pairs need to be verified. The algorithm has two main components.

First, the entry set is propagated forward over the given time horizon T, starting from the initial entry set $\mathfrak{I}_j$ (line 1). The initial states of the other cars are initialized by the scheduler using its $init()$ method (line 2). At each step the other vehicles' states

are propagated using their controllers C^i (line 4). Next, the scheduler's *update*() method is called (line 5) that spawns and removes vehicles, initializes new vehicles, and returns the number of vehicles according to the traffic model $\mathcal{T}$. Then, the ego-car's reachable set Z_{k+1} is computed (line 6). The unsafe states $\mathcal{O}_{k+1}$ are the union over the road complement $\neg\mathcal{R}$ and the union of possible footprints $\mathcal{B}^i(x^i_{k+1};\mathcal{R}) \oplus \mathcal{B}(Z_{k+1})$ of each of the other cars with the ego-car (line 7). The unsafe states $\mathcal{O}_{k+1}$ are pruned from Z_{k+1} (line 8) to obtain the *safe*, forward reachable set $\widetilde{Z}_{k+1}$. Note that the reachable set Z_{k+1} is computed over all possible control inputs $u \in \mathcal{U}$, and only afterwards (line 8) do we prune the set of unsafe states to obtain the safe, forward reachable set $\widetilde{Z}_k$ at that time step. In other words, we check for all solutions Z_{k+1} and then prune them accordingly to obtain the possible solutions $\widetilde{Z}_{k+1}$ considering the obstacles and the road with its associated rules $\mathcal{R}$. We also perform an additional step, which is explained in detail in Section 3.4.4, to ensure the set representation remains computationally tractable (line 9). The safe exit set $exit(m, j, j')$ is the set of safe reachable solutions within the exit set $\mathfrak{O}_{j'}$ (line 10).

Second, we compute the safe entry set for $exit(m, j, j') \subset \mathfrak{O}_{j'}$. and the controller contract $\mathcal{C}$. To this end, we employ backward propagation from the safe exit set (line 12). The safe exit set is backpropagated via the inverse dynamics of the ego-car and intersected with the safe forward reachable set $\widetilde{Z}_{k-1}$ since these are the only relevant solutions (line 14). The rules of the road $\mathcal{R}$ are enforced at all times since $\widetilde{Z}_{k-1}$ abides by them. Then, we can infer the controller contract $\mathcal{C}$ (line 16) and the safe entry set (line 17) from the safe reachable sets $\{\widehat{Z}_k\}_{k=0}^T$.

The necessity for the backpropagation step arises from the fact that, although we can infer the set of initial conditions ($\mathfrak{I}_j$) of the safe reachable set $\widetilde{Z}_T$, we cannot infer the set of *safe* initial conditions $entry(m, j, j')$ of the safe exit set $exit(m, j, j')$. We also require backpropagation to obtain a sound contract $\mathcal{C}$ since otherwise we are not guaranteed that the constraint set $\mathcal{C}_{k+1}$ at time $k+1$ is reachable from the constraint set $\mathcal{C}_k$ at the previous time k.

A graphic representation of the procedure is shown in Figure 3-1. The forward propagation procedure (line 6) is shown in 3-1(b), and the pruning step (line 8) is

shown in 3-1(c). Once the safe-reachable set at step T is computed, it is trimmed (line 10) to lie within the exit region, see 3-1(d). The second part of the procedure, the backward propagation (line 12-17) shown in 3-1(e), computes the safe entry set and the controller contract $\mathcal{C}$.

Algorithm 2 VERIFYFROMLIBRARY

Input: $V = (\mathcal{Z}, \mathcal{R}, \mathcal{U}, f_{ego}, g_p, g_q)$, $\mathcal{M}$: library of verified parameterized models
Output: Boolean value indicating *safety*

1: Extract topology graph $G = (I, R)$ of road network $\mathcal{R}$
2: Fit each node $\iota \in I$ (intersection) to $m^\iota \in \mathcal{M}$ with parameters A^ι
3: Fit each edge $r \in R$ (road) to $m^r \in \mathcal{M}$ with parameters A^r
4: **for all** $r_1 = (\iota_1, \iota_2), r_2 = (\iota_2, \iota_3) \in R$ **do**
5: **if** $\neg\big(exit(m^{r_1}, 1, 1) \subseteq entry(m^{\iota_2}, r_1, r_2) \wedge exit(m^{\iota_2}, r_1, r_2) \subseteq entry(m^{r_2}, 1, 1)\big)$ **then**
6: **return** $\bot$
7: **return** $\top$

3.2.4 Road Network Verification

Given a library of verified road models $\mathcal{M}$, we can verify road networks via composition using the models' assume-guarantee contracts. The procedure is summarized in Algorithm 2. First, we extract the topology graph G of the network $\mathcal{R}$ (line 1), and then fit models to all intersections $\iota \in I$ (line 2) and road segments $r \in R$ (line 3) corresponding to the graph's nodes and edges. Finally, we check for each two incident road segments $r_1, r_2 \in R$ if: (a) the safe exit set $exit(m^{r_1}, 1, 1)$ of the incoming road r_1 is included in the safe entry set $entry(m^{\iota_2}, r_1, r_2)$ of the common intersection m^{ι_2}, and (b) the safe exit set of the intersection $exit(m^{\iota_2}, r_1, r_2)$ is included in the safe entry set $entry(m^{r_2}, 1, 1)$ of the outgoing road r_2. If all checks pass, then the network is certified safe. In other words, if the safe set remains non-empty during the propagation throughout the network, the network is certified safe. In the case that a (pairwise) composition is deemed unsafe, Algorithm 1 can be run for the composition. Subsequently, Algorithm 2 can be rerun with the composition as an additional road model.

3.3 Analysis

In this section, we provide results on safety guarantees over paths in the road network provided that Algorithm 2 returns true.

Let $m^i \in \mathcal{M}$ be a local road model and j, j' the indices of a pair of entry and exit regions of m^i, respectively. We denote by $P^i_{j,j'} : entry(m^i, j, j') \to exit(m^i, j, j')$ the surjective map that propagates the states in the safe entry set to the safe exit set in model m^i from j to j'. Let G be the topology graph of a road network. Let $\boldsymbol{\iota} = (\iota_1, \ldots, \iota_p)$ be a path in G, and let $\boldsymbol{m} = (m^{r_1}, m^{\iota_2}, m^{r_2}, \ldots, m^{r_{p-1}})$ denote the sequence of road and intersection models traversed by $\boldsymbol{\iota}$, where $r_i = (\iota_i, \iota_i + 1) \in R$. The propagation map over the path $\boldsymbol{\iota}$ is $P^{\boldsymbol{\iota}} = P^{r_{p-1}}_{1,1} \circ P^{\iota_{p-1}}_{r_{p-2},r_{p-1}} \circ \ldots \circ P^{\iota_2}_{r_1,r_2} \circ P^{r_1}_{1,1}$, where $\circ$ denotes a function composition.

Theorem 1. Let $V = (\mathcal{Z}, \mathcal{R}, \mathcal{U}, f_{ego}, g_p, g_q)$ be an ego-car implementing a controller C satisfying controller contract $\mathcal{C}$ and $G = (I, R)$ the topology graph of road network $\mathcal{R}$. Let $\mathcal{M}$ be a library of road models verified against controller contract $\mathcal{C}$. If Algorithm 2 returns true, then for all paths $\boldsymbol{\iota} = (\iota_1, \ldots, \iota_p)$ in G the ego-car V executing C from a state z_0 in the safe entry set $entry(m^{r_1}, j, j')$ of road $r_1 = (\iota_1, \iota_2)$ is *safe* over $\boldsymbol{\iota}$.

Proof. Let $\boldsymbol{\iota} = (\iota_1, \ldots, \iota_p)$ be a path in G and $\boldsymbol{m} = (m^{r_1}, m^{\iota_2}, m^{r_2}, \ldots, m^{r_{p-1}})$ be the corresponding sequence of road and intersection models traversed by $\boldsymbol{\iota}$. Note that since $m^{\iota_i}, m^{r_i} \in \mathcal{M}$, $\forall i \in \{1, \ldots, p\}$, it follows that we have $P^{r_i}_{1,1}(entry(m^{r_i}, 1, 1)) \neq \emptyset$ and $P^{\iota_i}_{r_{i-1},r_i}(entry(m^{\iota_i}, r_{i-1}, r_i)) \neq \emptyset$, $\forall i \in \{1, \ldots, p\}$. Thus, $P^{\boldsymbol{\iota}}$ over $\boldsymbol{\iota}$ also maps the safe start set of m^{r_1} to a non-empty set. We are guaranteed that for every start state in $z_0 \in entry(m^{r_1}, 1, 1)$ a safe exit state $P^{\boldsymbol{\iota}}(z_0) \in exit(r^{p-1}, 1, 1)$ is reached. □

The next result shows that if we assume that local road models in $\mathcal{M}$ satisfy a robustness property, we then obtain robustness guarantees over paths in the road network. Informally, the robustness condition on the local models is a lower bound on the volume contraction of propagated sets. Larger volumes indicate higher robustness to perturbation.

Corollary 1 (Robustness)**.** Consider the same assumptions as in Theorem 1. If in addition for all $m^i \in \mathcal{M}$ and j, j' there exists $q > 0$ such that for all $S^i \subseteq entry(m^i, j, j')$ it holds that $\mu(E^i) > q\,\mu(S^i)$ with $E^i = P^i_{j,j'}(S^i) \subseteq exit(m^i, j, j')$, then $\mu(P^{\boldsymbol{\iota}}(Z_0)) > q^{2(p-1)}\,\mu(Z_0)$, where $\boldsymbol{\iota} = (\iota_1, \ldots, \iota_p)$, $Z_0 = entry(m^{r_1}, 1, 1)$, and μ is the Lebesgue measure.

Proof. Let $\boldsymbol{\iota} = (\iota_1, \ldots, \iota_p)$ be a path in G and $P^{\boldsymbol{\iota}}$ its propagation map. The robustness property of the local models implies that $\mu(P^{\boldsymbol{\iota}}(Z_0)) = \mu(P^{r_{p-1}}_{1,1}(P^{\iota_{p-1}}_{r_{p-2},r_{p-1}}(P^{\boldsymbol{\iota}'}(Z_0)))) \geq q^2\mu(P^{\boldsymbol{\iota}'}(Z_0))$, where path $\boldsymbol{\iota}' = (\iota_1, \ldots, \iota_{p-1})$. By induction, we obtain the desired bound $\mu(P^{\boldsymbol{\iota}}(Z_0)) > q^{2(p-1)}\,\mu(Z_0)$. □

3.4 Implementation

In the following, we introduce the tools for reachability analysis and set operations. Relevant details for the implementation of Algorithm 1 are mentioned as well.

3.4.1 Reachability Tool

We employ CORA [2] to compute an overapproximation $F(t_k; \cdot) \subseteq \mathcal{Z}$ of the reachable set for each time k (line 6 of Algorithm 1). CORA is a reachability tool for linear, nonlinear, and hybrid dynamical systems. Starting from some set $\widetilde{Z}_k$, represented as zonotope, it computes the reachable set Z_{k+1} at time $k+1$ by solving the underlying differential equation for the midpoint and the generators that describe the zonotope, see [2] for more details. Subsequently, Z_{k+1} is appropriately inflated to account for the variable control input $u \in \mathcal{U}$. Nonlinear systems, such as the dynamics model used in our experiments in Section 3.5, are abstracted to polynomial systems and the abstraction error is accounted for through an additional inflation of Z_{k+1} to obtain an overapproximation.

3.4.2 Set Representation

The road segments $\mathcal{R}$ and the collision constraints $\mathcal{O}$ are described via polytopes. Since polytopes, as opposed to zonotopes, are closed under intersection, they can be used to prune unsafe solutions from the reachable set. We use the MPT toolbox [34] for polytope operations. To be able to convert sets back between zonotope (for reachability) and polytope (for other purposes) representation, we use CORA to compute an overapproximation, i.e., an encompassing convex hull of the polytope. Note that non-convex regions are stored as an array of convex regions. Thus, any error caused by the conversion can be neglected.

3.4.3 Set Pruning

As expressed in Algorithm 1, we check for unsafe states and prune solutions that are in collision with either road boundaries or other cars, which generally yields non-convex sets (line 8). These operations are performed using polytopic representations. Note that any non-convex safe reachable set

$$\widetilde{Z}_k = \bigcup_{\ell=1}^{L_k} \widetilde{Z}_k^\ell$$

is segmented into L_k convex regions $\widetilde{Z}_k^\ell$ and stored accordingly. This is crucial as CORA requires convex input sets. The next reachable set is then computed as

$$Z_{k+1} = \bigcup_{\ell=1}^{L_k} F(h; \widetilde{Z}_k^\ell)$$

applying CORA separately to each convex region $\widetilde{Z}_k^\ell$, see Figure 3-4 for an example.

3.4.4 Reduction of Complexity

Representing sets as an array of convex sets, however, leads to an exponential growth in the number of convex segments L_k, i.e., in the cardinality $\left|\widetilde{Z}_k\right|$, because:

Algorithm 3 ReduceSetComplexity

Input: $\widetilde{Z}_k = \bigcup_{\ell=1}^{L_k} \widetilde{Z}_k^\ell$: an array of convex sets
Output: $\widetilde{Z}_k'$: an array of convex sets such that $\left|\widetilde{Z}_k'\right| \ll \left|\widetilde{Z}_k\right|$

1: $L_k \leftarrow \left|\widetilde{Z}_k\right|$
2: $\mu_{max} \leftarrow \max_{\ell \in \{0,\dots,L_k\}} \mu(\widetilde{Z}_k^\ell)$ ▷ store maximum volume from the array
3: $\widetilde{Z}_k' \leftarrow \emptyset$ ▷ initialize the new array
4: **for all** $\ell \in \{1, \dots, L_k\}$ **do**
5: **if** $\mu(\widetilde{Z}_k^\ell) \geq \varepsilon \mu_{max}$ **then** ▷ only consider large sets, where $\varepsilon \ll 1$
6: $\widetilde{Z}_k' \leftarrow \{\widetilde{Z}_{k'}, \widetilde{Z}_k^\ell\}$ ▷ append the convex set to the new array
7: $\widetilde{Z}_k' \leftarrow$ RemoveOverlaps$(\widetilde{Z}_k')$ ▷ remove overlapping sets
8: **return** $\widetilde{Z}_k'$

1. each pruning operation $Z_k^\ell \setminus \mathcal{O}_k$ can lead to a split into more convex regions, and

2. the reachability tool splits large partial sets $\widetilde{Z}_k^\ell$ into further subregions to minimize the error associated with the underlying approximation procedure.

Exponential growth in L_k will inevitably yield exponential growth in runtime as we must compute the reachable set individually for each of the L_k subsets. Thus, this calls for an efficient and effective method to reduce the cardinality of $\widetilde{Z}_k$ in order for the verification procedure to be computationally tractable.

The method to reduce the cardinality of $\widetilde{Z}_k$ is shown in Algorithm 3 and is called from Algorithm 1, Lines 9 and 15 during the verification procedure at each time k. In particular, Algorithm 3 is based on two key insights:

1. due to the constraint checks during each time k, many subsets $\widetilde{Z}_k^\ell$ are reduced to a negligible size, e.g. subsets that overlap the roadway, i.e., $Z_k^\ell \cap \neg\mathcal{R} \neq \emptyset$;

2. neighboring subsets $\widetilde{Z}_k^\ell$, $\widetilde{Z}_k^{\ell'}$ that will be propagated separately are likely to overlap at time $k+1$, i.e., $\widetilde{Z}_{k+1,\ell} \cap \widetilde{Z}_{k+1,\ell'} \neq \emptyset$.

We leverage these insights in Algorithm 3 as follows. First, we compute the maximum volume among all convex subsets $\widetilde{Z}_k^\ell$, which we use to prune convex subsets

Algorithm 4 RemoveOverlaps

Input: $Z = \bigcup_{\ell=1}^{L} Z^\ell$: an array of convex sets
Output: Z': an array of convex sets such that $|Z'| \ll |Z|$

1: $L \leftarrow |Z|$
2: $Z' \leftarrow \emptyset$ ▷ initialize the new array
3: **if** $L = 1$ **then** ▷ no sets can be removed in this case
4: $\quad Z' \leftarrow Z$
5: **else if** $L = 2$ **then**
6: $\quad$ **if** $Z^1 \subseteq Z^2$ **then** ▷ check if one set is a subset of the other
7: $\quad\quad Z' \leftarrow Z^2$
8: $\quad$ **else if** $Z^2 \subseteq Z^1$ **then**
9: $\quad\quad Z' \leftarrow Z^1$
10: $\quad$ **else**
11: $\quad\quad Z' \leftarrow Z$
12: **else** ▷ run the algorithm recursively on smaller sets
13: $\quad Z'' \leftarrow \text{RemoveOverlaps}(\{Z^1, \ldots, Z^{\lceil L/2 \rceil}\})$
14: $\quad Z''' \leftarrow \text{RemoveOverlaps}(\{Z^{\lceil L/2 \rceil + 1}, \ldots, Z^L\})$
15: $\quad Z' \leftarrow \{Z'', Z'''\}$ ▷ consolidate the results
16: **return** $\widetilde{Z}'_k$

that have significantly smaller volume than the largest one (lines 4-6). Since the neglected sets have very small volume, the impact on the overall reachable set is insignificant. In the second step, we remove overlapping subsets using a divide and conquer approach (line 7).

The divide and conquer approach is shown in detail in Algorithm 4. In particular, Algorithm 4 is called recursively on subsets of smaller cardinality (lines 13-15). The base case is a simple pairwise check, whether one of the two convex sets is a subset of the other (lines 3-11). This allows to remove redundant sets form the array with a runtime of $\mathcal{O}(L_k \log L_k)$, where L_k denotes the cardinality of $\widetilde{Z}_k$ at time k. Moreover, we note that a pairwise check on the entire array, which would run in $\mathcal{O}(L_k^2)$, is actually quite unfeasible in practical settings, thus elicits the use of the divide and conquer approach.

Our empirical results show that the runtime can be significantly reduced (up to 99%) when Algorithm 3 is applied at the end of each time step during forward and

backward propagation. Finally, we note that the runtime is dependent on the number of vehicles since each additional obstacle increases the complexity of $\mathcal{O}_k$. Empirically, we found that up to 5-7 vehicles can be added before noticing significant increases in the runtime.

3.5 Results

In this section, we will instantiate the verification framework to verify a specific receding horizon controller, also referred to as model predictive controller (MPC), that implements the controller contract $\mathcal{C}$. In the following, we specify the dynamic motion model, the vehicle's dynamic limitations, the collision constraints from other vehicles, and the drivable space constraints. The constraints constitute the rules of the road $\mathcal{R}$. At the end of the section, the Manhattan road library and traffic model, which were verified in the experiments, together with the results are presented. Recall that the Minkowski sum is denoted by $\oplus$.

3.5.1 Dynamic Motion Model and Dynamic Constraints

We follow the nonlinear MPC formulation in [57] and employ a car model with a fixed rear wheel and a steerable front wheel with state z and controls u. At time k, we denote the state of the ego-vehicle, typically position $p_k = [x_k, y_k] \in \mathbb{R}^2$, linear velocity v_k, orientation θ_k, and steering angle δ_k, by $z_k = [p_k, \theta_k, \delta_k, v_k] \in \mathcal{Z} \subseteq \mathbb{R}^5$, and the configuration by $q_k = [p_k, \theta_k] \in \mathcal{Q} \subseteq SE(2)$. Its control input, typically steering velocity $\dot{\delta}_k$ and acceleration a_k, is labeled $u_k = [u_k^\delta, u_k^a] \in \mathcal{U} \subset \mathbb{R}^2$. The rear-wheel driven vehicle with inter-axle distance L and continuous kinematic model

$$\dot{z} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\delta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v}{L}\tan(\delta) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} u^\delta \\ u^a \end{bmatrix}}_{u}, \tag{3.1}$$

is described by a discrete time model by integration

$$z_{k+1} = z_k + \int_{kh}^{(k+1)h} \dot{z}\,\mathrm{d}t = f(z_k, u_k),$$

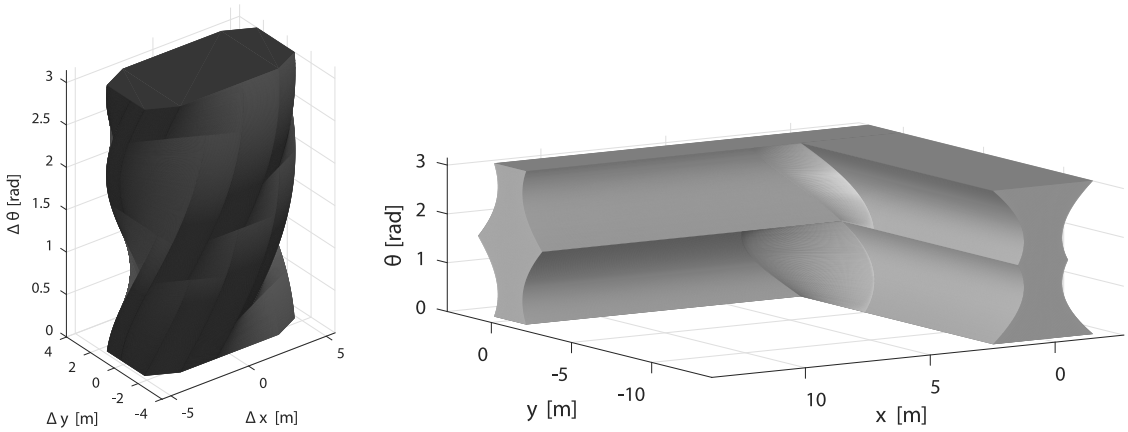
where h is the sampling period.

We limit the steering angle, $|\delta| \leq \delta_{\max}$, steering speed, $|u^\delta| \leq \dot{\delta}_{\max}$, longitudinal speed, $|v| \leq v_{\max}$, breaking and accelerations, $a_{\min} \leq u^a \leq a_{\max}$, such that they conform to the dynamical limitations and the rules of the road. The yaw-rate is limited to $|\dot{\theta}| \leq \dot{\theta}_{\max}$ allowing to neglect slip. The modification is in line with our main goal: driver safety. While our choice of motion model considers a more conservative yaw-rate constraint, the verification framework allows for straight-forward integration of more advanced motion models including slip and load-transfers. Uncertainty in the dynamical model may be accounted for through proper enlargement of the reachable set.

3.5.2 Other Vehicles

In the following, we will derive the representation of the safety constraints with respect to the other vehicles. To ensure real-time operation, motion planners frequently approximate their own or other cars' footprint $\mathcal{B}^i(x_k^i; \mathcal{R})$ by simpler geometries such as rotation invariant bounding boxes, enclosing ellipses, or polygons. For the reachability analysis, we approximate the shape of other vehicles by a polygon, enclosing the ellipse used by the MPC of [57]. Note that accounting for the ego-car's shape in the reachable set can become intractable due to the non-convex, disjoint nature of the set. We propose an approach where we instead compute the Minkowski sum of the other vehicle's polygon and the ego-car's rectangular shape for each possible difference in configuration $\Delta q_k^i = q_k - q_k^i$ to form a single representation of the collision region

$$\mathfrak{C}_k^i(z_k) = \mathcal{B}^i(x_k^i; \mathcal{R}) \oplus \mathcal{B}(z_k) \subset \mathbb{R}^2.$$



(a) Car-collision configurations (b) Drivable space

Figure 3-2: (a) The exact Minkowski swept volume between the ego-car and another vehicle, which represents configurations in collision, is shown. (b) Drivable space in a four-way intersection left-turn scenario obtained by taking the Minkowski sum between the rectangular ego-car and the road over Δq, i.e., $\mathfrak{D}(\mathcal{Z})$. This is used to trim the admissible state space. Because of symmetry and for brevity, θ is only shown for $[0, \pi]$. Approximations to 10 θ-slices are used in the implementation to reduce computational complexity.

The resulting volume $\mathfrak{C}_k^i(\mathcal{Z}) \subset \mathcal{Q}$ can be represented as a single, invariant shape in the Δq-space, where $\Delta q = q^i - q$, see Figure 3-2(a). This volume is translated and rotated according to each of the other vehicles' poses q_k^i to obtain the actual constraint in the configuration space of the ego-car. To reduce the computational complexity in the implementation, a coarser overapproximation compared to the one shown in Figure 3-2(a) is chosen.

3.5.3 Drivable Space

We take a similar approach to obtain the drivable space, where we compute the Minkowski sum between the ego-car's footprint and the non-road surface over all configurations q of the ego-car. To this end, let $\mathcal{R}'$ denote the road surface of the considered road segment $\mathcal{R}$. The complement of the aforementioned Minkowski sum constitutes the drivable space $\mathfrak{D}(z)$, i.e.,

$$\mathfrak{D}(z) = \neg \left(\neg \mathcal{R}' \oplus \mathcal{B}(z) \right) \subset \mathbb{R}^2.$$

The set $\mathfrak{D}(z)$ can then be lifted into a configuration volume $\mathfrak{D}(\mathcal{Z}) \subseteq \mathcal{Q}$ containing the allowed ego-configurations, see Figure 3-2(b). An underapproximation with fewer θ-slices is used in the implementation to reduce the computational effort.

3.5.4 Rules of the Road

As previously described, the road segment $\mathcal{R}$ contains the drivable space as well as rules associated with the road segment. Moreover, we assumed that $\mathcal{R} \subseteq \mathcal{Z}$, i.e., we can express the resulting rules as state space constraints. Below, we summarize the rules we consider:

- road Boundaries: $g_q(\widetilde{Z}_k) \subseteq \mathfrak{D}(\mathcal{Z})$, $\forall k$,
- safety (obstacle avoidance): $\mathcal{O}_k \cap \widetilde{Z}_k = \emptyset$, $\forall k$,
- speed limit (traffic rules): $|v| \leq v_{\max}$,
- dynamic limits: $|\delta| \leq \delta_{\max}$, $|u^\delta| \leq \dot{\delta}_{\max}$, $a_{\min} \leq u^a \leq a_{\max}$.

Note that all of these rules are expressed in terms of state space constraints as desired. In Section 4.2 we also consider a larger set of rules.

3.5.5 Admissible Configurations

The above constraints, i.e., the traffic cars (Section 3.5.2), the drivable space (Section 3.5.3), and the rules of the road (Section 3.5.4), are combined into a unified state space representation that can be used to prune the reachable set. Note that the representation is depended on the time k since the traffic cars are dynamic obstacles. To this end, we denote by $\mathfrak{A}_k \subset \mathcal{Q}$ the overall admissible configuration volume that is obtained by subtracting the other vehicles' swept volumes from the drivable configuration volume at time k, i.e.,

$$\mathfrak{A}_k = \mathfrak{D}(\mathcal{Z}) \setminus \bigcup_{i \in \{1,\ldots,N(k)\}} \mathfrak{C}_k^i(\mathcal{Z}),$$

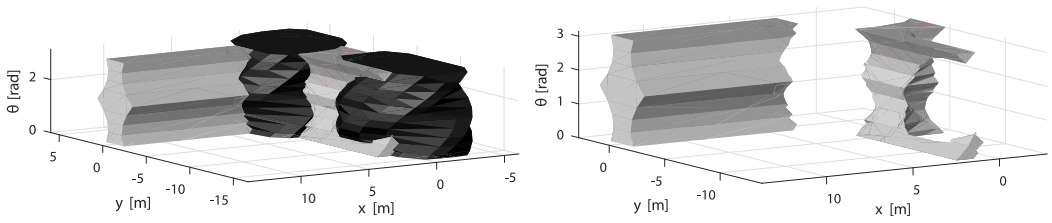


Figure 3-3: The dynamic obstacles (black) and the drivable space (grey) are shown for time k on the left. Set subtraction yields the admissible configuration volume $\mathfrak{A}_k$ (right).

see Figure 3-3. After lifting the admissible configuration volume into the state space $\mathcal{Z}$, we now have the unsafe states $\mathcal{O}_k = g_q^{-1}(\neg\mathfrak{A}_k)$ in a simple form. This can be further augmented through simple state space constraints to account for the speed limit and dynamic limits mentioned in Section 3.5.4. This representation enables us to efficiently compute the collision constraints once and to enforce them by simply applying a set difference operation to the forward propagated state set Z_k at each time k, as shown in Algorithm 1, Line 8.

3.5.6 Road Library and Network

We showcase the framework for a Manhattan-style library consisting of 17 intersections and 5 straight road segments. Roads can have one or more lanes and incoming roads into an intersection may be tilted with respect to each other, see Figure 3-6 for a selection of verified models. The library is used to verify the network shown in Figure 1-1, which consists of ca. 130 blocks in Mid-Manhattan with ca. 180 intersections and ca. 330 straight road segments.

3.5.7 Traffic Model

We describe other traffic participants as a hybrid system, i.e., a set of traffic flows. Each traffic flow is specified by a predefined path, including possible lane changes, connecting regions for entering ($\mathfrak{I}_j$) and leaving ($\mathfrak{O}_{j'}$) road model m, and a velocity profile, which together form a trajectory. The traffic scheduler S spawns vehicles at

some region $\mathfrak{I}_j$ with a fixed frequency ($0.1s^{-1}$ to $0.4s^{-1}$) [2] and removes them once they reach $\mathfrak{O}_{j'}$. Trajectories are defined as arc-length parameterized, continuously differentiable clothoid spline paths and velocity profiles generated by cubic Hermite spline interpolation. For simplicity, we restrict the class of controllers associated with the other traffic cars to open-loop. In Chapter 4, we extend the traffic model to consider probabilistic behaviors that are synthesized using a falsification approach.

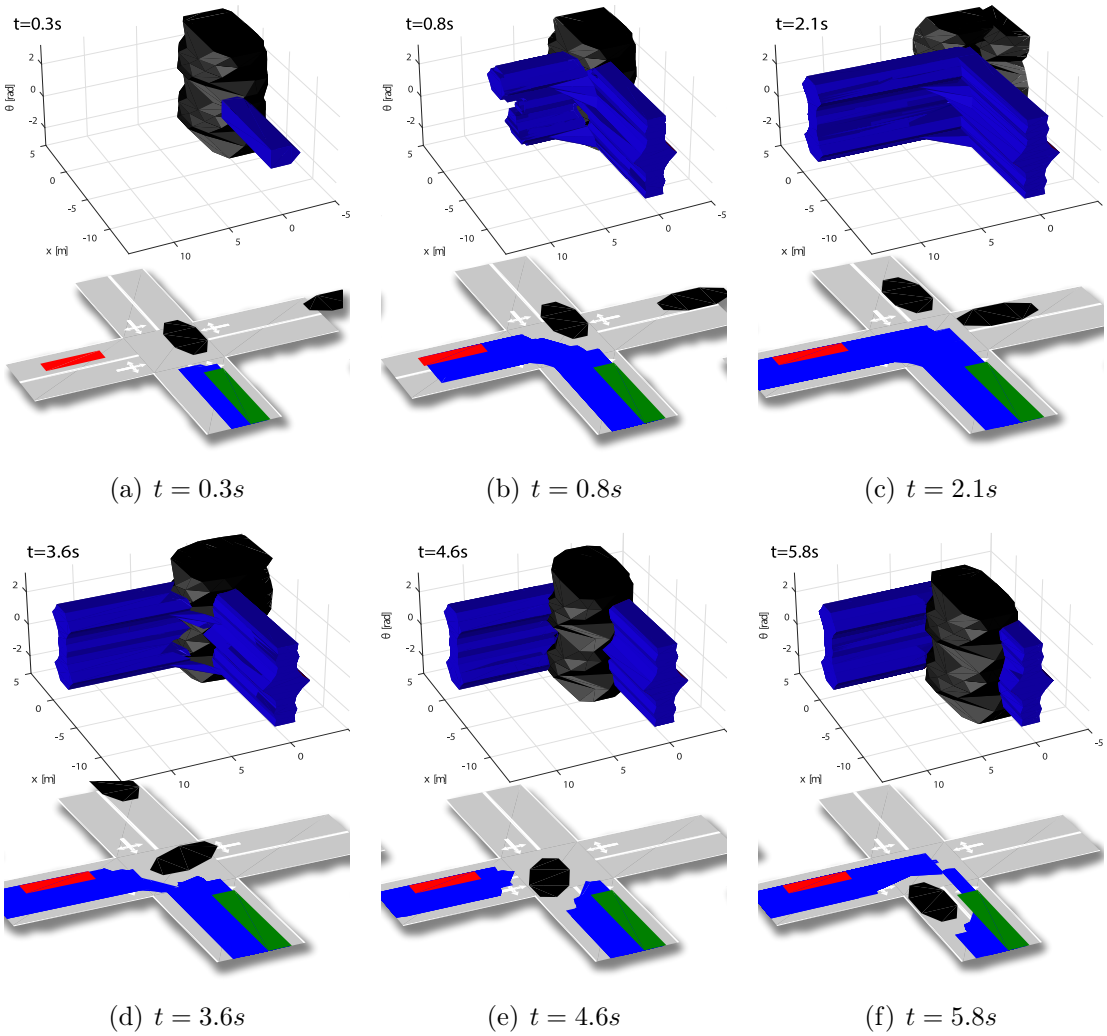
3.5.8 Experiments

The proposed procedure was successfully applied to the aforementioned library, including the four-way intersection m in Figure 3-4 where the verified left-turn maneuver is shown. The considered time horizon is $6.0s$ with a discrete timestep $h = 0.05s$ resulting in 120 iterations. In the shown example there are two streams of cars occupying the intersection, both of which the ego-car avoids. When the second car crosses the intersection all states in collision are pruned causing the contract $\mathcal{C}$ to become disjoint ($t = 3.6s, t = 4.6s$). However, as the ego-car controller has to abide by the contract $\mathcal{C}$ safety remains verified. Upon the exit of the second car, the safe reachable set is expanding again reflecting the fact that the intersection has become available ($t = 5.8s$).

As visible from Figure 3-4 the ego-car reaches the entire exit set, i.e., $exit(m, 1, 4) = \widetilde{Z}_T \cap \mathfrak{O}_4 = \mathfrak{O}_4$. Next, the safe entry set $entry(m, 1, 4)$ is computed by means of backward propagation, see Algorithm 1 for details. The contract $\mathcal{C}$ is also updated during that procedure. The results of the backward propagation are shown in Figure 3-5.

The remaining library was verified analogously. Note that for intersections, where there are multiple entry and exit regions, all combinations must be tested in order for the model to be deemed safe. This results in a total of 83 experiments for the 22 road models considered. In Figure 3-6, a selection of models with various geometries is shown together with the reachable set and traffic for some time step of the verification. Consequently, the library was matched with the topology graph of Mid-Manhattan, see Figure 1-1, and the compositions were tested for safety according to

[2] This range covers a large variety of situations, from occasional vehicles to dense car streams.



(a) $t = 0.3s$ (b) $t = 0.8s$ (c) $t = 2.1s$

(d) $t = 3.6s$ (e) $t = 4.6s$ (f) $t = 5.8s$

Figure 3-4: The forward propagation of the reachable set is shown for a left turning maneuver. Blue sets indicate the safely reachable configuration set $g_q(\widetilde{Z}_k)$ (top row), and the position set $g_p(\widetilde{Z}_k)$ (bottom row) of the ego-car for various times k. Black sets mark the swept volume $\mathfrak{C}_k^i(\mathcal{Z})$ (top row) and the footprint $\mathcal{B}^i(x_k^i; \mathcal{R})$ (bottom row) of other traffic participants. The entry and exit sets are shown in green and red, respectively. Note how the ego-car maintains a safe distance to the other cars and the road boundaries at all times.

Algorithm 2. All compositions have been deemed *safe*. Henceforth, any car abiding by the controller contract and the assume-guarantee contracts can safely transition through the network under the assumption of the used traffic model.

In Figure 3-7, average computation times are shown for one iteration. We observe that the largest cost comes from the reachability analysis itself. We also observe large variations in the computation times arising from the varying level of complexity,

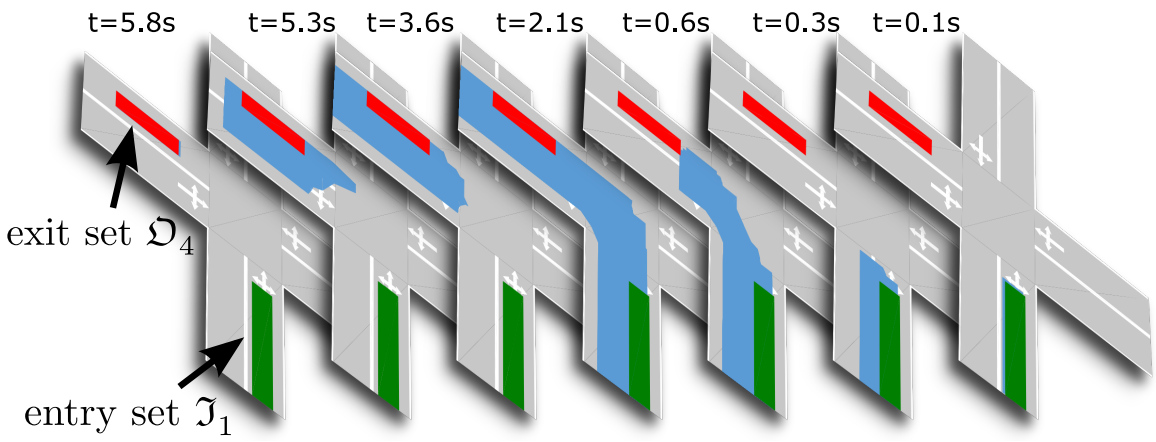


Figure 3-5: Backward propagation on the four-way intersection m for various times k. At $t = 6.0s$, we start out at $\widehat{Z}_T = exit(m, 1, 4) = \mathfrak{O}_4$ and compute the backward reachable set $\widehat{Z}_k$ (marked blue) for each time k to obtain the safe entry set $entry(m, 1, 4)$.

i.e., level of fragmentation of the reachable sets. The backward propagation shows a significantly higher computational demand due to the generally more fragmented sets caused by the computationally more intensive constraints. We conducted our experiments on an Intel Xeon E5-2680 2.8GHz 16 Core CPU with each experiment running on a separate core. On average one experiment took 21 hours.

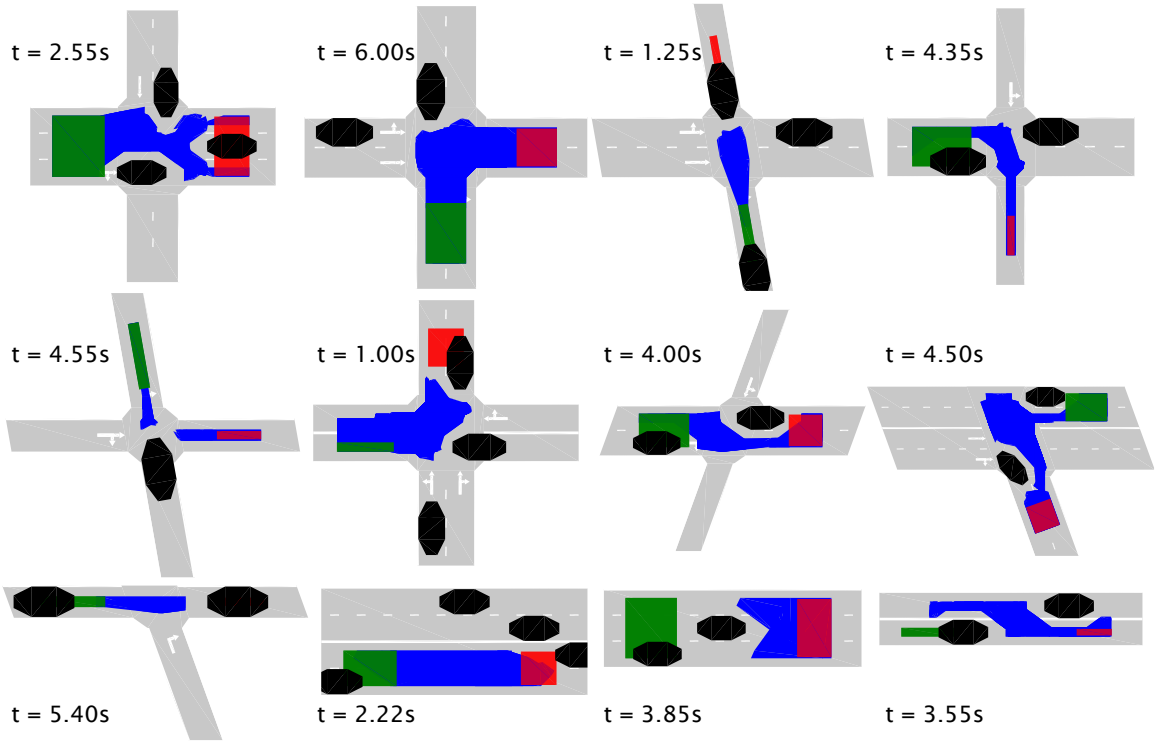


Figure 3-6: A selection of verified road models, comprised of various intersections and straight roads, is shown together with the reachable set (blue) and other traffic participants (black) at the indicated timestep. The initial and final set are marked green and red, respectively.

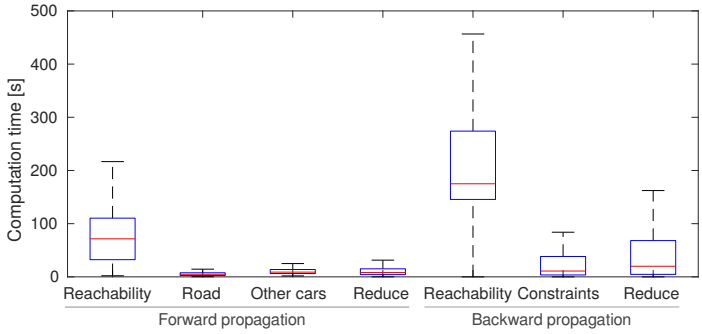


Figure 3-7: The box plot indicates computation times for various parts of one iteration with a fixed timestep of $h = 0.05s$ averaged over all conducted experiments.

Chapter 4

Contract Synthesis with Counterexample-Guided Obstacles

In this chapter[1], we relax the assumption of known, dynamic obstacles and consider the problem of finding more complex traffic scenarios through counterexample-guided search of traffic scenario that would lead to an unsafe behavior, i.e., we solve a falsification problem to search for relevant traffic scenarios. Moreover, we present a method to select a convex subset of the contract to ensure that the contract can be efficiently implemented by an online motion planer (which usually requires convexity of the problem).

4.1 Problem Definition

In this section, we introduce background and formally state the problem of computing contracts for driving scenarios. We give an overview of the notation in this chapter in Table 4.1. For previously introduced notation, please refer to Table 3.1.

[1]This chapter is based on Jonathan A DeCastro*, Lucas Liebenwein*, Cristian-Ioan Vasile, Russ Tedrake, Sertac Karaman, and Daniela Rus. Counterexample-guided safety contracts for autonomous driving. In *International Workshop on the Algorithmic Foundations of Robotics (submitted)*, 2018.

4.1.1 Stochastic Models of the Traffic System

We start with uncertain continuous-time parameterized models of the form

$$\dot{x} = f_\rho(x, u, w), \tag{4.1}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ are states, $u \in \mathcal{U} \subset \mathbb{R}^{n_u}$ are control inputs, $w \in \mathbb{R}^d$ is a Gaussian-distributed disturbance vector, $w \sim \mathcal{N}(0, \Sigma_\rho)$, where Σ_ρ is positive definite, and $\rho \in A$ are fixed model parameters. Our system model $f_\rho(\cdot)$ is assumed to be C^1 continuous, and the sets $\mathcal{X}$ and $\mathcal{U}$ to be compact. Compared to the previous chapter, we note that we explicitly consider uncertainty in the dynamics of the ego-car and the traffic participants. We work from a decomposition of our system model as a coupling of N closed-loop parameterized traffic vehicle models, plus one additional system capturing the physics model for the ego-car:

$$\dot{x} = \begin{bmatrix} \dot{z} \\ \dot{x}^1 \\ \vdots \\ \dot{x}^N \end{bmatrix} = \begin{bmatrix} f_{ego}(z, u) \\ f_{1,\rho_1}(x, w^1) \\ \vdots \\ f_{N,\rho_N}(x, w^N) \end{bmatrix}. \tag{4.2}$$

Here, we decompose x as $x^i \in \mathcal{X}^i \subseteq \mathbb{R}^{n_{i,\rho_i}}$, w as $w^i \in \mathbb{R}^{d_{i,\rho_i}}$, $w^i \sim \mathcal{N}(0, \Sigma^i_{\rho_i})$, $i = 1, \ldots, N$, as representing an uncontrollable perturbation for each traffic vehicle i, explaining the uncertainties in how individual drivers behave. We dedicate u as being the driving commands for the ego-car, whose state is $z \in \mathbb{R}^{n_{ego}}$ as before. Given a discretization $k = \{0, \ldots, T\}$, we define a trajectory as the sequence of $\{x_k, u_k, w_k\}_{k=0}^T$, and denote $p(w_0, \ldots, w_T)$ as the joint probability density function over the disturbances $\{w_0, \ldots, w_T\}$.

Note that the above disturbance model satisfies many learning-based structures in the literature. For instance, to implement the model of [8], each vehicle's behavior model would take on a feedback form involving a nonlinear function of state and an additive Gaussian-distributed stochastic term, which is a special case of (4.1). The parameter ρ may characterize particular styles of driving behaviors, for instance the

spectrum describing average driving to aggressive driving. We will illustrate this point further in Section 4.3.

4.1.2 Problem Formulation

Let a *scenario* be defined as a tuple $\mathcal{S} = (\mathcal{R}, A, \mathfrak{I}, \mathfrak{D})$ consisting of a specification of a road in $\mathbb{R}^2$ and its ruleset (a Boolean formula in states) $\mathcal{R}$, a fixed set of model parameters A, and a set of possible initial conditions for each car $\mathfrak{I} \subseteq \mathcal{X}$ and a final set for the ego-car $\mathfrak{D} \subseteq \mathcal{Z}$. This is similar to the road model m, defined in Section 3.2.1, but takes slightly different parameters.

Let φ be a *safety condition*, a Boolean formula

$$\varphi := \mu \mid \neg\mu \mid \varphi \wedge \psi \mid \varphi \vee \psi,$$

denoting functions of states that describe the conditions for safety of the vehicle. φ can represent, for instance, collisions between cars, departing a lane, or breaking certain rules or liability bounds. We further define $\psi : \mathbb{R}^n \to \mathbb{R}$ to be a quantitative measure on the state space for the entire traffic system and say that $x \in \mathbb{R}^n$ *satisfies* φ, i.e., $x \models \varphi$, if and only if $\psi(x) > 0$. Otherwise, the specification is *falsified*, i.e., $x \not\models \varphi$.

Our goal is, for a given scenario $\mathcal{S}$ and safety condition φ, to find a set of counterexamples to φ as bounded-time trajectories for all of the traffic participants. For each counterexample, we then seek a (controller) *contract* $\mathcal{C} \subset \mathcal{Z}$ that can be applied as a rule for the ego-car to follow in order to guard against the counterexample and thereby locally satisfy φ. We impose the following requirements:

1. $\mathcal{C}$ yields certain constraints on the ego-car's trajectories that prevent violating a given ruleset (e.g. rules of the road),

2. $\mathcal{C}$ yields additional constraints on the ego-car's trajectories that prevent violation of φ with respect to the counterexamples associated with $\mathcal{C}$,

3. $\mathcal{C}$ generalizes to protect the ego-car against a continuum of possible traffic vehicle

behaviors under w^i, in addition to those in the finite set of counterexamples, and

4. the counterexamples associated with $\mathcal{C}$ satisfy a chance constraint describing reasonable driver behaviors, i.e., $p(w_1, \ldots, w_T) \geq \alpha T$ for some $\alpha > 0$.

If a particular counterexample satisfies such a chance constraint, then we know that it is reasonably well-explained by the underlying behavior model of actual driver behaviors. On the other hand, if this check fails, then the counterexample can be considered to be "uncanny" behavior that does not resemble true driving behaviors and the ego-car needs not have a contract. Contracts with different road rules can be compared to examine the affordances or compromises to safety.

For the sake of simplicity of the contracts as well as computational efficiency, the approach in this paper seeks to attain a *convex* contract representation that asserts, under the assumptions of the scenario $\mathcal{S}$, the ego-car is guaranteed to remain safe with respect to a finite, but diverse, set of counterexamples associated with $\mathcal{C}$.

Compared to Chapter 3, in this chapter we focus on generating contracts that are applicable to a wide range of traffic behaviors – as opposed to a fixed, predetermined traffic scenario. Moreover, the contract can be readily implemented by any real-time motion planner or controller since we additional consider a method to convexify the proposed contract, which enables significant speed-ups during deployment – as opposed to a non-convex, more general representation.

4.2 Methods

Contracts are created by an alternation between falsification and reachability under the scenario model (4.2).

4.2.1 Overview

The overall approach is as shown in Algorithm 5. Starting with a set of initial contracts that enforce a ruleset, the falsification step (GenerateCounterexamples)

Table 4.1: Extended symbols table.

$\mathcal{X}$	joint state-space
$\mathcal{X}^i$	state-space of car i
ρ	dynamics parameters
$f_\rho(\cdot)$	joint state-space, parameterized uncertain dynamics model
x	joint state of ego-car and traffic cars
w	joint Gaussian distributed disturbance vector
Σ_ρ	parameterized Gaussian covariance matrix
$p(w_0, \ldots, w_T)$	joint probability density function for all times
$\mathcal{S}$	scenario tuple
φ	Boolean safety condition
$\psi(\cdot)$	quantitative safety measure
$\bar{u}$, $\bar{w}$, $\bar{x}$	time-indexed control, disturbance, and state sequence, respectively
$F_{safe}(t_k; Z_0)$	safe forward reachable set at time t_k starting from the set Z_0
a_k^i, b_k^i	hyperplane parameters guarding the ego-car from car i at time k

generates counterexamples to these contracts (if any exist) by solving for possible ego-car and traffic behaviors that result in failure of φ. In the reachability step (GenerateContract), a reach-avoid problem, similar to the one described in Algorithm 1, is then solved to find an overapproximation of the set of time-indexed states for the ego-car, for which the ego-car is able to steer away from the generated counterexample. The failure case is indicative of an undecidable result, where it is inconclusive whether the ego-car can take *any* action to remain safe under the given scenario.

Figure 4-1 depicts two iterations of the overall procedure. The left-hand side depicts the reachablility step, in which a ruleset and any existing contracts are considered as constraints in the reachable set computation. The right-hand side illustrates how we use falsification to find counterexamples with respect to the contracts. The counterexample is treated as an obstacle to avoid in the subsequent iteration, at which point, a set of constraints are created that separates the set difference between the reachable set at the previous step and the one at the current step.

Solving the reach-avoid problem, versus constructing contracts based only on counterexamples, serves two purposes. First, it allows selection of new contracts that minimize the volume of the reachable set treated as unsafe in the next iteration under the contract. Second, we can verify whether it is feasible for the ego-car to reach the final

Algorithm 5 SYNTHESIZECONTRACTS

Input: $\mathcal{S} = (\mathcal{R}, A, \mathfrak{I}, \mathfrak{O})$: scenario, $\psi(\cdot)$: safety condition function, α: chance constraint, T: time horizon.

Output: $\mathcal{C}$: safety contract for each timestep $k \in \{0, \ldots, T\}$.

1: $\mathcal{C} \leftarrow$ INITIALIZECONTRACT($\mathcal{S}$)
2: **repeat**
3: $\quad \bar{x}, p(\bar{w}) \leftarrow$ GENERATECOUNTEREXAMPLES($\mathcal{S}, \mathcal{C}, \psi(\cdot), T$)
4: $\quad \mathcal{C} \leftarrow$ GENERATECONTRACT($\mathcal{S}, \mathcal{C}, \bar{x}$)
5: **until** $(p(\bar{w}) < \alpha T) \vee (\mathcal{C} = \emptyset)$
6: **if** $(\mathcal{C} = \emptyset) \vee \neg$ISREACHABLE($\mathfrak{O}, \mathcal{C}_T$) **then**
7: $\quad$ **return failure**
8: **return** $\mathcal{C}$

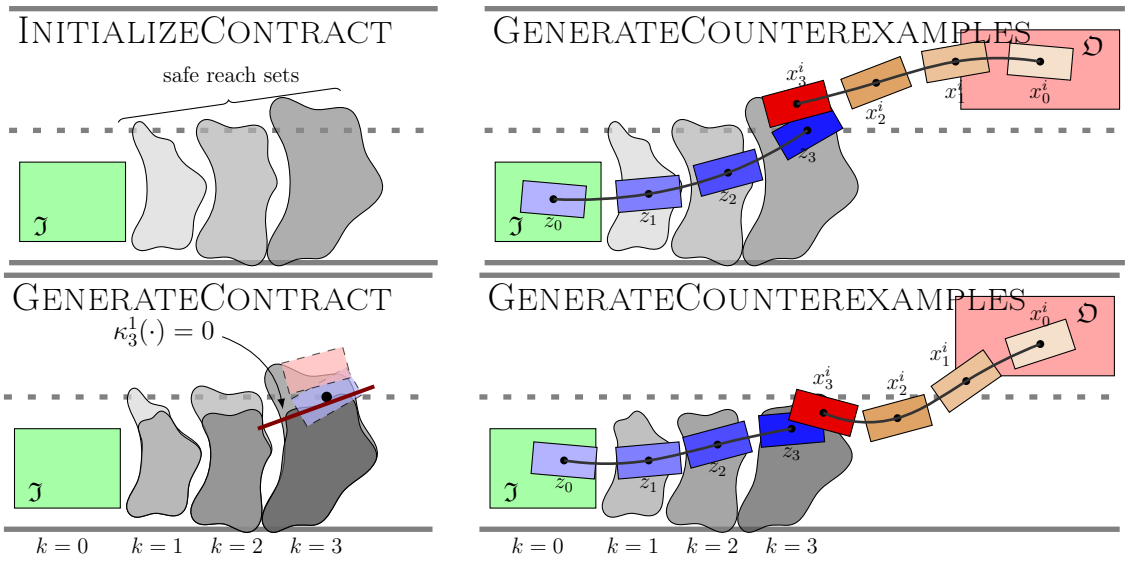


Figure 4-1: Two iterations of the overall approach.

set from within the initial set under the computed contracts (the check ISREACHABLE in Algorithm 5).

4.2.2 Gradient-Based Probabilistic Falsification

Below, we describe the subroutine GENERATECOUNTEREXAMPLES, which is called from Algorithm 5, Line 3 to generate counterexample traffic scenarios. In general, we are interested in finding counterexamples to the safety specification: dynamically feasible trajectories that falsify our safety condition φ within a given chance constraint on the underlying probabilistic behavior model. We solve a direct-collocation trajec-

tory optimization problem [31] by discretizing time $k = \{0, \ldots, T\}$ with time step h, where $t_k = hk$. We let $\bar{u}$, $\bar{w}$, $\bar{x}$ denote, respectively, the sequences $\bar{u} = \{u_k\}_{k=0}^T$, $\bar{w} = \{w_k\}_{k=0}^T$, and $\bar{x} = \{x_k\}_{k=0}^T$.

The counterexample trajectory is summarized by a collection of decision variables $\{h, \bar{u}, \bar{w}, \bar{x}\}$ that falsify the condition $\bar{x} \models \varphi$ but satisfy, at a minimum, the system dynamics (4.2), the initial conditions, and some threshold on the likelihood of selecting the random perturbations $\bar{w}$. We aim to find the most likely explanations of the failure under the given model motivating the following problem:

$$
\begin{aligned}
&\max_{h,\bar{u},\bar{w},\bar{x}} p(\bar{w}) \\
&\text{s.t. } x_{k+1} - x_k = h f_{collocation}, \quad \forall k = 0, \ldots, T-1 && \text{(dynamics)} \\
&\quad x_k \in \mathcal{X}, \quad \forall k = 0, \ldots, T \\
&\quad u_k \in \mathcal{U}, \quad \forall k = 0, \ldots, T-1 \\
&\quad x_0 \in \mathcal{X}_0, \quad u_0 \in \mathcal{U} && \text{(initial conditions)} \\
&\quad \psi(x_T) \leq 0 && \text{(safety specification)} \\
&\quad \kappa_k^j(x_k) \leq 0, \quad \forall j = 1, \ldots, Q, \ \forall k = 0, \ldots, T && \text{(contracts)} \\
&\quad p(\bar{w}) \geq \alpha T && \text{(chance constraint)}
\end{aligned}
\tag{4.3}
$$

where $f_{collocation} = \frac{1}{6}(f_k + 4\tilde{f} + f_{k+1})$, $f_k = f_\rho(x_k, u_k, w_k)$, and

$$
\tilde{f} = f_\rho\left(\frac{1}{2}(u_k + u_{k+1}) + \frac{h}{8}(f_k + f_{k+1}), \frac{1}{2}(u_k + u_{k+1}), \frac{1}{2}(w_k + w_{k+1})\right).
$$

The function $\kappa_k^j(\cdot) \in \mathcal{C}$ represents constraints of the form $(a_k^j)^T x_k \leq b_k^j$, $a_k^j \in \mathbb{R}^{n_{ego}}$, $b_k^j \in \mathbb{R}$ at time step k due to a contract j, the safety-preserving contracts the ego-car must adhere to. When (4.3) is solvable, we end up with corner cases to the hypothesis for $\kappa^j(\cdot)$ found thus far. Every time a new constraint is added to $\mathcal{C}$, the condition φ becomes harder and harder to falsify. We revisit the computation of $\kappa^j(\cdot)$ in Section 4.2.4.

Notice that we can choose to leave out the last constraint in (4.3) since the optimal choice of $\bar{w}$ is a maximizer for $p(\bar{w})$ and hence a check of the optimal values is sufficient

to verify the chance constraint.

The task now is to find the representation $p(\bar{w})$ and express $J(\bar{w})$ as a convex cost such that $\arg\max_{\bar{w}} p(\bar{w}) = \arg\max_{\bar{w}} J(\bar{w})$. Taking $w_k \sim \mathcal{N}(0, \Sigma)$ (where Σ is block-diagonal of Σ^i) and noting the probability of action w_k is drawn from the distribution

$$p(w_k) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} \exp(-\frac{1}{2} w_k^T \Sigma^{-1} w_k)$$

we obtain a log-probability distribution

$$p(w_1, \ldots, w_T) = -\prod_{k=0}^{T} p(w_k).$$

We can easily obtain the log-likelihood representation of the probability as

$$\log p(\bar{w}) = \sum_{k=0}^{N} \log p(w_k) = -\frac{nN}{2} \log 2\pi - \frac{N}{2} \log |\Sigma| - \frac{1}{2} \sum_{k=0}^{N} w_k^T \Sigma^{-1} w_k.$$

Due to monotonicity of the log operator, the cost function may be more simply written as

$$J(\bar{w}) = \sum_{k=0}^{N} w_k^T \Sigma^{-1} w_k,$$

and the chance constraint as $\log p(\bar{w}) \geq \log \alpha - \log T$. We argue that maximizing $p(\bar{w})$ serves the important purpose of maximizing the diversity of the new counterexamples.

Note that the problem in (4.3) is a nonconvex one to solve in general. Hence, we cannot guarantee a solution will be found and hence cannot hope to exhaust all possible counterexamples (i.e. achieve completeness). However, it is important to note that we can achieve *soundness* of our solutions to (4.3).

4.2.3 Collision-Free Safety Conditions

The safety specification constraint in the formulation (4.3) of the falsification problem involves Minkowski operations, which are difficult to solve analytically. To this end, we consider the following reformulation of the safety specification. Let $\mathcal{B}^i(x_k^i; \mathcal{R}) \subset \mathbb{R}^2$ be the orientation-dependent footprint of the i^{th} vehicle at time k, that is, the

Cartesian space occupied at state x^i. The footprints may be adjusted to consider, e.g., uncertainty about the state of the other cars, i.e., it may be an overapproximation to the true footprint of the traffic cars, as previously described in Section 3.1.3. Our safety criteria is one where we wish to avoid crashes with other vehicles, i.e.,

$$\varphi = \bigwedge_{i,k} (\mathcal{B}(z_k) \cap \mathcal{B}^i(x_k^i; \mathcal{R})) = \emptyset.$$

We apply the mild assumption that we only search for conditions in which the ego-car is in collision with only one other vehicle at a time, which makes it easy to reduce a potentially combinatorial problem into one in which we solve (4.3) sequentially.

Unfortunately, finding analytical forms for collision of two rectangular objects involves Minkowski operations, which are difficult to solve analytically as previously mentioned. We instead express collision in terms of two inscribing ellipses using the following result.

Lemma 1. Let a^i and b^i be the length and width of vehicle i, $p^i = [\ x^i \quad y^i\ ]^T$ be its Cartesian coordinates, and θ^i its angle. Let $C^i = R(\theta^i) \begin{bmatrix} a^i & 0 \\ 0 & b^i \end{bmatrix}$, where $R(\theta^i)$ is a rotation matrix, and let $\tilde{p} = C_0^{-1}(p^1 - p^0)$. Then,

$$\mathcal{B}^i(x^0; \mathcal{R}) \cap \mathcal{B}^i(x^1; \mathcal{R}) \neq \emptyset \Rightarrow \tilde{z}^T (C_1 + R_1)^{-1} (C_1 + R_1)^{-T} \tilde{z} \leq 1. \tag{4.4}$$

Proof. (sketch) Condition (4.4) can be obtained directly by transforming one of the ellipses to the unit disc, then applying the same transform to the other ellipse and writing out the expression for containment of the origin. □

We note that the constraint (4.4) preserves soundness of the falsification problem: when a trajectory is found that satisfies this condition, that trajectory falsifies φ.

4.2.4 Reachability with Contracts

In the following, we describe the subroutine GenerateContract, which is called from Algorithm 5, Line 4, and generates a new proposal contract at each iteration.

Let $F(t_k; Z_0)$ denote an over-approximation to the reachable set at time t_k at iteration j of the main loop in Algorithm 5, i.e., the time-indexed set of states $z_k \in \mathcal{Z}$ for which there exists a control $u : \mathbb{R}_{\geq 0} \mapsto \mathcal{U}$ containing the trajectories satisfying $\dot{z} = f_{ego}(z, u)$ when starting in the initial set Z_0. We also refer the reader to Section 3.2.2 for more details on how the reachable set is defined.

Our objective is essentially the converse of the falsification problem: to compute a safe reachable set for the ego-car $F_{safe}(t_k; Z_0) \subseteq F(t_k; Z_0)$ such that it preserves the ruleset $\mathcal{R}$ and is not in collision with any other traffic vehicle at all timesteps.

An overview of the approach may be found in Algorithm 6. For simplicity, we only present the computation of forward reachable sets but this can be extended to backward reachable sets with the modifications explained in Chapter 3. Once a contract is created, we extend the reachable set to verify that it intersects the goal region.

Algorithm 6 GENERATECONTRACT ($\oplus$ denotes the Minkowski sum)

Input: $\mathcal{S} = (\mathcal{R}, A, \mathfrak{I}, \mathfrak{O})$: scenario, $\mathcal{C}'$: previous contract, $\bar{x}$: a trajectory for the system of traffic cars, T: time horizon.
Output: $\mathcal{C}$: safety contract for each timestep $k \in \{0, \ldots, T\}$.

1: $Z_0 \leftarrow \mathfrak{I}$ ▷ assign initial set from scenario
2: $F_{safe}(t_0; Z_0) \leftarrow Z_0 \cap C'_0$
3: $\mathcal{C}_k \leftarrow \mathcal{Z}, \quad \forall k \in \{0, \ldots, T\}$ ▷ initialize the contract for all times
4: **for all** $k \in \{0, \ldots, T\}$ **do**
5: $\quad F(t_k; Z_0) \leftarrow F(h; F_{safe}(t_{k-1}; Z_0)) \cap \mathcal{C}'_k$ ▷ Compute the reachable set
6: $\quad \mathcal{O}^i_k \leftarrow \mathcal{B}^i(x^i_k; \mathcal{R}) \oplus \mathcal{B}(F(t_k; Z_0))$ ▷ Compute the traffic constraints
7: $\quad$ **for all** $i \in \{1, \ldots, N\}$ **such that** $\mathcal{O}^i_k \neq \emptyset$ **do**
8: $\quad\quad \mathcal{C}_k \leftarrow$ COMPUTECONTRACT$(F(t_k; Z_0) \cap \mathcal{C}'_k, \mathcal{O}^i_k) \cap \mathcal{C}_k$
9: $\quad F_{safe}(t_k; X^0_0) \leftarrow F(t_k; X^0_0) \cap \mathcal{C}_k$ ▷ Compute the safe reachable set
10: **return** $\mathcal{C}$

4.2.5 Computation of Contract

We now describe the subroutine COMPUTECONTRACT, called from Algorithm 6, Line 8, which computes a new contract from the reachable set. In particular, given a reachable set $F(t_k; Z_0)$ computed at some iteration k of the main loop in Algorithm 6,

we want to select a hyperplane for each vehicle i, $i \in \{1, \ldots, N\}$, of the form

$$(a_k^i)^T z_k \leq b_k^i$$

such that our new safe set $F_{safe}(t_k; Z_0)$ at time step k is a valid reachable set that is safe with respect to the counterexample as well as respects the current contract $\mathcal{C}_k$. Precisely, in Line 8 of Algorithm 6, values for a_k^i, b_k^i are selected such that $(a_k^i)^T \chi^i > b_k^i$ for all vertices χ^i of $\mathcal{O}_k^i$, the i^{th} footprint of the counterexample, so that we obtain

$$F_{safe}(t_k; Z_0) = F(t_k; Z_0) \cap \{z_k \mid (a_k^i)^T z_k \leq b_k^i, \forall i = 1, \ldots, N\}. \tag{4.5}$$

That is, we select a_k^i and b_k^i such that we may treat it as an obstacle in computing the reachable set at future times. In line 6, we let $\mathcal{B}(Z)$ denote an orientation-dependent Cartesian expansion of some set $Z \subset \mathcal{Z}$, and let $\mathcal{B}^i(x_k^i; \mathcal{R})$ denote a state-dependent inflation of vehicle i according to the ruleset $\mathcal{R}$, as previously described.

Within ComputeContract, we select one hyperplane, i.e., a_k^i and b_k^i, in such a way as to maximize the volume of the resulting safe reachable set $F_{safe}(\cdot; \cdot)$. If we assume $F(\cdot; \cdot)$ is a union of polytopes, we can easily choose one from among the facets that maximizes the union of the remaining polytopes in (4.5) and satisfies $(a_k^i)^T \chi^i > b_k^i$. Finally, $\mathcal{C}_k$ is returned as the intersection of $\mathcal{C}_k'$ and the new contract.

4.2.6 Rules of the Road

In the following we consider the subset of rules from the *Vienna Convention on Traffic Rules* [22], see Table 4.2. The rules described here are associated with the respective road segments, and, together, they constitute the ruleset $\mathcal{R}$. We select these rules as they form a subset of engagement rules for highway scenarios and exclude rules involving traffic signals and other discrete conditionals. For simplicity, we show the constraint sets for straight road segments and equally sized cars.

We assume that the centerline of carriageway is along the x axis of the ego-car for straight road segments. The length of the road segment is denoted by L, the

width of a lane by W, and the number of left and right lanes by n_{left} and n_{right}, respectively. A sequence $0 \leq \xi_x^1 < \zeta_x^1 < \xi_x^2 < \ldots < \zeta_x^{n_{solid}} \leq L$ defines the solid line segments $(\xi_x^\ell, \zeta_x^\ell)$ along the centerline of the road. The pose and longitudinal speed of the vehicles are denoted by (x_c^i, y_c^i, θ^i) and v^i, respectively, where $0 \leq i \leq N$, and $i = 0$ represents the ego-car. The average speed of vehicles around the ego-car and in the same lane is denoted by $\bar{v}$. The safe distances to other vehicles ahead and behind the ego is expressed as $\epsilon_x^{safe} > 0$, while the lateral safe distance to oncoming vehicles is expressed as $\epsilon_y^{safe} > 0$. Overtaking maneuvers are performed within a stretch of the road segment of length $2\,\epsilon_x^{overtake} > 0$ centered on the car that is being overtaken. Overtaking is safe if there are no other cars in the left lane where the ego-car performs the maneuver within a distance of $\epsilon_x^{safe-overtake}$ around the car being overtaken. Lastly, the legal speed limit for a lane is given by $\epsilon_v^{legal} > 0$.

Those rules that are only a function of the ego-car (rules 1, 4, 8) are included in InitializeContract, while those that are functions of the joint state space are included only when a counterexample is obtained from the falsification step. Hence, these rules are included in ComputeContract as a modification to the traffic car footprint, i.e., $\mathcal{B}^i(x_k^i; \mathcal{R})$.

4.3 Implementation

We implemented the falsification algorithm, scenario, and system models using the Drake toolbox [61]. We use the SNOPT optimization package [28] for solving the sequential quadratic program (SQP) in (4.3). We furthermore parallelize the constraint evaluation before passing to the solver in order to speed up the solve time. To generate new contracts, we compute the reachable sets using a Taylor expansion to the nonlinear dynamics with sets being expressed as zonotopes; we do this with the aid of the CORA reachability tool [2]. Set operations are carried out using the MPT toolbox [34], which is based on a polytopic representation of sets.

Table 4.2: Rules of the road for highway scenarios.

No.	Rule	Constraint set
1	Don't drive in the left lanes.	$\{0 \leq x_c^0 \leq L, -n_{right} \cdot W \leq y_c^0 \leq 0\}$
2	If driving behind another car, keep a reasonable distance away to avoid collision if it suddenly stops.	$\{x_c^i - x_c^0 \geq \epsilon_x^{safe}\, v^0 \mid \forall i \,.\, x_c^i - x_c^0 \geq 0 \wedge \lvert y_c^i - y_c^0\rvert < W\}$
3	If you want to slow down, give clear warning and do not inconvenience drivers behind you.	$\{x_c^0 - x_c^i \geq \epsilon_x^{safe}\, v^0 \mid \forall i \,.\, x_c^0 - x_c^i \geq 0 \wedge \lvert y_c^i - y_c^0\rvert < W\}$
4	Don't cross solid lines.	$\{\xi_x^\ell \leq x_c^0 \leq \zeta_x^\ell \wedge -n_{right} \cdot W \leq y_c^0 \leq 0 \mid 1 \leq \ell \leq n_{solid}\}$
5	Overtake on the left when it is safe.	$\{y_c^0 - y_c^i > W \wedge v^0 > v^i \mid \forall i \,.\, v^i > 0 \wedge \lvert x_c^0 - x_c^i\rvert \leq \epsilon_x^{overtake} \wedge \nexists j \,.(\, \lvert x_c^j - x_c^i\rvert \leq \epsilon_x^{safe-overtake} \wedge y_c^0 - y_c^j \leq W)\}$
6	If another vehicle is trying to overtake you keep right and don't accelerate. If necessary, slow down and pull over.	$\{u_a^0 \leq 0 \wedge y_c^i - y_c^0 \geq W \wedge y_c^0 \leq 0 \mid \forall i \,.\, y_c^i - y_c^0 \leq 1.5W \wedge v^i > 0 \wedge \lvert x_c^i - x_c^0\rvert \leq \epsilon_x^{overtake}\}$
7	If passing oncoming traffic, leave sufficient lateral space to not get hit. If obstructed, slow down.	$\{y_c^i - y_c^0 \geq \epsilon_y^{safe} \mid y_c^i \geq 0 \wedge v^i \leq 0\}$
8	Don't drive abnormally slowly such that you impede the progress of other vehicles. Don't drive above the speed limit or abnormally fast.	$\{\lvert v^0 - \bar{v}\rvert \leq \epsilon_v, \lvert v^0\rvert \leq \epsilon_v^{legal}\}$

We consider the following model for both the ego-car and traffic vehicles:

$$\dot{x} = \begin{bmatrix} \dot{x_c} \\ \dot{y_c} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v}{L}\tan(u_\delta) \\ u_a \end{bmatrix}, \quad u = \begin{bmatrix} u_\delta \\ u_a \end{bmatrix}.$$

As previously, x_c and y_c are Cartesian positions at the center of the vehicle, θ is the heading angle, and v is the forward speed, while u_δ and u_a denote the steering angle and acceleration inputs, respectively.

To represent naturalistic behaviors for the traffic cars, we consider the intelligent driver model (IDM) [63], a model whose parameters are typically fit to driver data and which is used to represent the longitudinal actions (acceleration) of actual drivers. We consider a pure-pursuit controller [17] to model the lateral actions (steering) of drivers. Essentially, the IDM model allows cars to react to one another while adapting to a driver's preferences for speed, acceleration, and time headway between vehicles. The pure-pursuit controller allows steering to be adjusted smoothly so that the vehicle converges to a desired curve. In our experiements, we set the desired curve to be fixed as the centerline of a target lane to drive to. We randomize these traffic behaviors by defining Σ^i, where $\Sigma^i = \text{diag}\{\sigma_\delta, \sigma_a\}$, and by treating the disturbance signal as additive noise to the nominal acceleration and steering commands provided by IDM and pure pursuit. We adapt to different driving styles by using the complete list of parameters in Table 4.3. We furthermore augment the implementation by ignoring vehicles beyond a limited perception range.

The road geometry is configurable for highway scenarios in which any number of lanes and lane sizes can be chosen for the scenario. Our aggregate model of the entire traffic system is implemented to account for all of the traffic participants in the context of their positions on the road. The models are encoded such that partial derivatives can be easily obtained via automatic differentiation.

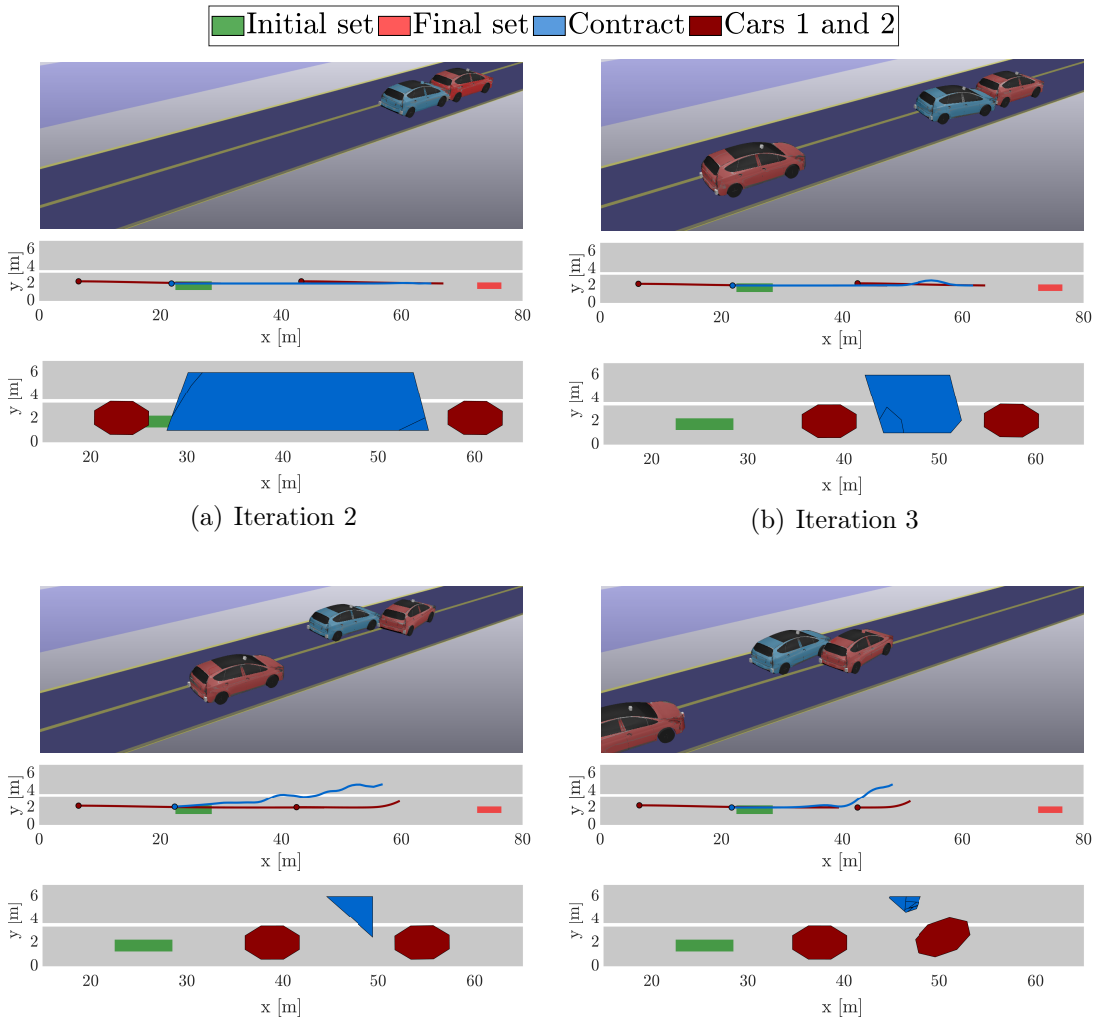
Table 4.3: Parameters used to model driver behaviors for the traffic cars.

	Description	Symbol	Driving Style Normal	Driving Style Aggressive
IDM	Reference speed (m/s)	v_{ref}	10	1.5
	Maximum acceleration (m/s^2)	a	1	4
	Comfortable deceleration (m/s^2)	b	3	6
	Minimum-desired net distance (m)	s_0	1	0.5
	Time headway to lead vehicle (s)	t_h	0.1	0.05
	Free-road exponent	δ	4	4
Pure-Pursuit	Lookahead distance (m)	s_{look}	15	10
Perception	Range (m)	$s_{perception}$	100	100
Disturbances	Steering angle variance (rad^2)	σ_δ	0.1	5
	Acceleration variance (m^2/s^4)	σ_a	0.1	2.5

4.4 Results

In this scenario, we consider the ego-car sandwiched between two traffic cars in the right lane of a two-lane highway with opposing traffic lane, which may be used for overtaking if free. We synthesized contracts using both the ruleset in Table 4.2 and a relaxed ruleset, in which we disable rules 1 and 3 to enable evasive maneuvers onto the other lane. For both rulesets, we explore traffic models having two levels of aggressiveness (normal and aggressive) using the parameters in Table 4.3. In Figure 4-2, we depict different iterations of Algorithm 5 for the relaxed rules and normal driving style. We compare the contracts obtained at a fixed iteration of the algorithm for each case in Figure 4-3 and, for each case, report the log-likelihood of the counterexample normalized on $|\Sigma|$ in Figure 4-4.

We observe that with more iterations (and more unlikely behaviors of the traffic cars) more contracts are added, making the contract more restrictive but also harder to falsify, as indicated by the log-likelihood. With a greater number of rules and more aggressive traffic, we note that the contract gets smaller and more prohibitive (see Figure 4-3). We also note that relaxing the ruleset (e.g. allowing lane switches) enables more behaviors for the ego-car demonstrating that safety can be preserved at the expense of rule-breaking in some scenarios. Moreover, the ego-car can readily



(a) Iteration 2

(b) Iteration 3

(c) Iteration 7

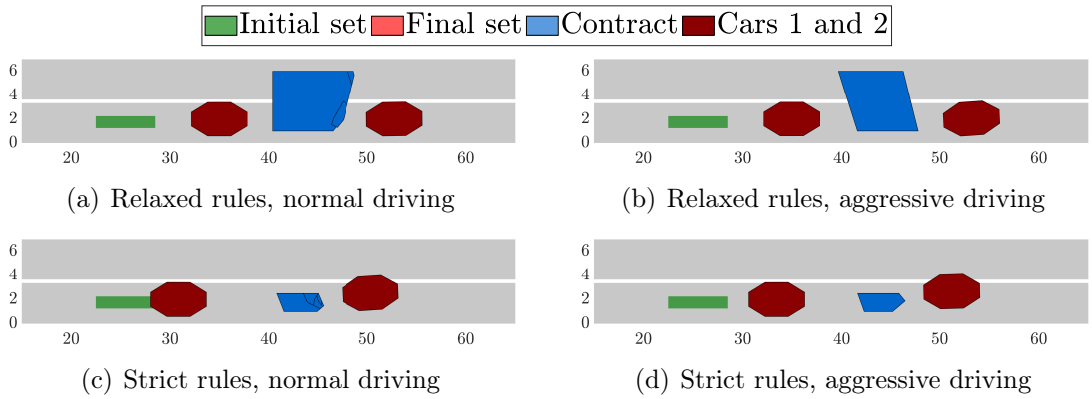
(d) Iteration 14

Figure 4-2: Different iterations of the approach. Within each iteration, the upper two plots indicate a counterexample trajectory of the traffic system that falsifies collision-free behavior under the proposed contracts. The lower plot illustrates a new contract that guards against the counterexample.

estimate the cost of violating rules of the road by observing the varying contracts depending on the set of actively enforced rules.

We note that for the normal driving style, the log-likelihood quickly decreases, whereas for the aggressive driving styles, the log-likelihood remains high as contracts are added indicating that aggressive traffic can induce failure regardless of the ego-car's behavior. In both of the agressive-driving cases, empty contracts were returned before exhausting possible counterexamples. Of the normal-driving cases, the relaxed

set provides a contract with 14 counterexamples, whereas the strict set provides five counterexamples, indicating that changing lanes presents more possible failure events to guard against.



(a) Relaxed rules, normal driving

(b) Relaxed rules, aggressive driving

(c) Strict rules, normal driving

(d) Strict rules, aggressive driving

Figure 4-3: The contract for timestep $t = 4.8s$ at iteration 4 for each set of parameters.

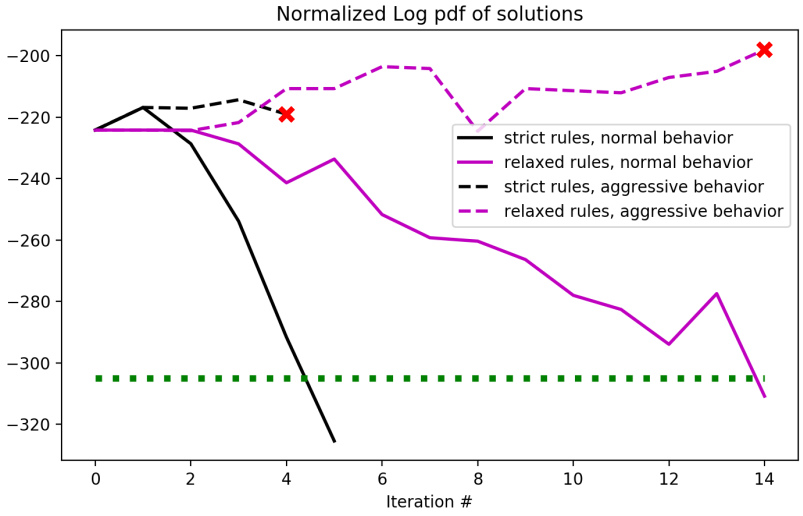


Figure 4-4: The log-likelihood for each test case across all iterations. The red $\times$ marks iterations where the contract terminated with an empty set, and the green dashed line indicates the chance constraint α.

Chapter 5

Conclusion

5.1 Conclusion

In this thesis, we presented a novel approach to safety verification for autonomous systems based on contract synthesis.

For the first part, we studied the problem of safety verification of controllers for autonomous systems and proposed a novel framework for synthesizing safety guarantees for entire road networks building upon compositional assume-guarantee contracts. Our framework hinges on verifying a library of local road models against a given ego-car and fixed traffic models, concurrently with synthesizing safety contracts, which may also be used for the composition of road models. The library can then be used to certify the safety of executing ego-car controllers satisfying a controller contract over road networks. We further demonstrated the effectiveness of our approach on a case study involving a library of local road models, which enabled us to verify a substantial part of Mid-Manhattan.

In the second part, we presented an extension of the framework to facilitate the incorporation of a large variety of probabilistic traffic behaviors and the subsequent generation of the appropriate safety contracts. We overcome issues of computational tractability by iteratively generating a set of safety constraints, based on the methods presented in the first part, and generating counterexamples, i.e., traffic scenarios, using gradient-based probabilistic falsification. We judiciously account for rules of the

roads in terms of state space constraints enforced during reachability analysis. The empirical results on a variety of real-world inspired scenarios validate the favorable performance of our approach and reaffirm its practical applicability.

5.2 Future Work

We envision that our method can be used to inform the decision-making and planning system of an autonomous agent about the appropriate safety constraints applicable in a particular traffic and road scenario in order to guarantee safety while executing a motion plan.

In future work, we plan to extend the library of locally verified road models to capture a wider range of road geometries, including multiway intersections and curved roads. This will enable us to build up a more complete library of road segments and to compose safety guarantees for larger, more complex road networks potentially even during online deployment given that the verified library is large enough.

We are also interested in extending our method to synthesize safety constraints that are simultaneously applicable across a wide variety of traffic scenarios as a way to capture more traffic scenarios. A potential avenue of research in this realm is to use learning-based strategies to synthesize safety constraints that are then verified using formal techniques such as reachability analysis.

Learning-based methods are also a crucial component of perception, prediction, and social planning algorithms. An interesting extension to our current work is to allow our framework to consider those type of algorithms. The challenges here lie in providing safety guarantees of systems that are hard to describe analytically and potentially have a very high-dimensional state space making it difficult to use traditional verification techniques. An alternative verification technique to tackle these challenges could entail probabilistic safety guarantees, as opposed to deterministic guarantees, that provide guarantees for most of the scenarios, as opposed to all the scenarios.

Finally, we envision to augment the verification framework with a roll-out strategy

for controllers that leverage the local safety contracts. The goal is to provide an online safety contract that ensures the long-term safety of the ego-car and show the effectiveness of using safety contracts in real-world experiments.

5.3 Lessons Learned

The work presented in this thesis has taught me numerous crucial insights that will be very valuable for future research. The field of verification lies at the intersection of formal methods, motion planning (including optimization methods), and numerical approximation techniques. This requires the careful consideration of the trade-off between accuracy and computational tractability, even more so than in other aspects of robotics. On the one hand, high-fidelity models enable us to synthesize guarantees that capture complex real-world scenarios, but at the same time such models can quickly become computationally intractable. On the other hand, overly simple models might be very efficient in terms of computation time, but such models make it increasingly difficult to transfer the resulting guarantees to real-world systems. In order to push the boundaries of achievable verification guarantees, it is therefore important to gain problem-specific insights that can help increase the considered model complexity while maintaining computational tractability.

In the research presented here, two insights were critical to the success of the proposed methods. First, by appropriately decomposing the problem, both in terms of the considered traffic scenarios and the spatial domain, we gained computational tractability while being able to consider fairly complex models. Second, by considering contracts instead of actual control systems, we achieved generality of the safety guarantees and simultaneously circumvented the issue of having a potentially complex controller implementation in the loop.

Finally, I would like to mention some of the prevalent challenges in the implementation. A fundamental aspect is to consider appropriate numerical tools that can provide the necessary approximate solutions. Usually, we cannot hope for exact solutions, which necessitates and motivates the use of numerical approximation

tools. I spent quite some time to select the appropriate tools for the simulations and experiments, which would serve our needs. Most of the computations mentioned in this thesis were set-based and required set operations, such as set difference and set intersection. On numerous occasions, I have encountered that the complexity of the set representation had a significant impact on the runtime of the algorithms. A notable portion of effort, thus, went into implementing numerical methods that ensure that the set representation is fairly compact, e.g., for a polytope representation of sets this required to always keep the number of vertices at a minimum. Obtaining some of the results was only made possible by carefully maintaining an efficient set representation.

5.4 Funding

This research was supported in part by the Toyota Research Institute (TRI) and National Science Foundation award IIS-1723943. This article solely reflects the opinions and conclusions of its authors, and not TRI or any other Toyota entity.

Bibliography

[1] D. Althoff, M. Althoff, D. Wollherr, and M. Buss. Probabilistic collision state checker for crowded environments. In *IEEE ICRA*, pages 1492–1498, May 2010. doi: 10.1109/ROBOT.2010.5509369.

[2] Matthias Althoff. An introduction to CORA. In *ARCH, CPSWeek*, pages 120–151, 2015.

[3] Matthias Althoff and John M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE T-RO*, 30(4):903–918, 2014. doi: 10.1109/TRO.2014.2312453. URL `http://dx.doi.org/10.1109/TRO.2014.2312453`.

[4] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, P-H Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical computer science*, 138(1): 3–34, 1995.

[5] Rajeev Alur, Thao Dang, and Franjo Ivančić. Predicate abstraction for reachability analysis of hybrid systems. *Transactions on Embedded Computing Systems*, 5(1):152–199, 2006.

[6] Rajeev Alur, Salar Moarref, and Ufuk Topcu. Compositional synthesis with parametric reactive controllers. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 215–224. ACM, 2016.

[7] Sweewarman Balachandran, Necmiye Ozay, and Ella M. Atkins. Verification guided refinement of flight safety assessment and management system for takeoff. *Journal of Aerospace Information Systems*, 13:357–369, 2016. doi: doi:10.2514/1.I010408. URL `http://dx.doi.org/10.2514/1.I010408`.

[8] Nir Baram, Oron Anschel, Itai Caspi, and Shie Mannor. End-to-end differentiable adversarial imitation learning. In *International Conference on Machine Learning*, pages 390–399, 2017.

[9] Amit Bhatia and Emilio Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 142–156. Springer, 2004.

[10] Davide Bresolin, Luca Geretti, Riccardo Muradore, Paolo Fiorini, and Tiziano Villa. Verification of robotic surgery tasks by reachability analysis: A comparison of tools. In *17th Euromicro Conference on Digital System Design*, pages 659–662. IEEE, 2014.

[11] 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.

[12] Xin Chen, Stefan Schupp, Ibtissem Ben Makhlouf, Erika Ábrahám, Goran Frehse, and Stefan Kowalewski. A benchmark suite for hybrid systems reachability analysis. In *NASA Formal Methods Symposium*, pages 408–414. Springer, 2015.

[13] Peng Cheng and Vijay Kumar. Sampling-based falsification and verification of controllers for continuous dynamic systems. *The International Journal of Robotics Research*, 27(11-12):1232–1245, 2008.

[14] Alongkrit Chutinan and Bruce H Krogh. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In *International Workshop on Hybrid Systems: Computation and Control*, pages 76–90. Springer, 1999.

[15] Edmund Clarke, Orna Grumberg, and D Long. Verification tools for finite-state concurrent systems. In *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*, pages 124–175. Springer, 1993.

[16] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *International Conference on Computer Aided Verification*, pages 154–169. Springer, 2000.

[17] R. Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Carnegie Mellon University, Pittsburgh, PA, January 1992.

[18] Eric Dallal and Paulo Tabuada. Decomposing controller synthesis for safety specifications. In *55th Conference on Decision and Control*, pages 5720–5725. IEEE, 2016.

[19] Jonathan A DeCastro and Hadas Kress-Gazit. Synthesis of nonlinear continuous controllers for verifiably correct high-level, reactive behaviors. *The International Journal of Robotics Research*, 34(3):378–394, 2015.

[20] Jonathan A DeCastro and Hadas Kress-Gazit. Nonlinear controller synthesis and automatic workspace partitioning for reactive high-level behaviors. In *International Conference on Hybrid Systems: Computation and Control*, Vienna, Austria, April 2016. ACM. doi: 10.1145/2883817.2883832.

[21] Jonathan A DeCastro*, Lucas Liebenwein*, Cristian-Ioan Vasile, Russ Tedrake, Sertac Karaman, and Daniela Rus. Counterexample-guided safety contracts for autonomous driving. In *International Workshop on the Algorithmic Foundations of Robotics (submitted)*, 2018.

[22] *Convention on Road Traffic*. Economic Commission for Europe – Inland Transport Committee, Vienna, Austria, E/CONF.56/16/Rev.1/Amend.1 edition, 1968.

[23] Kyle Edelberg, Dennis Wai, Jason Reid, Eric Kulczycki, and Paul Backes. Workspace and reachability analysis of a robotic arm for sample cache retrieval from a mars rover. In *AIAA SPACE Conference and Exposition*, page 4538, 2015.

[24] Stephen M Erlien, Susumu Fujita, and Joseph Christian Gerdes. Shared steering control using safe envelopes for obstacle avoidance and vehicle stability. *Transactions on Intelligent Transportation Systems*, 17(2):441–451, 2016.

[25] 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. ACM, 2015.

[26] Paul Furgale, Ulrich Schwesinger, Martin Rufli, Wojciech Derendarz, Hugo Grimmett, Peter Mühlfellner, Stefan Wonneberger, Julian Timpner, Stephan Rottmann, Bo Li, et al. Toward automated driving in cities using close-to-market sensors: An overview of the v-charge project. In *Intelligent Vehicles Symposium*, pages 809–816. IEEE, 2013.

[27] Roland Geraerts and Mark H Overmars. Reachability analysis of sampling based planners. In *Proceedings of the International Conference on Robotics and Automation*, pages 404–410. IEEE, 2005.

[28] Philip E Gill, Walter Murray, and Michael A Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, 47(1):99–131, January 2005. ISSN 0036-1445. doi: 10.1137/S0036144504446096. URL `http://dx.doi.org/10.1137/S0036144504446096`.

[29] Jeremy H Gillula, Haomiao Huang, Michael P Vitus, and Claire J Tomlin. Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. In *International Conference on Robotics and Automation*, pages 1649–1654. IEEE, 2010.

[30] Jeremy H Gillula, Gabriel M Hoffmann, Haomiao Huang, Michael P Vitus, and Claire J Tomlin. Applications of hybrid reachability analysis to robotic aerial vehicles. *The International Journal of Robotics Research*, 30(3):335–354, 2011.

[31] Charles R Hargraves and Stephen W Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987.

[32] Thomas A Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. *HyTech: A model checker for hybrid systems*, pages 460–463. Springer, Berlin, Heidelberg, 1997. ISBN 978-3-540-69195-2. doi: 10.1007/3-540-63166-6_48. URL `http://dx.doi.org/10.1007/3-540-63166-6_48`.

[33] Thomas A Henzinger, Shaz Qadeer, and Sriram K. Rajamani. *You assume, we guarantee: Methodology and case studies*, pages 440–451. Springer, Berlin, 1998. ISBN 978-3-540-69339-0. doi: 10.1007/BFb0028765. URL `http://dx.doi.org/10.1007/BFb0028765`.

[34] Martin Herceg, Michal Kvasnica, Colin N Jones, and Manfred Morari. Multi-parametric toolbox 3.0. In *European Control Conference*, pages 502–510. IEEE, 2013.

[35] Fabian Immler. Verified reachability analysis of continuous systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 37–51. Springer, 2015.

[36] Nidhi Kalra and Susan M Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94, 2016. URL `http://www.rand.org/pubs/research_reports/RR1478.html`.

[37] James Kapinski, Jyotirmoy V Deshmukh, Sriram Sankaranarayanan, and Nikos Arechiga. Simulation-guided lyapunov analysis for hybrid dynamical systems. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, pages 133–142. ACM, 2014.

[38] Sertac Karaman and Emilio Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *49th Conference on Decision and Control*, pages 7681–7687. IEEE, 2010.

[39] Eric S Kim, Murat Arcak, and Sanjit A Seshia. Compositional controller synthesis for vehicular traffic networks. In *54th Annual Conference on Decision and Control*, pages 6165–6171. IEEE, 2015.

[40] Eric S Kim, Sadra Sadraddini, Calin Belta, Murat Arcak, and Sanjit A Seshia. Dynamic contracts for distributed temporal logic control of traffic networks. In *56th Annual Conference on Decision and Control*, pages 3640–3645. IEEE, 2017.

[41] Jerome Le Ny and George J Pappas. Sequential composition of robust controller specifications. In *International Conference on Robotics and Automation*, pages 5190–5195. IEEE, 2012.

[42] Lucas Liebenwein, Wilko Schwarting, Cristian-Ioan Vasile, Jonathan DeCastro, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Compositional and contract-based verification for autonomous driving on road networks. In *International Symposium on Robotics Research*. International Foundation of Robotics Research, 2017.

[43] Lucas Liebenwein*, Cenk Baykal*, Igor Gilitschenski, Sertac Karaman, and Daniela Rus. Sampling-based approximation algorithms for reachability analysis with provable guarantees. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, PA, June 2018. doi: 10.15607/RSS.2018.XIV.014.

[44] Stefan B Liu, Hendrik Roehm, Christian Heinzemann, Ingo Lütkebohle, Jens Oehlerking, and Matthias Althoff. Provably safe motion of mobile robots in human environments. In *International Conference on Intelligent Robots and Systems*, pages 1351–1357. IEEE, 2017.

[45] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36 (8):947–982, 2017.

[46] David Q Mayne, María M Seron, and SV Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[47] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *Transactions on Automatic Control*, 50(7):947–957, 2005.

[48] Petter Nilsson, Omar Hussien, Ayca Balkan, Yuxiao Chen, Aaron D Ames, Jessy W Grizzle, Necmiye Ozay, Huei Peng, and Paulo Tabuada. Correct-by-construction adaptive cruise control: Two approaches. *Transactions on Control Systems Technology*, 24(4):1294–1307, 2016.

[49] Matthew O'Kelly, Houssam Abbas, Sicun Gao, Shin'ichi Shiraishi, Shinpei Kato, and Rahul Mangharam. Apex: a tool for autonomous vehicle plan verification and execution. In *SAE World Congress and Exhibition*, 2016.

[50] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

[51] Erion Plaku, Lydia E Kavraki, and Moshe Y Vardi. Falsification of ltl safety properties in hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 368–382. Springer, 2009.

[52] Oliver Porges, Roberto Lampariello, Jordi Artigas, Armin Wedler, Christoph Borst, and Máximo A Roa. Reachability and dexterity: Analysis and applications for space robotics. In *Proceedings of the Workshop on Advanced Space Technologies for Robotics and Automation*, 2015.

[53] Matthias Rungger and Majid Zamani. Compositional construction of approximate abstractions of interconnected control systems. *Transactions on Control of Network Systems*, 2016.

[54] Sadra Sadraddini, Janos Rudan, and Calin Belta. Formal synthesis of distributed optimal traffic control policies. In *International Conference on Cyber-Physical Systems*, Pittsburgh , PA, 2017.

[55] Alberto Sangiovanni-Vincentelli, Werner Damm, and Roberto Passerone. Taming Dr. Frankenstein: Contract-based design for cyber-physical systems. *European Journal of Control*, 18(3):217–238, 2012.

[56] Sriram Sankaranarayanan and Georgios Fainekos. Falsification of temporal properties of hybrid systems using the cross-entropy method. In *Proceedings of the 15th International Conference on Hybrid Systems: Computation and Control*, pages 125–134. ACM, 2012.

[57] Wilko Schwarting, Javier Alonso-Mora, Liam Paull, Sertac Karaman, and Daniela Rus. Parallel autonomy in automated vehicles: safe motion generation with minimal intervention. In *International Conference on Robotics and Automation*, pages 1928–1935. IEEE, 2017.

[58] Homayoun Seraji. Reachability analysis for base placement in mobile manipulators. *Journal of Robotic Systems*, 12(1):29–43, 1995.

[59] Victor A Shia, Yiqi Gao, Ramanarayan Vasudevan, Katherine Driggs Campbell, Theresa Lin, Francesco Borrelli, and Ruzena Bajcsy. Semiautonomous vehicular control using driver modeling. *Transactions on Intelligent Transportation Systems*, 15(6):2696–2709, 2014.

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

[61] Russ Tedrake and the Drake Development Team. Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems, 2016. URL `http://drake.mit.edu`.

[62] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. LQR-trees: feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[63] Martin Treiber and Arne Kesting. *Traffic Flow Dynamics: Data, Models and Simulation*. Springer, Berlin, 2013. ISBN 978-3-642-32459-8.

[64] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[65] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. Receding horizon temporal logic planning for dynamical systems. In *Proceedings of the 48th Conference on Decision and Control*, pages 5997–6004. IEEE, 2009.

[66] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. Receding horizon temporal logic planning. *Transactions on Automatic Control*, 57(11):2817–2830, 2012.

[67] Zhixing Xue and Ruediger Dillmann. Efficient grasp planning with reachability analysis. *International Journal of Humanoid Robotics*, 8(04):761–775, 2011.