Technische Universität München
TUM School of Computation, Information and Technology

Generating Safety-Critical Test Scenarios for Motion Planning Algorithms of Autonomous Vehicles

Moritz Sebastian Klischat

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Marco Caccamo

Prüfer der Dissertation:

1. Prof. Dr.-Ing. Matthias Althoff
2. Prof. Dr.-Ing. Markus Lienkamp

Die Dissertation wurde am 17.05.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 19.12.2023 angenommen.

Acknowledgement

I want to thank especially my supervisor Matthias Althoff for providing guidance throughout the years. I would also like to thank my former colleagues for the time at the Chair of Robotics, Artificial Intelligence and Embedded Systems – especially those whom I collaborated with and who provided feedback to the dissertation. Furthermore, I want to thank the students that supported me with their thesis, as HiWis, or in practical courses. Lastly, I want to thank Lisa and my family for their continuous support.

Garching, May 07, 2023 *Moritz Sebastian Klischat*

Abstract

Ensuring the safety of autonomous vehicles is one of the main challenges in their development. This is especially difficult due to the open-ended nature of the traffic in which they must operate after deployment. They may encounter an infinite number of situations that must be handled safely. This presents not only a challenge for the development of the algorithms that control the vehicles but also for the validation of their safety before the vehicles can be deployed. Efficient approaches are crucial in addressing this issue, and virtual testing using simulations is one of the main pillars for managing the effort. Focusing on safety-critical edge cases further enables reducing the number of test scenarios required to assess the safety of a system or to find failures during the development more quickly. In this thesis, we propose methods for creating safety-critical test scenarios and falsification methods that uncover failures of motion planning algorithms and are suitable for a scenario-based setting.

In the first part of the thesis, we formulate scenario generation as an optimization problem to increase the criticality of provided scenarios using a metric based on the reachable set of the ego vehicle. The reachable set abstracts the behavior of any motion planning algorithm by representing the solution space of feasible trajectories. Thus, our method enables the generation of scenarios independently of a specific system. For solving the highly nonlinear optimization problem, we propose a parameterization and evolutionary algorithms that are combined with convex optimization to find feasible and critical scenarios efficiently. To further reduce the computation time of the optimization, we develop a method to compute the reachable set of road vehicles more efficiently than previous methods by exploiting precomputed reachability graphs. To scale the generation of scenarios without relying on real-world traffic data, we integrate the scenario generation with an automated workflow where we extract high-definition maps from OpenStreetMap and use a traffic simulator to create traffic data. Our optimization algorithm then enhances the criticality of these scenarios.

In the second part, we present methods for deriving concrete scenarios from abstract scenario specifications. We develop a constraint-based representation for scenario specifications, which is incorporated into a reachability analysis to compute the reachable set of all traffic participants that complies with the specification. Our method to sample trajectories within this reachable set is combined with Monte-Carlo tree search to falsify provided motion planners. We demonstrate the efficiency and scalability of our methods in several numerical experiments and provide the generated scenarios in the open-source CommonRoad benchmark suite.

Zusammenfassung

Die Sicherheit autonomer Fahrzeuge zu gewährleisten, ist eine der größten Herausforderungen bei ihrer Entwicklung aufgrund der theoretisch unendlichen Anzahl an möglichen Verkehrssituationen, denen sie während ihres Betriebs ausgesetzt sein können. Diese Situationen zuverlässig und sicher bewältigen zu können, stellt nicht nur eine Herausforderung dar für die Algorithmen, welche die Fahrzeuge steuern, sondern insbesondere für Testverfahren im Rahmen ihrer Validierung. Effiziente Testverfahren sind daher von entscheidender Bedeutung, um der Komplexität dieser Aufgabe begegnen zu können. Hierbei ist virtuelles Testen mithilfe von Simulationen eines der wichtigsten Werkzeuge. Dabei ermöglicht der Fokus auf sicherheitskritische Fälle, die Anzahl der benötigten Testszenarien zur Beurteilung der Sicherheit eines Systems reduzieren zu können und eine schnellere Detektion von Fehlern während der Entwicklung. In dieser Dissertation schlagen wir Methoden vor, um automatisiert sicherheitskritische Testszenarien zu generieren und entwickeln effiziente Falsifizierungsmethoden, mit denen Fehler von Bewegungsplanungsalgorithmen gezielt gefunden werden können.

Im ersten Teil der Arbeit formulieren wir die Erzeugung von Testszenarien als Optimierungsproblem, in welchem sicherheitskritische Szenarien mithilfe einer Kritikalitätsmetrik basierend auf der erreichbaren Menge des Egofahrzeugs erzeugt werden. Die erreichbare Menge abstrahiert dabei das Verhalten beliebiger Bewegungsplaner, indem sie den Lösungsbereich aller gültigen Trajektorien darstellt. Dadurch ermöglicht unsere Methode die Erstellung von Szenarien unabhängig von einem spezifischen System. Um das daraus resultierende nichtlineare Optimierungsproblem zu lösen, schlagen wir eine effiziente Parametrisierung und evolutionäre Algorithmen vor, die mit konvexer Optimierung kombiniert werden. Durch unseren Ansatz können somit effizient gültige kritische Szenarien gefunden werden. Um die Berechnungszeit der Optimierung zu reduzieren, entwickeln wir außerdem eine Methode, mit der die erreichbare Menge von Fahrzeugen effizienter berechnet werden kann als mit bisherigen Methoden. Hierbei nutzen wir vorberechnete Erreichbarkeitsgraphen und entwickeln eine Methode, welche eine geringe Überapproximation der berechneten Menge ermöglicht. Um die Generierung von Szenarien ohne reale Verkehrsdaten durchführen zu können, integrieren wir die Szenariengenerierung in einen automatisierten Prozess, in dem hochauflösende Karten aus OpenStreetMap extrahiert werden und eine Verkehrssimulation verwendet wird, um Verkehrsszenarien zu erzeugen. Diese werden anschließend mit unser zuvor vorgestellten Methode hinsichtlich ihrer Kritikalität optimiert.

Im zweiten Teil stellen wir Methoden zur Ableitung konkreter Szenarien aus abstrakten Szenariospezifikationen und einen Algorithmus zur Falsifizierung von Bewegungsplanern vor. Dazu entwickeln wir zunächst eine Repräsentation von Szenariospezifikationen über Randbedingungen. Diese werden in eine Erreichbarkeitsanalyse eingebunden, mit der die spezifikationskonforme erreichbare Menge aller Verkehrsteilnehmer des Szenarios berechnet werden kann. Darauf aufbauend entwickeln wir eine Methode zum Sampling von Trajektorien aus dieser erreichbaren Menge, welche in ein Monte-Carlo Baumsuchverfahren zur Falsifizierung integriert wird. Wir untersuchen die Effizienz und Skalierbarkeit unserer Methoden anhand mehrerer numerischen Experimente und stellen die generierten Szenarien in der Benchmark-Datenbank CommonRoad öffentlich zugänglich zur Verfügung.

Contents

Chapter 1

Introduction

1.1 Motivation

By 2018, many of the major car companies announced the release of fully autonomous vehicles within less than four years[1]. Despite the investment of vast amounts of resources into research and development, this promise is still waiting to be realized, while development programs are canceled and timelines are shifted[2]. One of the major challenges, especially for their large-scale deployment, is that these vehicles are required to safely handle an unforeseeable number of situations. This large number is the result of the many possible behaviors of other traffic participants, road topologies, or environmental aspects that can be encountered. Moreover, most parameters of the real world have continuous domains, for which many testing approaches targeting discrete domains are not well suited. While for driver assistance systems with a low degree of autonomy existing approaches from the domain of functional safety or real-world driving tests can be conducted within reasonable financial budgets and time frames, new methods are required to validate the safety of highly autonomous systems that do not depend on human supervision and operate within more complex environments.

Simulative testing of autonomous vehicles in virtual environments expands the possibilities for their validation and enables driving mileages that greatly exceed those realizable in real-world tests. To assess the safety of a system with sufficient confidence, hundreds of millions of kilometers are required [11], which is intractable even when using simulations. However, most situations a vehicle encounters in traffic are repetitive and not challenging, hence do not provide valuable insights for its validation. Instead of focusing on the driven distance, it is more efficient to define a catalog of scenarios that aims at representing the situations the autonomous vehicle can encounter during its operation. This idea describes the essence of scenario-based testing [12]. Especially interesting in this context are edge cases, which are critical in terms of safety or other relevant metrics. Placing particular emphasis on such scenarios in the test strategy can reduce the number of simulations required to detect failures and can increase the confidence in the robustness of the system. However, finding safety-critical scenarios is especially challenging since they are typically rare in real-world datasets.

In this thesis, we present methods for automatically generating safety-critical test scenarios focusing on challenging motion planning algorithms. Our methods are not exclusive to testing of fully autonomous vehicles but can be applied during the development and validation of partially automated vehicles as well. For simplicity, we only use the term *autonomous*

[1] https://www.reuters.com/article/ctech-us-bmw-autonomous-self-driving-idCAKBN16N1Y2-OCATC (last accessed: 26.02.23)
https://www.reuters.com/article/us-autoshow-detroit-ford-motor-idUSKBN1F30YZ (last accessed: 26.02.23)
https://money.cnn.com/2017/11/30/technology/gm-autonomous-cars-2019 (last accessed: 26.02.23)

[2] https://europe.autonews.com/automakers/ford-vw-will-shut-argo-ai-self-driving-joint-venture (last accessed: 26.02.23)
https://edition.cnn.com/2022/11/01/business/self-driving-industry-ctrp (last accessed: 26.02.23)

vehicles from now on. Our methods address the following research questions:

- How can we create complex safety-critical test scenarios while ensuring the autonomous vehicle does not inevitably collide with obstacles?
- How can we generate a large number and variety of safety-critical traffic scenarios relying on virtual methods only?
- How can we ensure reactive, adversarial behavior of other traffic participants that comply with abstractly specified scenarios?

To this end, we propose optimization and search algorithms to create scenarios that uncover failures of the autonomous vehicle.

1.2 Safety Concepts for Motion Planning of Autonomous Vehicles

The methodologies for ensuring the safety of autonomous vehicles can be divided into different categories. Classical methods typically target at ensuring the functional safety of systems. Related standards such as ISO 26262 [13] typically require experts to manually define the failure modes and risks and how these are addressed in the system. While this is sufficient for systems with a lower degree of autonomy, ensuring safety for highly automated systems is more complex as these need to handle situations that are unforeseen at design time and consist of more tightly integrated components [14]. A standard addressing these issues is SOTIF [15].

Another approach is provided by formal methods, which can prove the safety of an autonomous system under reasonable assumptions on the behavior of other traffic participants considering, e.g., traffic rules or physical constraints. Formal methods then guarantee the correctness of a model with respect to a specification using mathematically rigorous proofs. In the case of autonomous vehicles, this transfers to guarantees for not causing a collision or violating traffic rules. Formal methods initially were proposed for problems of discrete nature, such as theorem proving or model checking but specifically for cyber-physical systems, concepts applicable to systems operating in continuous domains are developed to guarantee the safety of planning and control algorithms [16–18]. A comprehensive overview of such approaches is provided in [15].

While formal methods can provide safety guarantees, they can pose limitations to the software or hardware of the system. In contrast to formal methods, validation methods are used to demonstrate the safety of an autonomous system through testing, however without being able to provide strict guarantees. Depending on the level at which tests are performed, different methods are used: While on lower levels, classical methods from software testing, such as unit testing or fault injection [19], can be used, we focus on the system-level validation, particularly of motion planners. Specifically for autonomous vehicles, scenario-based approaches are becoming widespread for testing because they are able to structure the complexity of the environment in which the vehicles are going to be deployed. In our work, we especially focus on methods from this field, which we review in the next section.

During the development and approval process of autonomous vehicles, likely none of these methods is applied exclusively. Rather, approaches from all three categories are used in parallel to complement each other since even formally safe systems will not be deployed without a validation in simulation to rule out coding or modeling errors.

1.3 Scenario-Based Testing of Autonomous Vehicles

In scenario-based testing, the objective is to define a scenario catalog that represents the operational design domain (ODD) of the system under test (SUT). The ODD defines the envelope of the situations that the system is intended to handle safely [20]. The scenarios in this catalog are usually structured into several abstraction levels that we review in Sec. 1.3.1 followed by a summary of formats to represent such scenarios in Sec. 1.3.2 and methods for their parameterization in Sec. 1.3.3. To select scenarios and to evaluate the performance of the SUT, different metrics are used for quantifying, e.g., safety or other performance indicators such as passenger comfort. In Sec. 1.3.4, we review such metrics focusing on the criticality of scenarios. We conclude this section by summarizing methods for testing and falsifying autonomous systems in Secs. 1.3.5 and 1.3.6.

Scenario-based testing is not necessarily limited to virtual testing and can integrate real-world test drives [21] as well. However, in this work, we focus on simulative approaches. For a broader overview of scenario-based testing, we refer interested readers to surveys such as [12, 20, 22].

1.3.1 Abstraction Levels for Scenarios

To structure the validation of autonomous vehicles, the test scenarios are typically organized along different abstraction levels as proposed in [23, 24]. The authors propose the terms functional, logical, and concrete scenarios with decreasing abstraction levels as depicted in Fig. 1.1. The exact definitions and notations for the abstraction levels vary in the literature, however, on the abstract level, the scenarios typically describe the maneuvers and relations between the traffic participants on a semantic level, while in concrete scenarios, either the parameters required for the execution in a simulation or the exact trajectories of the surrounding traffic participants are provided [20].

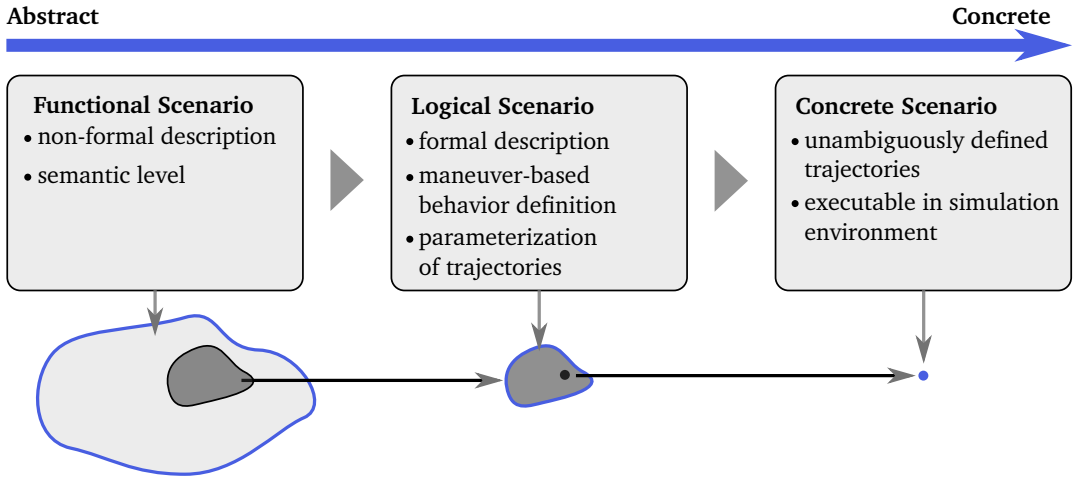


Figure 1.1: Abstraction levels for defining test scenarios to structure the scenario space that is visualized below.

To create scenario catalogs, there are in general two directions proposed: knowledge-based approaches first create scenarios on the most abstract level and detail the scenarios representations yielding concrete scenarios [25–27]. Vice-versa, data-driven approaches start with concrete traffic scenarios obtained e.g., from data recorded in real-world traffic or simulation. These scenarios are assigned to abstract scenario representations [28, 29], clustered

using hand-crafted features [30, 31] or using unsupervised learning [32, 33]. The advantage of knowledge-based approaches is that they provide a structured approach for the validation of autonomous vehicles. Challenges arise when scaling this approach to large ODDs, e.g., for urban driving, which requires methods for creating and identifying relevant scenarios from a large database of abstract scenarios. The question of the relevance of scenarios can be partly answered by data-driven methods that provide insights into probabilities as described in Sec. 1.3.5. Furthermore, relying on expert knowledge like in knowledge-based approaches might lead to an incomplete scenario catalog, especially when it comes to edge cases [33]. Challenges of data-driven approaches arise from the costly data acquisition and processing of a large amount of data. Furthermore, when using unsupervised learning to cluster scenarios, problems might emerge from a lack of interpretability of the generated clusters.

1.3.2 Scenario Representation Formats

Formats for representing scenarios of any abstraction level need to consider several aspects: The scenario should be machine-readable for an automated generation or execution in simulations but should be interpretable by humans as well. Ideally, scenarios are defined unambiguously so that simulations are consistent across different simulation tools.

To formalize the representation of traffic scenarios, several file formats, and domain-specific languages have been proposed [34–40]. One of the first scenario formats was OpenScenario focusing on concrete scenarios while its successor OpenScenario 2.0 expands towards more abstract scenario representations[3]. However, it is still not fully supported by most of the prevalent simulators such as CARLA [41]. The domain-specific language Paracosm is closely integrated with a Unity-based simulator focusing on visual and simulating interaction with other traffic participants. It comes with a tool to sample for probabilistic and combinatorial coverage considering parameter distributions or sets [42]. Scenic, a similar language with integrated simulation and test generation tools, focuses on scenario parameterizations based on probability distributions and corresponding sampling techniques [43]. The format SceML focuses on modeling abstract traffic scenarios [39] and provides an interface to the simulator CARLA. For a more in-depth comparison of scenario representations, we refer to the survey [44].

1.3.3 Scenario Parameterization

To derive concrete scenarios from abstract representations or when generating variations of concrete scenarios, there exist many different methods for parameterizing the behavior of the traffic. The choice of parameterization can have a major influence on the efficiency and capabilities of the test methods that build upon the scenarios. The parameterization needs to balance between two conflicting objectives: on the one hand, the parameterization should provide a sufficient degree of flexibility to allow the test method to cover a large variety of different scenarios, to avoid excluding relevant edge cases. On the other hand, the parameterization should not define too many decision variables for the simulation, as this increases the complexity of the testing procedure making it less efficient. Depending on the use case of the testing procedure, another factor to consider is the degree to which the generated scenarios reflect realistic behavior of traffic participants. We divide the parameterization techniques into the following categories:

[3] https://www.asam.net/standards/detail/openscenario

- **Rule-based models**: Rule-based driver models are typically implemented in traffic simulation tools such as CARLA [41] or SUMO [45] to model human driving behavior. Examples of these models are the intelligent driver model (IDM) [46], which controls the acceleration of the vehicle, or MOBIL [47] for modeling lane-change decisions. Behavioral parameters of the models can be chosen as variables for deriving scenarios. While simulations with such models are simple to create, the resulting scenarios often do not provide a large behavioral variety due to the simplistic underlying models. As an additional downside, the models are typically optimized towards ensuring safety and preserving traffic flow, which prevents them from exposing safety-critical behavior that challenges the SUT.

- **Learning-based models:** As reviewed in more detail in Secs. 1.3.5 and 1.3.6, another approach is to learn traffic behavior using machine-learning techniques or by computing probabilistic models. These models are then used to synthesize new concrete scenarios. This parameterization is typically used in combination with probabilistic sampling methods. One advantage of this approach is that the resulting scenarios tend to be realistic and diverse in terms of behavior.

- **Maneuver primitives**: Maneuver primitives are used for parameterizing certain maneuvers by providing short sequences of motions, e.g., lane changes described by quintic polynomials. Such primitives can also be fitted to naturalistic trajectories in order to create variations of recorded data [48]. While maneuvers of the traffic participants can be defined explicitly, the resulting motion is limited to the shape defined by the primitive.

- **Conditional triggers**: Especially domain-specific languages for traffic scenarios (cf. Sec. 1.3.2) specify the timing of actions using conditional triggers. Parameters in this case are usually the durations or variables of the conditions. These triggers can be combined, e.g., with maneuver primitives to model more complex behavior.

1.3.4 Criticality Metrics

Criticality metrics assess the criticality of a traffic scenario. For the validation of autonomous vehicles, the metrics can be used, e.g., to select test scenarios or to assess the behavior of a SUT in terms of safety and provide more insights compared to event-based metrics, such as accident rates or traffic rule violations. Apart from validation, these metrics are also used as an input for driver assistance systems. Classical metrics typically estimate the remaining distance or the time until a critical event occurs, such as time-to-collision (TTC), time-headway (THW), and similar variants [49]. These metrics require assumptions on the movement of all traffic participants including the ego vehicle. For the TTC, the prediction assumes a constant-velocity motion or another concrete trajectory for the future behavior. Hence, these metrics are limited to certain behaviors and do not provide a holistic threat assessment of a traffic scenario that covers the variety of its possible outcomes. Similar metrics that are especially relevant for motion planning algorithms are the time-to-react (TTR) [50] and related metrics [51] that provide the time available for starting a collision-avoidance maneuver. One common characteristic of the metrics mentioned so far is that each of them typically focuses on a single type of threat. Especially for the validation of autonomous vehicles, a more holistic assessment of the criticality is required. This can be achieved, e.g., by considering multiple specialized metrics simultaneously. A method based on optimal control [52,53] uses the minimal system inputs for the ego vehicle for collision-free driving as an indicator of criticality.

Thus, the considered motion of the ego vehicle is not restricted to a single trajectory. For more detailed reviews, we refer interested readers to [49, 54].

1.3.5 Test Methods

The objective of test methods typically is to simulate a SUT in concrete scenarios to evaluate its performance using defined metrics. These approaches are typically non-reactive in the sense that the process of selecting concrete scenarios is not informed about the concrete model of SUT or its performance during testing. Thus, the scenario catalog does typically not depend on the SUT but on the test strategy. Since the test scenarios are independent of the SUT, the approaches are suitable for benchmarking the performance of different SUTs or different versions of its software. The approaches for selecting test scenarios typically aim at reducing the number of scenarios required to obtain a certain test result. We provide an overview of strategies we divide into the following two categories.

Coverage-driven testing

When creating test suites of scenarios, there are different objectives that can be followed. One objective is to use coverage-driven metrics aiming to reduce the number of scenarios to be tested. Coverage-driven methods are typically used in knowledge-based approaches where concrete scenarios need to be derived from logical scenarios that define parameter ranges. For continuous parameters such as positions or velocities, the number of possible concrete scenarios is typically infinite. However, also for parameters with discrete domains, the number of possible parameter combinations renders exhaustive testing procedures infeasible. For example, a logical scenario defined by 8 parameters with 10 values each already has 10^8 possible scenario concretizations. To reduce the number of scenarios for testing, coverage-driven testing methods aim to replace the need for exhaustive testing through the definition of coverage measures for the scenario catalog for which fewer test cases are sufficient.

Combinatorial testing methods define test cases for a set of parameters each having a finite number of values. K-wise testing is based on the assumption that failures result from the combination values from critical combinations of a fixed number, i.e., k parameters. Hence, the goal is to define a set of test cases where each k-wise combination of parameters is represented. This principle is applied to the domain of autonomous vehicles in [55, 56]. Even when creating k-wise test suites, the number of test cases is often still uneconomically large when applied to practical applications for autonomous vehicles [57]. Hence, ideas to reduce the number of test cases are proposed [57, 58]. Apart from k-wise testing, other approaches focus on the coverage of high-level behavior in a test suite [59, 60].

Probabilistic Sampling Methods

Another possible objective when creating scenario databases is to obtain realistic distributions of scenarios that reflect their occurrence in real-world driving. These scenario databases can be used especially to derive statistical evidence about the performance of an SUT in the real world, e.g., the probability of accidents. In this case, typically recordings of naturalistic driving data are used and the concretization of scenarios is posed as a problem of sampling from the scenario database. One challenge with naturalistic driving databases is that they are typically imbalanced since the majority of driving data does not contain safety-critical scenarios or other challenging edge cases as illustrated in Fig. 1.2 using the example of the time-to-collision. Thus, performing simulations with scenarios sampled using uniform probability distributions is inefficient. To reduce the number of simulations while preserving probabilistic

properties, importance sampling techniques are proposed [61–63]. Sampling-based methods can further be used to generate previously unseen, yet realistic, scenarios using Bayesian networks [64, 65]. The authors of [66] present a technique that allows sampling from probabilistic distribution considering linear constraints that allow specifying behaviors of traffic participants in logical scenarios. Other works use deep learning to learn realistic models of traffic behavior that can be used for creating new scenarios [67, 68], or train models with a focus on maximizing the diversity of the traffic behavior [69].

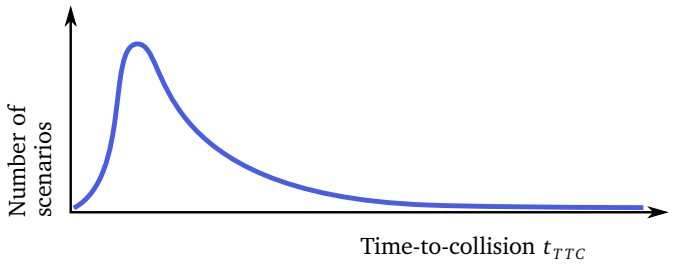


Figure 1.2: Typical distribution of the time-to-collision t_{TTC} among scenarios in a naturalistic dataset where safety-critical scenarios with a t_{TTC} close to zero are rare. Figure adapted from [70] (original figure licensed under Creative Commons License CC BY 4.0).

1.3.6 Falsification Methods

In contrast to testing, falsification methods are informed about the performance of the SUT and adapt the test scenarios in a closed loop with the SUT to uncover failures with respect to its specification. Therefore, falsification is often more efficient when it comes to uncovering failures in the sense that it requires less concrete scenarios to find failures compared to test methods. Contrary to the test strategies presented above, falsification is less suitable for supporting claims about the overall performance of an SUT, e.g., through statistical evaluations. Detected failures can be used, e.g., during the development to improve the software. However, since the resulting selected scenarios always depend on the SUT, falsification is less suitable than test methods for benchmarking different parameterizations of an SUT or entirely different systems.

For cyber-physical systems in general, there exists a variety of approaches to falsify a SUT with respect to a specification typically formulated using temporal logic. Failures can be found by formulating the falsification problem as an optimization problem, where the objective is to minimize the robustness of the solution with respect to the specification. The optimization problem can be solved using probabilistic sampling methods [71], multiple-shooting methods involving graph-search [72], hierarchical search methods combining Monte Carlo tree search (MCTS) with continuous solvers, or reinforcement learning [73]. An extensive overview of such algorithms can be found in [74].

When it comes to autonomous vehicles, applying such falsification methods is challenging due to the complexity introduced by the variety of the possible behaviors of the environment, where failures can be the result of interactions between multiple traffic participants. When using optimization-based approaches, the cost functions thus exhibit local minima, which is addressed by using dedicated solvers such as evolutionary algorithms [75–79], simulated annealing [80], Bayesian optimization [78, 81], or explorative search algorithms such as rapidly-exploring random trees (RRT) [82, 83]. These approaches typically involve cost functions based on criticality metrics (cf. Sec. 1.3.4) or define requirements formalizing the desired behavior of the SUT using temporal logic [76, 84, 85]. In [86], rules for other traffic

participants are encoded using temporal logic and integrated into a reinforcement learning algorithm to control their degree of adversariality and to ensure that collisions are not caused by rule violations of other traffic participants. Another approach is to learn adversarial models for the traffic behavior through reinforcement learning [87, 88]. Traffic behavior learned from real data is used to generate realistic collision scenarios in [70, 89, 90]. When considering the likelihood of failures, the falsification problem is formulated as a Markow decision process, which is referred to as *adaptive stress testing* [91]. The goal of this approach is to find the most likely failure.

Another line of work aims at computing safe invariant sets for a SUT and sampling potentially safety-critical scenarios from the boundary of these sets [92, 93]. Similarly, scenarios are sampled in [94]. Most falsification methods consider the model as a black box as it is too complex for considering it explicitly. Surrogate models are approximations of complex SUTs with a less complex structure that can be considered explicitly to generate counterexamples. In [95, 96] such models are learned on the fly during the falsification and are utilized to generate counterexamples.

While approaches using a broad problem formulation might uncover unexpected failures, they are less suited for systematic testing in a scenario-based framework and might as well miss failures when getting trapped in local minima. Hence, other approaches are proposed that include scenario specifications in the problem formulation, e.g., by including compliance with specified behavior in the cost function [97]. An approach using an SMT solver to find potential collision scenarios based on a constraint-based scenario specification is presented in [98]. A falsification tool using scenarios described in the scenario language Scenic is presented in [99].

1.4 Research Gaps and Contributions

In the reviewed literature, we identify two gaps that we mainly address in this work:

1. Approaches for testing autonomous vehicles mostly rely on existing scenarios and typically differ by the method for sampling the scenarios either using coverage-driven or probabilistic methods. These sampling strategies do not take into account a specific SUT. Identifying safety-critical scenarios independently of an SUT is not straightforward in general; most approaches rely on using a concrete SUT, which places them rather in the category of falsification methods. Methods for generating complex, system-agnostic safety-critical scenarios created using a generalizable criticality metric are missing in the literature.

2. As described in Sec. 1.3.6, falsification methods for autonomous vehicles typically require a specification of the desired behavior of the SUT. Only little focus is put on performing falsification in a scenario-based setting, which requires integrating methods to synthesize behavior complying with a scenario specification with falsification algorithms.

This dissertation proposes new algorithms to generate traffic scenarios for testing software for autonomous vehicles focusing on motion-planning algorithms. The developed methods advance the state of the art of the automated validation through the following contributions:

The method presented in Sec. 3.1 extends the initial idea to use the drivable area for the generation of traffic scenarios [100] in order to enable the efficient optimization of complex scenarios with multiple traffic participants. This is achieved through evolutionary optimization algorithms and our method to prune irrelevant regions from the parameter space of the

scenarios. Due to the utilized metric, the scenarios are generated independently of a specific SUT since we consider all feasible trajectories that could solve the planning problem of a scenario. Since we quantify the solution space using our metric, we are able to directly add constraints to the optimization problem so that there is a valid solution for the scenarios we generate. Hence, we avoid generating scenarios that do not provide meaningful insights into the capabilities of the SUT, e.g., where a collision is inevitable. Our method generalizes to a wide range of traffic scenarios as we demonstrate in Sec. 3.2, where we combine the scenario optimization with the automatic extraction of high-definition maps from OpenStreetMap[4] and traffic simulation to obtain a fully automated framework for generating safety-critical test scenarios.

To enable the efficient implementation of the scenario generation, we propose in Sec. 3.3 an approach that increases the efficiency of computing the reachable sets of vehicles in traffic scenarios. By storing the reachability of subsets in a graph and reusing it during runtime, we avoid repetitive set operations. Compared to previous work, our method requires significantly less computation time.

Our method developed in Sec. 4.1 is the first to formulate the synthesis of concrete traffic scenarios as a mixed-integer optimization problem. Hence, it enables checking whether a concrete scenario can be synthesized from an abstract scenario. Our approach presented in Sec. 4.2 enables the falsification of motion planning algorithms by finding scenario concretizations synthesized from abstract scenarios. By combining reachability analysis with search algorithms that sample within the reachable sets, we only consider the solution space of feasible scenario parameters with respect to scenario specifications. Thus, unlike previous work, we do not need to detect infeasible parameters by executing computationally costly simulations. Furthermore, we show that our falsification algorithm becomes more efficient the more complex the scenario becomes due to the shrinking solution space from which we need to sample.

In addition to the methodological contributions, the software originating from this work was also published as part of several open-source tools for map conversion, traffic simulation using SUMO, and reachability analysis as part of the CommonRoad project[5]. The scenarios generated using the presented methods contribute to the CommonRoad scenario database [35] for benchmarking motion planning algorithms of autonomous vehicles.

[4] https://www.openstreetmap.org
[5] https://commonroad.in.tum.de

1.5 Publications and Outline of the Thesis

Next, we provide an outline of the thesis and refer to the included publications. We start in Ch. 2 by introducing the relevant preliminaries providing the background for the problem statements that are provided subsequently. We introduce in [1] the optimization-based method for generating safety-critical test scenarios using a criticality metric based on the drivable area. In [2], we extend the framework to automatically generate a large number of diverse scenarios by combining automated map sourcing, traffic simulation, and the scenario optimization. The chapter closes with our work Ch. 3 presenting a new method for computing the drivable area of road vehicles efficiently [3]. In Ch. 4, we present the methods introduced in [4] for synthesizing scenarios based on abstract scenario specifications, which are subsequently integrated in [5] with a falsification algorithm. We conclude the thesis in Ch. 5 and propose directions for future research building on our work.

Included Publications

This thesis is composed of the following works published as peer-reviewed journal and conference articles.

[1] **M. Klischat** and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 2352–2358.

[2] **M. Klischat***, E. I. Liu*, F. Höltke, and M. Althoff, "Scenario Factory: Creating safety-critical traffic scenarios for automated vehicles," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, 2020, pp. 2964–2970.

[3] **M. Klischat** and M. Althoff, "A multi-step approach to accelerate the computation of reachable sets for road vehicles," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, 2020, pp. 2306–2312.

[4] **M. Klischat** and M. Althoff, "Synthesizing traffic scenarios from formal specifications for testing automated vehicles," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 2065–2072.

[5] **M. Klischat** and M. Althoff, "Falsifying motion plans of autonomous vehicles with abstractly specified traffic scenarios," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1717–1730, 2023.

Excluded Publications

The following works comprise publications on software tools partly originating from the included publications above and are not included in this thesis.

[6] **M. Klischat**, O. Dragoi, M. Eissa, and M. Althoff, "Coupling sumo with a motion planning framework for automated vehicles," in *SUMO User Conference*, 2019, pp. 1–9.

[7] N. Kochdumper, F. Gruber, B. Schürmann, V. Gaßmann, **M. Klischat**, and M. Althoff, "AROC: A toolbox for automated reachset optimal controller synthesis," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control (HSCC)*, 2021, pp. 1–6.

[8] S. Maierhofer*, **M. Klischat***, and M. Althoff, "CommonRoad Scenario Designer: An open-source toolbox for map conversion and scenario creation for autonomous vehicles," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pp. 3176–3182.

[9] B. Schürmann, **M. Klischat**, N. Kochdumper, and M. Althoff, "Formal safety net control using backward reachability analysis," *IEEE Transactions on Automatic Control*, vol. 67, no. 11, 2022.

[10] E. I. Liu*, G. Würsching*, **M. Klischat**, and M. Althoff, "CommonRoad Reach: A toolbox for reachability analysis of automated vehicles," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, 2022, pp. 2313–2320.

* These authors have contributed equally to the work.

Chapter 2

Preliminaries and Problem Statements

In this section, we introduce the definitions required for formulating the main problems addressed in this thesis.

2.1 Scenario Representations

We use two abstraction levels for traffic scenarios: abstract scenarios and concrete scenarios. We consider *abstract scenarios* describing the behavior of the traffic participants semantically on the spatiotemporal maneuver level using a scenario specification ϕ_s. Less formally, an abstract scenario formalizes, e.g., a scenario verbalized by "*The ego vehicle is driving behind vehicle B, then merges to the left lane and overtakes vehicle B.*". In Ch. 4, we provide a formal definition of the format we developed to specify abstract scenarios. In a *concrete scenario*, the trajectories $x_i(k) \in \mathbb{R}^v$ with $i \in [1, n]$ denoting the index of a traffic participant are provided for a time horizon k_f, i.e., for $k \in [0, k_f]$. The state of the SUT is represented by $x_{ego}(k)$. We summarize the states of all traffic participants and the SUT in the stacked vector $x^S(k) = [x_{ego}(k), x_1(k), \ldots, x_n(k)]^T$ and further denote the trajectories of a concrete scenario defined over a time interval $[0, k_f]$ by $x^S([0, k_f])$. Hence, the motion of all traffic participant in a scenario is unambiguously defined by $x^S([0, k_f])$. A concrete scenario is said to be derived from an abstract scenario if it complies with the abstract specification, i.e., $x^S([0, k_f]) \models \phi_s$. We say that x_s entails ϕ_s, denoted by $x^S([0, k_f]) \models \phi_s$, if x_s satisfies the constraints of scenario ϕ_s. In other words, $x^S([0, k_f]) \models \phi_s$ if and only if there is no possible interpretation in which the trajectory belongs to a scenario and the constraints of the corresponding specification ϕ_s are not fulfilled.

In our test scenarios, we assume for the provided SUT a *black-box* property, i.e., that we can only execute the system and observe its outputs for our provided inputs.

Definition 1 (System Under Test):
The system under test (SUT) is represented by a model $x_{ego}(k+1) = M_{SUT}(x^S(k))$ that returns the next state of the ego vehicle based on the state of the scenario. □

This assumption is also reasonable in case we have full insights about the system but the system model is too complex for explicitly exploiting these insights. For our test scenarios, we consider the task of the SUT to plan a collision-free trajectory starting at an initial state $x_{ego}(0)$.

2.2 Reachability Analysis

In this work, we use different notions of reachable sets for the two problem statements introduced in Sec. 2.3. For Problem 1, we consider the collision-free reachable set of the SUT

starting from an initial state $x_{\text{ego}}(0) \in \mathbb{R}^v$. When computing the reachable set, we consider a vehicle model $x_{\text{ego}}(k+1) = f(x_{\text{ego}}(k), u_{\text{ego}}(k))$ with inputs $u_{\text{ego}}(k) \in \mathcal{U} \subset \mathbb{R}^w$ where $\mathcal{U}$ denotes the set of admissible inputs. We use the notation $\chi\big(k, u_{\text{ego}}([0,k]), x_{\text{ego}}(0)\big)$ for a state that is reached starting from $x_{\text{ego}}(0)$ by applying the inputs $u_{\text{ego}}([0,k])$.

We further introduce the operator $\mathsf{occ} : \mathbb{R}^v \to \mathbb{R}^2$ returning the occupied area of a traffic participant in a state $x_i(k)$. The road area, on which the ego vehicle is allowed to drive, is provided by $\mathcal{W}_{\text{lane}} \in \mathbb{R}^2$. We use the reachable set to represent the solution space of a motion planning algorithm starting at the initial state $x_{\text{ego}}(0)$.

Definition 2 (Collision-Free Reachable Set):
We define the set containing all trajectories that are feasible with respect to the vehicle dynamics and not colliding with the occupancies of other vehicles as the *reachable set*.

$$\begin{aligned}\mathcal{R}^{\text{drivable}}(k, \mathcal{U}, x_{\text{ego}}(0), x^{\text{S}}([0,k_{\text{f}}]), \mathcal{W}_{\text{lane}}) = \Big\{ \chi\big(k, u_{\text{ego}}([0,k_{\text{f}}]), x_{\text{ego}}(0)\big) \;\Big|\; \\ \forall \kappa \in [0,k_{\text{f}}], \exists u_{\text{ego}}(\kappa) \in \mathcal{U} : \\ \mathsf{occ}\big(\chi\big(\kappa, u_{\text{ego}}([0,k_{\text{f}}]), x_{\text{ego}}(0)\big)\big) \subseteq \mathcal{W}_{\text{lane}} \setminus \bigcup_i \mathsf{occ}\big(x_i(\kappa)\big) \Big\}. \quad (2.1)\end{aligned}$$

□

To quantify the size of the reachable set, we use the drivable area, which we define using the operator $\underset{\text{pos}}{\mathsf{proj}} : \mathbb{R}^v \to \mathbb{R}^2$ returning the projection of the reachable set to the position domain.

Definition 3 (Drivable Area):
We denote the projection of the reachable set to the position domain as the *drivable area*

$$\mathcal{D}\big(k, \mathcal{U}, x_{\text{ego}}(0), x^{\text{S}}([0,k_{\text{f}}]), \mathcal{W}_{\text{lane}}\big) = \underset{\text{pos}}{\mathsf{proj}}\Big(\mathcal{R}^{\text{drivable}}\big(k, \mathcal{U}, x_{\text{ego}}(0), x^{\text{S}}([0,k_{\text{f}}]), \mathcal{W}_{\text{lane}}\big)\Big).$$

□

The size of the drivable area at each time step yields the drivable area profile

$$A\big(k, \mathcal{U}, x_{\text{ego}}(0), x^{\text{S}}([0,k_{\text{f}}]), \mathcal{W}_{\text{lane}}\big) = \mathsf{area}\Big(\mathcal{D}\big(k, \mathcal{U}, x_{\text{ego}}(0), x^{\text{S}}([0,k_{\text{f}}]), \mathcal{W}_{\text{lane}}\big)\Big),$$

which we obtain using the operator $\mathsf{area} : \mathbb{R}^2 \to \mathbb{R}^+$ that returns the size of the drivable area.

When being provided with a scenario specification ϕ_{s} from which we want to derive a concrete scenario $x^{\text{S}}([0,k_{\text{f}}])$, this specification restricts not only the solution space of the trajectory of the ego vehicle but of the trajectories of all other traffic participants as well. To represent the combined solution space of all traffic participants, we define the combined reachable set of all vehicles, which is compliant with a provided scenario specification ϕ_{s}. This yields the following definition we use in Problem 2, which generalizes $\mathcal{R}^{\text{drivable}}$.

Definition 4 (Specification-Compliant Reachable Set for Multiple Traffic Participants):
We define

$$\begin{aligned}\mathcal{R}^{\text{spec}}(k, \mathcal{U}^{\text{S}}, \mathcal{R}_0, \phi_{\text{s}}) = \Big\{ \chi\big(k, u^{\text{S}}([0,k]), x^{\text{S}}(0)\big) \;\Big|\; x^{\text{S}}(0) \in \mathcal{R}_0, \\ \forall \kappa \in [0,k_{\text{f}}], \exists u^{\text{S}}(\kappa) \in \mathcal{U}^{\text{S}} \colon \chi\big(\kappa, u^{\text{S}}([0,\kappa]), x^{\text{S}}(0)\big) \models \phi_{\text{s}} \Big\} \quad (2.2)\end{aligned}$$

containing all states compliant with a specification ϕ_{s} for the time interval $[0,k_{\text{f}}]$ as the *specification-compliant reachable set*. □

2.3 Problem Statements and Overview of Methods

2.3.1 Generation of Non-Reactive Test Scenarios

The first problem we address is the generation of safety-critical test scenarios. Following a data-driven approach for defining scenarios, we consider provided concrete scenarios for which we optimize the behavior of the traffic participants with respect to a criticality metric $C(x_{\text{ego}}(0), x_i([0,k_\text{f}])) \in \mathbb{R}^+$ that assesses the criticality considering the trajectories of all traffic participants and the initial state of the SUT. The initial scenarios can be taken, e.g., from databases with real-world data or from simulation.

To optimize the scenarios, we consider parameters $p \in \mathbb{R}^o$ parameterizing the trajectories $x_i([0,k_\text{f}])$ of the traffic participants in a provided scenario. This yields the parameterized trajectories $x_i^*([0,k_\text{f}],p)$.

Problem 1 (Optimization of System-Independent Traffic Scenarios):
Our goal is to determine the scenario parameters p that solve the optimization problem

$$\begin{aligned} \arg\min_p \quad & C(x_{\text{ego}}^*(0,p), x_i^*([0,k_\text{f}],p)) \\ \text{subject to} \quad & \forall k, \forall i, \forall j \neq i: \quad \text{occ}(x_i^*(k,p)) \cap \text{occ}(x_j^*(k,p)) = \emptyset \end{aligned} \tag{2.3}$$

where we consider as a constraint that the other traffic participants do not collide with each other.

□

One of our goals is to optimize the scenarios not considering a specific SUT but arbitrary behavior of an SUT starting from an initial state $x_{\text{ego}}(0)$ instead. Hence, our approach is based on a criticality metric using the drivable area profile $A(k, \mathcal{U}, x_{\text{ego}}(0), x_i^*([0,k_\text{f}],p), \mathcal{W}_{\text{lane}})$, which only depends on the initial state and abstracts all possible behaviors of arbitrary SUTs for the remaining time interval of the scenario through the reachable set. From now on, we use the simplified notation $A(k,p)$. Given there is a critical area profile $A_{\text{crit}}(k)$, we compute the difference between the actual profile $A(k,p)$ and the critical profile using

$$C(x_{\text{ego}}^*(0,p), x_i^*([0,k_\text{f}],p)) = \sum_{k=0}^{k_\text{f}} \big(A(k,p) - A_{\text{crit}}(k)\big)^2. \tag{2.4}$$

This metric is based on the assumption that the criticality of a scenario is related to the size of the solution space of feasible trajectories for the SUT. In a small solution space, it can be more difficult for planning algorithms to find a trajectory, e.g., when using sampling-based motion planning algorithms that discretize the search space and might fail at identifying narrow passages between obstacles. A small solution space can also indicate that a vehicle is surrounded by more obstacles that need to be considered simultaneously when planning the trajectory.

The principle of the optimization problem is illustrated in Fig. 2.1 depicting a mostly unrestricted drivable area of the SUT before the optimization and the drivable area after the optimization, which calls for an evasive maneuver of the SUT. A solution to the optimization problem requires solving a multi-agent planning problem where the motions of the other traffic participants need to be coordinated so that other traffic participants jointly affect the drivable area. Additionally, we need to consider constraints regarding the collision avoidance between other traffic participants to exclude unrealistic scenarios. With the criticality metric as the objective function, we obtain a highly nonlinear problem, because the size of the drivable area depends nonlinearly on the road geometry, the initial state, and the interaction

with other traffic participants. To solve the optimization problem for a large number of scenarios, we further require efficient methods for computing the drivable area. However, this is challenging due to the non-convex shape of the reachable set and the constraints that need to be considered.

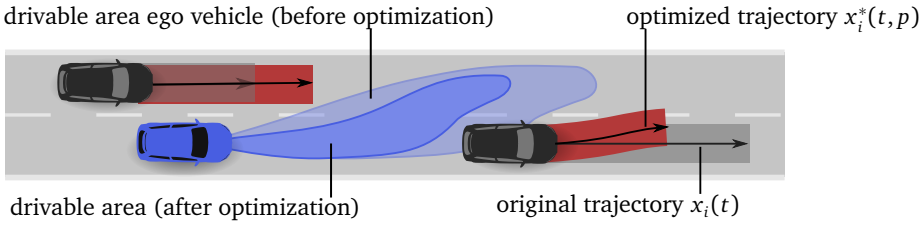


Figure 2.1: In Problem 1, we want to find trajectories of other traffic participants to obtain a critical size of the drivable area reducing the solution space for an SUT to find a feasible trajectory. This example depicts the drivable area before and after the optimization of the scenario.

The approach we present in our work addresses the aforementioned challenges using the following methodologies:ability analysis to compute the spe To solve the optimization problem, we propose using evolutionary algorithms with a local correction scheme to correct infeasible scenario candidates using efficient convex optimization. We handle the complexity of the optimization problem through a lane-aligned parameterization of the trajectories. By superimposing the initial trajectories of the traffic participants with a linear motion model, we preserve the behavioral characteristics of the initial trajectories and can formulate linear constraints for the correction using convex optimization. To increase the efficiency of the optimization, we develop an algorithm that identifies the relevant parameter ranges where the motion of other traffic participants interferes with the drivable area of the ego vehicle. The parameter space is pruned to avoid exploring irrelevant scenario candidates during the optimization. In addition, we develop a method for computing reachable sets efficiently, which enables reducing the computation time of the optimization problem.

Abstracting the SUT using the drivable area has several advantages. The resulting scenarios are independent of the SUT, which is only exposed to the scenarios after the optimization as depicted in Fig. 2.2. At the same time, we can simultaneously consider all feasible maneuvers the SUT can choose from and aim for scenarios in which a feasible solution for the SUT exists by demanding $\forall k : A(k,p) > 0$. The scenarios are therefore system-agnostic and can be used to benchmark different SUTs, i.e., entirely different systems but also different versions or parameterizations of the same system. While there are many approaches that consider the probability of scenarios, c.f., Sec. 1.3.5, we deliberately focus with our metric on the physical feasibility of a scenario. We argue that in order to achieve the high confidence level required for the homologation of autonomous vehicles, even highly unlikely scenarios need to be handled safely.

The resulting scenarios are non-reactive in the sense that they do not adapt to the behavior SUT. Testing using non-reactive scenarios is more efficient compared to reactive scenarios since they do not need to be generated again for every SUT. Since the drivable area can be computed for arbitrary road layouts and trajectories of the traffic participants, it can be applied as a criticality metric even to complex traffic scenarios. A downside of generating scenarios non-reactively is that they can be less efficient when searching for specific failures. This case is addressed in the next problem statement.

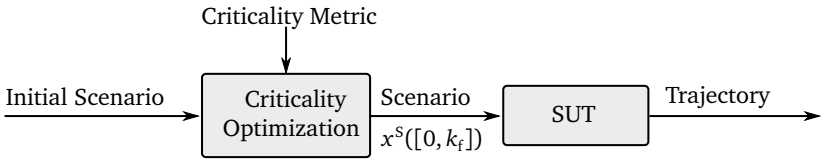


Figure 2.2: Problem 1: open-loop testing using SUT-agnostic scenarios.

2.3.2 Falsification using Reactive Scenarios

The second part of this work considers the synthesis of scenarios to falsify specific SUTs. In contrast to Problem 1, we obtain reactive scenarios where we adapt the behavior of the other traffic participants in a scenario in a closed loop with a SUT as shown in Fig. 2.3. Our objective is to find a concrete scenario that uncovers a failure of the SUT $M_{\text{SUT}}(x^S(k))$, which is provided to our falsification algorithm. The other input to the algorithm is the abstract scenario specification ϕ_s, which we divide into a maneuver specification ϕ_e describing the behavior of all traffic participants and a failure specification ϕ_f that describes the failure of the SUT we consider in our search:

$$\phi_s = \phi_e \wedge \phi_f.$$

The maneuver specification ϕ_e enables embedding our method into a scenario-based approach, while the failure specification enables using falsification methods that actively search for the failures. Since we focus on motion planning algorithms, a failure can comprise, e.g., a collision with another traffic participant or a violation of traffic rules. The specification can be part of the knowledge-driven approach described in Sec. 1.3.1, in which concrete scenarios derived from abstract scenarios are part of the test strategy. In this work, we assume the specification is already provided by methods described in Sec. 1.3. We summarize the problem as follows.

Problem 2 (Falsification of a System Under Test Using Specified Traffic Scenarios):
We consider the problem of finding a concrete scenario $x^S([0, k_f])$ that complies with ϕ_s. □

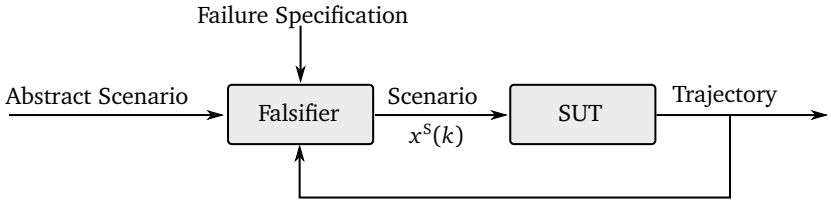


Figure 2.3: Problem 2: Falsification of a specific SUT. The behavior of other traffic participants complies with a specification ϕ_s.

This problem formulation requires synthesizing trajectories of the other traffic participants that comply with the specification of their behavior ϕ_e. In this case, we assume the trajectories $x_i(k)$ to be related to the inputs $u_i(k) \in \mathbb{R}^w$ through a discrete-time vehicle model $x_i(k+1) = f_M(x_i(k), u_i(k))$. While the falsification algorithm can directly control the other traffic participants, it cannot control the SUT with the trajectory $x_{\text{ego}}(k)$. Therefore, we use a search strategy to find a concrete scenario in which the behavior of the other traffic participants results in a failure of the SUT defined by ϕ_f. Challenges arise from the high dimensionality when planning the actions of agents simultaneously. We use Monte Carlo tree search

(MCTS), which is well-suited for search problems with high branching factors, to solve the falsification problem. To reduce the branching factor we propose using specification-compliant reachable sets $\mathcal{R}^{\text{spec}}(k, \mathcal{U}^{\text{S}}, \mathcal{R}_0, \phi_{\text{s}})$ to adaptively constrain the search space to areas where ϕ_{s} can be fulfilled. This prevents the search from exploring concrete scenarios that cannot possibly end in a compliant scenario at a later point in time resulting in a more efficient falsification.

To compute $\mathcal{R}^{\text{spec}}(k, \mathcal{U}^{\text{S}}, \mathcal{R}_0, \phi_{\text{s}})$, we develop a constraint-based representation of the scenario specification ϕ_{s}, which we integrate with reachability analysis to compute the specification-compliant reachable set $\mathcal{R}^{\text{spec}}(k, \mathcal{U}^{\text{S}}, \mathcal{R}_0, \phi_{\text{s}})$ of all traffic participants in a scenario. The reachable set is updated at each sampling step during the MCTS and from the set we derive constraints that exclude incompliant actions of other traffic participants. Fig. 2.4 illustrates how the trajectories of three vehicles are sampled within specification-compliant reachable sets. In our specification, we consider the scenario specification to be temporally structured into successive scenes, which describe the spatial relations of traffic participants using sets of predicates. The predicates hold true for the entire duration of the scene and define, e.g., the interval of the distance, in which a vehicle has to drive behind another vehicle. Through several scenes, maneuvers of each traffic participant are composed. The constraint representations of all predicates are collected and used during the reachability analysis to constrain the reachable sets.

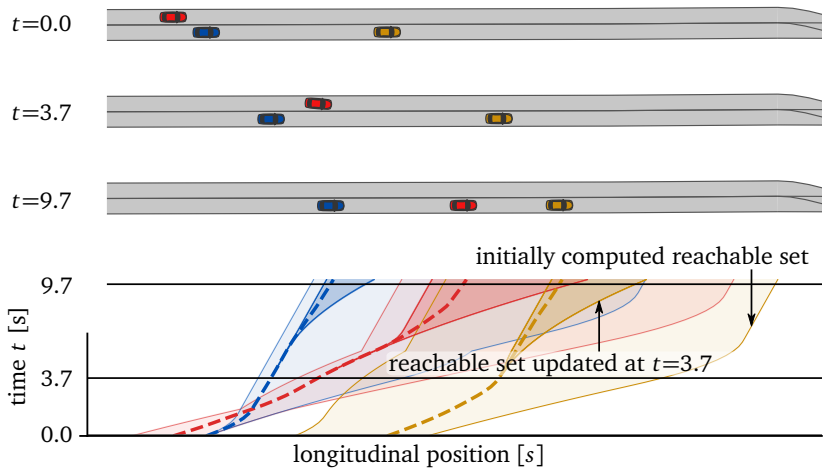


Figure 2.4: Approach for sampling concrete scenarios using reachability analysis to continuously update sampling constraints. For illustrational purposes, the reachable sets are only shown for the initial state and for an update after $t = 3.7$. The dotted lines depict the sampled trajectories of traffic participants. (Modified from [5].)

Chapter 3

Generating Non-Reactive Safety-Critical Test Scenarios Using Optimization

In this chapter, we present the method for creating test scenarios that addresses the problem described in Sec. 2.3.1. First, the optimization-based method to increase the criticality using the drivable area is introduced in Sec. 3.1. To apply it on a larger scale, we develop a pipeline that creates scenarios based on open-source map data and a traffic simulator in Sec. 3.2. In Sec. 3.3, we present a method for efficiently computing the drivable area.

3.1 Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms

Creating safety-critical test scenarios for unkown SUT as introduced in Problem 1 is challenging because the behavior model of the SUT is not available when creating the scenarios. However, most existing criticality metrics can only be evaluated with respect to a concrete state of the trajectory of an SUT. Furthermore, optimizing traffic scenarios with respect to a criticality metric is a highly nonlinear problem where optimization algorithms might get stuck in local minima.

To address these issues, we develop an approach to optimize the trajectories of other traffic participants in initially uncritical concrete traffic scenarios using evolutionary algorithms and a criticality metric based on the drivable area. This metric abstracts arbitrary SUTs by representing them using the solution space of all collision-free trajectories. Hence, we can optimize the criticality without depending on a concrete SUT. A major advantage of using the drivable area is that we can demand a non-empty drivable area to aim at finding scenarios for which a collision-free motion plan for the ego vehicle exists. To generate realistic traffic scenarios, it is necessary to account for additional constraints when adjusting the trajectories of other traffic participants, such as preventing collisions between other traffic participants. Our proposed method ensures valid scenarios by correcting invalid scenarios through solving locally convexified problems. Furthermore, we are only interested in traffic scenarios, where the trajectories of other traffic participants affect the drivable area of the SUT. To improve the computation time, we develop an algorithm that prunes irrelevant parameter intervals for which the drivable area is not affected.

We evaluate our approach in urban and highway scenarios and demonstrate that it can handle complex scenarios with many traffic participants. Additionally, we compare the performance of two evolutionary algorithms, which indicates a superior performance of particle swarm optimization compared to differential evolution.

Contributions M. A. initiated the idea of using the drivable area to optimize the criticality of traffic scenarios. M. K. developed the algorithms for correcting invalid scenarios during

the optimization and for pruning the parameter intervals of the scenario parameters. M. K. evaluated the approach and wrote the article, M. A. provided feedback and improved the manuscript.

Conference Article In this thesis, the accepted version submitted by the author is reprinted. The edited version is available under https://dx.doi.org/10.1109/IVS.2019.8814230.

IEEE Requesting permission to reuse content from an IEEE publication

Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms

Conference Proceedings: 2019 IEEE Intelligent Vehicles Symposium (IV)

Author: Moritz Klischat

Publisher: IEEE

Date: June 2019

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms

Moritz Klischat and Matthias Althoff

Abstract— **Virtual testing of automated vehicles using simulations is essential during their development. When it comes to the testing of motion planning algorithms, one is mainly interested in challenging, critical scenarios for which it is hard to find a feasible solution. However, these situations are rare under usual traffic conditions, demanding an automatic generation of critical test scenarios. We present an approach that automatically generates critical scenarios based on a minimization of the solution space of the vehicle under test. By formulating a scenario parametrization and automatic determination of relevant parameter intervals, we are able to optimize the criticality of complex scenarios. We use evolutionary algorithms to tackle the resulting highly nonlinear optimization problem. Compared to our previous approach, we are now able to handle complex situations, in particular those involving intersections. Finally, we demonstrate our approach by generating critical scenarios from initially uncritical scenarios.**

I. Introduction

When automated vehicles are deployed in the real world, they are subjected to an unforeseeable number of situations. This requires extensive testing during their development to ensure safety, especially with regards to motion planning. To demonstrate that automated vehicles have a better performance than humans with a 95% confidence level, they need to be driven for 440 million km [1]. Because this cannot be achieved through real-world testing alone, virtual tests are a common practice, making it possible to test many aspects faster than in real-time, see e.g., [2]–[4]. Even virtual tests, however, can be very time-consuming since dangerous or interesting situations are rare events. This motivates automatic generation of critical test cases for automated vehicles—in this work, we focus on test cases for motion planning. These tests cases expose the vehicle under test to short traffic scenarios for which a feasible motion needs to be found.

A. Related work

One straightforward approach for increasing the efficiency of virtual testing is the extraction and classification of relevant scenarios from large databases of recorded traffic data as demonstrated in, e.g., [5], [6]. In [7], scenarios are grouped by unsupervised learning to find situations where small deviations of the environment lead to changes in behavioral modes, e.g., when the vehicle under test is forced to take a different path. Despite the fact that collecting data at this scale is challenging, one is restricted to observed situations, which are typically not critical most of the time.

All authors are with the Technische Universität München, Fakultät für Informatik, Lehrstuhl für Robotik, Künstliche Intelligenz und Echtzeitsysteme, Boltzmannstraße 3, 85748, Garching, Germany. {moritz.klischat, althoff}@in.tum.de

For generating new scenarios based on existing data, test cases are created automatically based on observed cause-effect relations, which are checked subsequently during test execution in [8]. Combination and mutation of recorded data is used in [9] to create new test cases. In [10] the authors propose using learned behavior from recorded or simulated traffic data to generate traffic scenarios with neural networks, which are subsequently searched for accidents. However, an accident does not necessarily imply that a situation is critical; it might have been easily avoided. In return, a critical situation might be resolved by a good driver and not be classified as critical.

Another practice for more systematic testing is to define scenarios on an abstract level and by deducting test cases through the variation of parameters, see e.g., [11], [12]. However, without considering the criticality of a scenario, the number of generated test cases quickly becomes uneconomical. Criticality is explicitly considered in [13]–[15], where automated vehicles are tested by optimizing scenarios towards a short Time-To-Collision or using related cost functions. Similarly, in [16], systems are forced into faulty behavior with respect to previously-defined specifications. In [17] and [18], test case generation for automated vehicles is realized using S-TaLiRo. In [19], an approach using S-TaLiRo is applied to motion planners based on machine learning, including simulated camera processing using deep neural networks.

Our previous work [20] is the first approach that generates critical scenarios with a small solution space for the vehicle under test, which we refer to as the ego vehicle from now on. In that work, the drivable area is used as a measure for criticality. The drivable area denotes the solution space in which the ego vehicle can operate safely without colliding. This measure enables the quantification of criticality in many more driving situations than the Time-To-Collision. However, [20] only allows the generation of simple scenarios on straight, non-intersecting roads and only realized translation of other traffic participants.

B. Contributions

We present an approach to generate critical scenarios for testing motion planners in complex traffic situations. Based on the optimization of the drivable area of the ego vehicle, our method provides the following improvements compared to [20]:

1) A new parametrization of scenarios is presented for realizing more complex and diverse scenarios.

2) By using evolutionary algorithms (EA), we find better local minima over a larger range of scenarios. This is especially advantageous compared to the local linearization used in [20] despite the highly nonlinear optimization problem at hand.
3) To improve the optimization process and to consider only relevant driving situations, a method to prune the parameter space of a scenario is presented. As a result, only parameter regions where traffic participants can possibly interfere with the ego vehicle are considered.
4) By formulating a repair algorithm based on a linear program, obtained scenarios do not contain collisions among other traffic participants.

Our approach considers all feasible trajectories of the ego vehicle unlike previous approaches that generate scenarios based on a closed-loop simulation with the motion planner of the ego vehicle [13]–[15]. Thus, our approach yields test cases whose criticality is not depending on the performance of the motion planner. This enables a more objective comparison of multiple motion planners and the creation of a database with generic critical scenarios. Since we only generate scenarios with a non-empty drivable area, we also ensure that a collision-free trajectory exists, compared to approaches that focus on finding accident scenarios.

II. Problem Statement

The subsequent formulation of the underlying optimization problem is similar to [20], except that we consider collisions among other traffic participants. For an illustration of the introduced variables, we refer to Fig. 2a.

A. Traffic Participants

We define the list $\mathcal{V}$ of traffic participants $V^{(i)}$. In the remainder of this work, the superscript $\square^{(i)}$ refers to the i-th traffic participant. For each traffic participant, we assume a parametrized trajectory $x^{(i)}(t;p) \in \mathbb{R}^n$, where the parameter vector $p \in \mathcal{P}$ and parameter space $\mathcal{P} \in \mathbb{R}^{1\times q}$ are presented in detail in Sec. III-A. We introduce the operator $\mathrm{occ}(x)$ returning the occupied space of a vehicle with state x. The occupancy set of a traffic participant is denoted by $\mathcal{O}^{(i)}(t,p) = \mathrm{occ}(x^{(i)}(t;p))$. To each traffic participant we assign a lane $\mathcal{L}^{(i)} \in \mathbb{R}^2$ of the road network.

B. Motion Planning Problem of the Ego Vehicle

Let us introduce the motion planning problem for the ego vehicle as follows: By $f(x(t),u(t))$ we denote the right-hand side of the state-space model of the ego vehicle so that

$$\dot{x}(t) = f(x(t),u(t)),$$

where $x \in \mathbb{R}^n$ is the state vector and $u \in \mathbb{R}^m$ is the input vector. We further require the initial state $x_0 = x(t_0)$, the initial time t_0, and the time horizon t_f. The possibly time-varying, allowed space on the road surface is denoted by $\mathcal{W}_{\mathrm{lanes}}(t) \subset \mathbb{R}^2$. The occupancy of the ego vehicle has to lie within the allowed space, while avoiding other traffic participants $\forall t \in [t_0,t_f] : \mathrm{occ}(x(t)) \subseteq \mathcal{W}_{\mathrm{lanes}}(t)\backslash\mathcal{O}(t,p)$. We also require constraints $g(x(t),u(t),t) \leq 0$, such as speed limits or other traffic rules [21].

After introducing an input trajectory as $u(\cdot)$ (in contrast to a value $u(t)$ at time t) and the cost function of the obtained solution $J(x(t),u(t),t_0)$, we can finally formulate the motion planning problem as finding

$$u^*(\cdot) = \arg\min_{u(\cdot)} J(x(t),u(t),t_0)$$

subject to

$$\begin{aligned} &\dot{x}(t) = f(x(t),u(t)), \quad \mathrm{occ}(x(t)) \subseteq \mathcal{W}_{\mathrm{lanes}}(t)\backslash\mathcal{O}(t,p), \\ &g(x(t),u(t),t) \leq 0, \quad x(t_0) = x_0\,. \end{aligned} \tag{1}$$

Finally, we denote a scenario by the tuple

$$S(p) = (x_0, \mathcal{O}(\cdot,p), \mathcal{W}_{\mathrm{lanes}}(\cdot)).$$

C. Drivable Area

To consider not only the optimal solution of the motion planning problem, but the space of all solutions, we require the set of reachable states [22]. In particular, we use a so-called *anticipated reachable set*, which excludes states that will inevitably result in an accident [23] in the time interval $t \in [t_0,t_f]$. We denote a feasible trajectory as $\chi(t;x_0,u(\cdot))$, which meets all constraints in (1). After introducing the set of input trajectories $\mathcal{U}$, we define the anticipated reachable set as

$$\begin{aligned} \mathcal{R}(t;x_0,\mathcal{O}(\cdot,p),\mathcal{W}_{\mathrm{lanes}}(\cdot),t_f) = \Big\{ &\chi(t;x_0,u(\cdot)) \,\Big|\, \exists u(\cdot) \in \mathcal{U}, \\ &\forall \tau \in [t_0,t_f] : \mathrm{occ}\big(\chi(\tau;x_0,u(\cdot))\big) \subseteq \mathcal{W}_{\mathrm{lanes}}(\tau)\backslash\mathcal{O}(\tau,p) \Big\}. \end{aligned}$$

By applying the projection operator for projecting to the position domain in Euclidean space $\underset{xy}{\mathrm{proj}}(x) : \mathbb{R}^n \to \mathbb{R}^2$, the drivable area becomes

$$\begin{aligned} &\mathcal{D}(t;x_0,\mathcal{O}(\cdot,p),\mathcal{W}_{\mathrm{lanes}}(\cdot)) = \\ &\bigcup_{x\in\mathcal{R}(t;x_0,\mathcal{O}(\cdot,p),\mathcal{W}_{\mathrm{lanes}}(\cdot),t_f)} \underset{xy}{\mathrm{proj}}(x). \end{aligned} \tag{2}$$

To quantify the solution space over time, we introduce the function $\mathrm{area}(\mathcal{X}) : \mathbb{R}^2 \to \mathbb{R}^+$, returning the area of a set. We write

$$A(p,t) \coloneqq \mathrm{area}\big(\mathcal{D}(t;x_0,\mathcal{O}(\cdot,p),\mathcal{W}_{\mathrm{lanes}}(\cdot))\big)$$

to obtain the area profile of the drivable area over time.

D. Optimization Problem

The goal of this work is to create a scenario $S(p)$ with a desired area profile $A_{ref}(t)$ by optimizing

$$\arg\min_p \kappa(S(p)), \quad \kappa(S(p)) = \int_0^{t_f} \big(A(p,t) - A_{ref}(t)\big)^2 \tag{3}$$

$$\text{subject to} \qquad \forall t, \forall i, \forall j : \quad \mathcal{O}^{(i)}(t,p) \cap \mathcal{O}^{(j)}(t,p) = \emptyset. \tag{4}$$

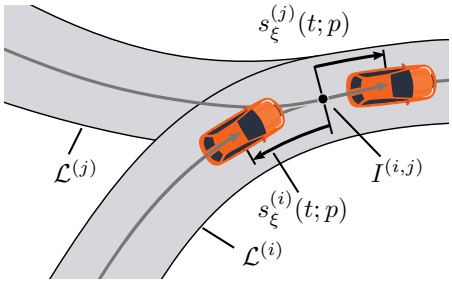


Fig. 1: Longitudinal coordinates $s_\xi(t_k;p)$ formulated relative to the intersection point $I^{(i,j)}$ of lanes $\mathcal{L}^{(i)}$ and $\mathcal{L}^{(j)}$.

The constraint in (4) ensures that no traffic participants collide with each other. We refer to the set of parameters, for which this constraint holds, as the *feasible set*. In this work, we use for $A_{ref}(t)$ the drivable area computed without any traffic participants and the scalar $\gamma \in]0,1[$, which quantifies the reduction of the drivable area: $A_{ref}(t) = \gamma \cdot \text{area}(\mathcal{D}(t; x_0, p, \emptyset, \mathcal{W}_{\text{lanes}}(\cdot)))$.

III. Parametrization of the Optimization Problem

For the trajectories $x^{(i)}(t;p)$ of all traffic participants $\mathcal{V}$, we require a parametrization that can be applied to complex road networks and enables efficient handling of collision constraints in (4), yet is lightweight enough for being solved in reasonable time. To this end, we introduce the curvilinear coordinate system C, in which a state is defined as $x_C = [s_\xi, \dot{s}_\xi, s_\eta, \dot{s}_\eta]^T$, where ξ denotes longitudinal and η denotes lateral coordinates with respect to the center line of a lane $\mathcal{L}$. The operator $\underset{\xi}{\text{proj}}(x) : \mathbb{R}^2 \to \mathbb{R}$ projects the Euclidean space to the longitudinal position domain. Furthermore, we use discretized time $t_k = \Delta t \cdot k$ with time steps $k \in \mathbb{N}$ and step size $\Delta t \in \mathbb{R}^+$.

A. Parametrization of Traffic Participants

For each vehicle, we assume an initial trajectory to be given and independent dynamics in lateral and longitudinal direction. We parametrize the longitudinal position trajectory as

$$s_\xi^{(i)}(t_k; p^{(i)}) = \hat{s}_\xi(t_k) + p_s^{(i)} + t_k p_v^{(i)} + \frac{1}{2} t_k^2 p_a^{(i)} \qquad (5)$$

with the initial longitudinal trajectory $\hat{s}_\xi(t_k) \in \mathbb{R}$ and the parameter vector

$$p = \left[p_s, p_v, p_a\right]$$

for translations $p_s \in \mathbb{R}^{1\times\nu}$, initial velocity variations $p_v \in \mathbb{R}^{1\times\nu}$, and acceleration variations $p_a \in \mathbb{R}^{1\times\nu}$. The parameter p is bounded by the multidimensional interval set

$$\mathcal{P} = \{[p_s, p_v, p_a] \mid p_s \in \mathcal{S}, p_v \in \mathcal{B}, p_a \in \mathcal{A}\}$$

with

$$\mathcal{S} = \left[\underline{p_s}, \overline{p_s}\right], \mathcal{B} = \left[\underline{p_v}, \overline{p_v}\right], \mathcal{A} = \left[\underline{p_a}, \overline{p_a}\right].$$

By $\overline{\square}$ we denote the supremum and by $\underline{\square}$ the infimum of interval sets.

B. Collision Constraints

In order to solve the collision constraints in (4) efficiently, we approximate them by a formulation as linear inequality constraints of the form $d(t_k,p) \leq \Delta r$ with $d(t_k,p), \Delta r \in \mathbb{R}^\rho$, which represent a convex feasible set. We introduce the solution candidate $\tilde{p} \in \mathcal{P}$, which is repaired using an Euclidean projection of $\tilde{p}$ onto the convex feasible set. This projection is trivial and can be solved by a quadratic program [24]. The repair mechanism is used during optimization as described in Sec. IV.

The elements of $d(t_k,p)$ are obtained by pair-wise formulations of collision constraints between traffic participants. Due to the scenario parametrization in (5), not all traffic participants can collide, e.g., if the lanes of two vehicles never intersect. Therefore, we first identify all pairs of traffic participants $V^{(i)}, V^{(j)}, \forall i \neq j$, which can possibly collide by checking for intersection of their lanes, $\mathcal{L}^{(i)} \cap \mathcal{L}^{(j)}$.

Constraints are composed of the distances between vehicles along lanes. While obtaining distances between traffic participants in the same lane is trivial, defining longitudinal distances of merging or intersecting lanes is not obvious. To this end, we define the intersection point of lanes of traffic participants $V^{(i)}$ and $V^{(j)}$ as $I^{(i,j)} \in \mathbb{R}^2$ as shown in Fig. 1, which serves as the origin of the curvilinear coordinate systems. Longitudinal distances in these coordinate systems are then classically obtained as $|s_\xi^{(i)}(t_k;p) - s_\xi^{(i)}(t_k;p)|$.

Depending on $\tilde{p}$, two configurations of two traffic participants are possible: either $V^{(i)}$ is passing before (case ①) or behind (case ②) $V^{(j)}$. In order to preserve the configuration, which results from the parameter $\tilde{p}$ at hand, we distinguish the cases by

$$d^{(i,j)}(t_k,p) = \begin{cases} s_\xi^{(j)}(t_k;p) - s_\xi^{(i)}(t_k;p) & \text{for case ①} \\ s_\xi^{(i)}(t_k;p) - s_\xi^{(j)}(t_k;p) & \text{for case ②} \end{cases}.$$

Finally, we write the collision constraint for each t_k as

$$d^{(i,j)}(t_k,p) \leq (r^{(i)} + r^{(j)}), \qquad (6)$$

where r is the radius of the circle inscribing the shape of a traffic participant, including a safety margin.

C. Pruning of the Parameter Space

Solving our optimization problem is complicated since a traffic participant can only reduce the drivable area $\mathcal{D}$ if it intersects it at some point in time. However, large regions of possibly occupied positions $\{\mathcal{O}(\cdot,p) \mid p \in \mathcal{P}\}$ of traffic participants might never intersect the drivable area $\mathcal{D}$ like vehicle $V^{(1)}$ in Fig. 2a. To quickly converge to solutions reducing the drivable area, we automatically want to remove regions from the translational parameter space $\mathcal{S}$ which have no influence on the drivable area, as shown in Fig. 2b for $V^{(1)}$. In contrast, there exist parameters within $\mathcal{P}$ that can drastically reduce the drivable area; this typically depends largely on the traffic participant. Thus, we first identify the best traffic participants $\mathcal{V}^B$ which have a large influence on

the reduction of the drivable area and separate them from the worse remaining traffic participants $\mathcal{V}^W$:

$$\mathcal{V} = \mathcal{V}^B \cup \mathcal{V}^W \,. \tag{7}$$

From now on, we denote by superscripts $\square^B$ and $\square^W$ the relation to traffic participants of the corresponding sets.

For identifying traffic participants with a large influence on the drivable area, i.e., the cost function κ, we define the criterion

$$\lambda(V^{(i)}, \tilde{p}) = \frac{\kappa(S(x_0, \mathcal{O}(\cdot, \tilde{p}), \mathcal{W}_{\text{lanes}}(\cdot)))}{\kappa(S(x_0, \hat{\mathcal{O}}(\cdot, \tilde{p}), \mathcal{W}_{\text{lanes}}(\cdot))},$$
$$\hat{\mathcal{O}}^{(i)}(\cdot, \tilde{p}) = \mathcal{O}(\cdot, \tilde{p}) \setminus \mathcal{O}^{(i)}(\cdot, \tilde{p}),$$

which expresses the ratio of costs for the scenario with and without the i-th traffic participant for the parameter $\tilde{p}$. As shown in Algorithm 1, $\mathcal{V}^B$ is initially empty and new members $\mathcal{V}_{b,new}$ are selected by

$$V_{b,new} = \underset{V^{(i)} \in \mathcal{V}^W}{\arg\min}\, \lambda(V^{(i)}, \tilde{p}) \tag{8}$$

to obtain the traffic participant with the largest impact on the cost function.

After adding $V_{b,new}$ to $\mathcal{V}^B$, we want to identify the intervals $\mathcal{S}^{W,(i)} = [\underline{p_s}^{(i)}, \overline{p_s}^{(i)}]$ for the remaining $V^{(i)} \in \mathcal{V}^W$, which contain all parameters that can intersect with $\mathcal{D}(\cdot, x_0, \mathcal{O}(\cdot, [\tilde{p}^B, p^W]), \mathcal{W}_{\text{lanes}}(\cdot))$. By computing $\mathcal{D}^B(\cdot, x_0, \mathcal{O}^B(\cdot, \tilde{p}^B), \mathcal{W}_{\text{lanes}}(\cdot))$ considering the occupancies $\mathcal{O}^B(\cdot, \tilde{p}^B)$ only, we obtain a superset of the drivable area

$$\forall p^W \in \mathcal{P}^W : \mathcal{D}(\cdot, x_0, \mathcal{O}(\cdot, [\tilde{p}^B, p^W]), \mathcal{W}_{\text{lanes}}(\cdot)) \subseteq \mathcal{D}^B(\cdot, x_0, \mathcal{O}^B(\cdot, \tilde{p}^B), \mathcal{W}_{\text{lanes}}(\cdot)), \tag{9}$$

where $[\tilde{p}^B, p^W]$ denotes the combined parameter vector of $\tilde{p}^B$ and p^W. From now on, the shortened notation $\mathcal{D}^B(\cdot, \tilde{p}^B) = \mathcal{D}^B(\cdot, x_0, \mathcal{O}^B(\cdot, \tilde{p}^B), \mathcal{W}_{\text{lanes}}(\cdot))$ is used. By restricting $\mathcal{S}^W$ to the interval which results in an intersection with $\mathcal{D}^B(\cdot, \tilde{p}^B)$, we consequently guarantee that $\mathcal{S}^W$ contains the desired parameters only. This is formalized as

$$\underline{p_s}^{(i)} = \min \{p_s^{(i)} \mid \mathcal{O}^{(i)}(\cdot, [p_s^{(i)}, p_v^{(i)}, p_a^{(i)}]) \cap \mathcal{D}^B(\cdot, \tilde{p}^B) \neq \emptyset, \forall p_v^{(i)} \in \mathcal{B}^{(i)}, \forall p_a^{(i)} \in \mathcal{A}^{(i)}\},$$

$$\overline{p_s}^{(i)} = \max \{p_s^{(i)} \mid \mathcal{O}^{(i)}(\cdot, [p_s^{(i)}, p_v^{(i)}, p_a^{(i)}]) \cap \mathcal{D}^B(\cdot, \tilde{p}^B) \neq \emptyset, \forall p_v^{(i)} \in \mathcal{B}^{(i)}, \forall p_a^{(i)} \in \mathcal{A}^{(i)}\}.$$

Since this is a challenging problem, we formulate an over-approximative representation using an over-approximation of $\mathcal{D}^B(\cdot, \tilde{p}^B)$ in curvilinear coordinates, thus guaranteeing that the entire drivable area is considered when computing $\mathcal{S}^W$. For that purpose, we intersect the drivable area $\mathcal{D}^B(\cdot, \tilde{p}^B)$ with the corresponding lane $\mathcal{L}^{(i)}$ and project it to the longitudinal position domain to obtain the over-approximative interval of $\mathcal{D}^B(\cdot, \tilde{p}^B)$ in curvilinear coordinates (see Fig. 2b):

$$d_\xi^{(i)}(t_k) = \left[\inf \left\{ \underset{\xi}{\text{proj}} \left(\mathcal{D}^B(t_k, \tilde{p}^B) \cap \mathcal{L}^{(i)} \right) \right\}, \sup \left\{ \underset{\xi}{\text{proj}} \left(\mathcal{D}^B(t_k, \tilde{p}^B) \cap \mathcal{L}^{(i)} \right) \right\} \right].$$

Afterwards, we obtain the bounds of $\mathcal{S}^{W,(i)}$ for the time interval $[t_0, t_f]$ as

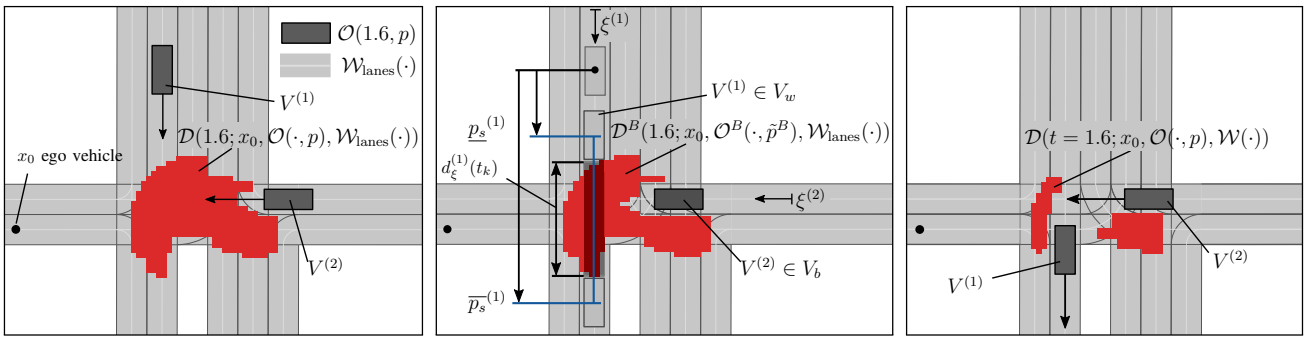
$$\underline{p_s}^{(i)} = \min_{t_k \in [t_0, t_f]} \Big(\underline{d_\xi}^{(i)}(t_k) - \hat{s}_\xi(t_k) \underbrace{- (t_k - t_0)\overline{p_v} - \frac{1}{2}(t_k - t_0)^2 \overline{p_a}}_{\text{minimizes } \underline{p_s}^{(i)}} \Big), \tag{10}$$

$$\overline{p_s}^{(i)} = \max_{t_k \in [t_0, t_f]} \Big(\overline{d_\xi}^{(i)}(t_k) - \hat{s}_\xi(t_k) \underbrace{- (t_k - t_0)\underline{p_v} - \frac{1}{2}(t_k - t_0)^2 \underline{p_a}}_{\text{maximizes } \overline{p_s}^{(i)}} \Big) \tag{11}$$

which directly follows from solving (5) for $p_s^{(i)}$.

IV. Optimization Routine

For solving the optimization problem in (3), we utilize evolutionary algorithms. These algorithms are especially suited for global optimization problems for which no analytic gradient can be formulated [6], as is the case for the cost function $\kappa(S(p))$. One can incorporate any EA in our approach as shown in Algorithm 1; we exemplarily



(a) Initial scenario. (b) Pruning parameter region $\mathcal{S} = [\underline{p_s}, \overline{p_s}]$. (c) Solution of optimization: critical scenario.

Fig. 2: Generating a critical scenario. Depicted are drivable area D and traffic participants $V^{(i)}$ at $t = 1.6\,\text{s}$.

use differential evolution (DE) [25] and particle swarm optimization (PSO) [26] and compare them.

During optimization, we perform intermediate tightening of parameter bounds $\mathcal{P}$ according to Sec. III-C. For the optimization, we define the population Q as the set consisting of all n_p solution candidates p_e, $e \in \{0, ..., n_p\}$ for the EA.

Initially, all traffic participants are assigned to $\mathcal{V}^W$, and initial bounds are computed according to Sec. III-C. After conducting n_b iterations with the respective solver, all solution candidates $p_e \in Q$ which violate the constraints in (6) are repaired by computing the closest projection to the feasible set as described in Sec. III-B. Afterwards, intermediate updating of parameter bounds $\mathcal{S}$ is performed by first selecting the most relevant member $V_{b,new} \in \mathcal{V}^W$ with respect to $\kappa(S(p))$ for adding it to $\mathcal{V}^B$. With the updated sets $\mathcal{V}^B$ and $\mathcal{V}^W$, the parameter bounds $\mathcal{S}^W$ are tightened in the subsequent iteration step l using (10) and (11). Next, elements of solution candidates $p_e \in Q$, which violate $p_e \in \mathcal{P}$, are resampled within the newly computed bounds. This routine is repeated until either all traffic participants are assigned to $\mathcal{V}^B$ or the optimization algorithm converged. For a step-by-step example that illutstrates the iterative computation of bounds, we refer to Sec. V-A.

Algorithm 1 IterativeBoundingOptimization

Require: scenario $S(p)$, initial solution $\tilde{p}$, traffic participants $\mathcal{V}$, number of traffic participants n_o, initial bounds $\mathcal{P}$
Ensure: Critical Scenario S

$\mathcal{V}^W \leftarrow \mathcal{V}$
$\mathcal{V}^B \leftarrow \emptyset$
$Q \leftarrow$ INITPOPULATION
$l \leftarrow 0$
$converged \leftarrow false$
while $l < n_o$ and $\neg converged$ **do**
  $\mathcal{P}^W \leftarrow$ TIGHTENPARAMETERBOUNDS$(\mathcal{V}^W, \mathcal{V}^B, \tilde{p})$ ▷ see (10) and (11)
  $Q, converged \leftarrow$ EA$(\mathcal{P}, Q)$
  $Q \leftarrow$ REPAIRINFEASIBLE(Q) ▷ see Sec. III-B
  $\tilde{p} \leftarrow \arg\min_{p_e \in Q} \kappa(S(p_e))$
  $V_{b,new} \leftarrow$ SELECTRELEVANT$(\mathcal{V}^W, \mathcal{V}^B, \tilde{p})$ ▷ see (8)
  $\mathcal{V}^B \leftarrow \mathcal{V}^B \cup V_{b,new}$
  $\mathcal{V}^W \leftarrow \mathcal{V}^W \backslash V_{b,new}$
  $l \leftarrow l + 1$
end while
return $S(\tilde{p})$

V. RESULTS

The proposed approach is demonstrated by two initial scenarios from the CommonRoad benchmark collection [27]. We also compare results from the two evolutionary algorithms DE and PSO. Due to the lack of comparable algorithms, no comparison to existing work is possible. For the computation of the drivable area, we use the method presented in [23]. Used parameters are listed in Table I. We used different settings for the solvers in both scenarios due to their different complexities. The settings are described in the respective paragraphs. Computation times were measured on a machine with an Intel i7-8650U 1.90 GHz processor.

TABLE I: Scenario Parameters

max. acceleration ego vehicle $\lvert a_{max} \rvert$	$5.0\,\mathrm{m/s^2}$
time step size Δt	$0.1\,\mathrm{s}$
time horizon t_f	$3.0\,\mathrm{s}$
initial velocity variation $\mathcal{B}$	$[-3, 3]\,\mathrm{m\,s^{-1}}$
acceleration variation $\mathcal{A}$	$[-5, 5]\,\mathrm{m/s^2}$

A. Scenario I: Intersection

The first scenario is a hand-crafted, unregulated urban intersection that can be found in the CommonRoad benchmark collection[1] under ID DEU_Ffb-1_3_T-1. The ego vehicle has an initial velocity $v_0 = 7.1\,\mathrm{m\,s^{-1}}$ and is surrounded by 3 other vehicles. This results in 9 parameters, for which a population of 90 individuals is used during the optimization. We conduct 45 iterations with both solvers, while performing parameter bounding every $n_b = 15$ steps. The computation time for this scenario is 22.09 minutes.

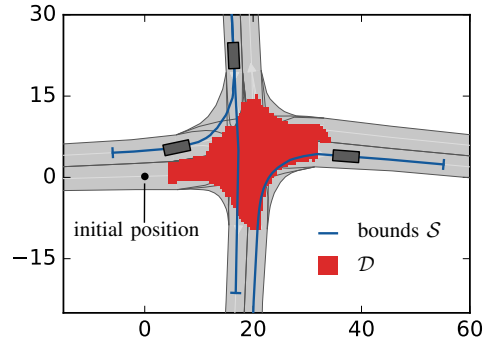
The initial configuration at $t = 2.5\,\mathrm{s}$ is depicted in Fig. 3a. The other vehicles almost do not restrict the drivable area $\mathcal{D}$ and thus the situation is uncritical. The final result of the particle swarm algorithm is shown in Fig. 3. To illustrate the optimization routine, we show three intermediate solutions at different iterations of the optimization. In the initial configuration, the depicted position bound $\mathcal{S}$ for all vehicles are large due to the large drivable area. However, after two adaptation iterations, the bounds could be decreased due to the smaller drivable area. After the final iteration $k = 3$, the drivable area has decreased even further.

The optimized scenario shows a vehicle coming from the left and ignoring the right of way of the ego vehicle while breaking with $a = -2\,\mathrm{m\,s^{-1}}$. The ego vehicle either needs to perform an emergency braking maneuver or evade to the right. Even though another traffic participant would be blamed for the potential collision in this case, one is still interested in protecting the passengers of the ego vehicle through a safe maneuver of the motion planner. When comparing the convergence of solvers in Fig. 4, DE and PSO perform almost equally in the beginning; however, after 12 iterations, DE almost does not improve anymore, while the PSO still improves substantially.

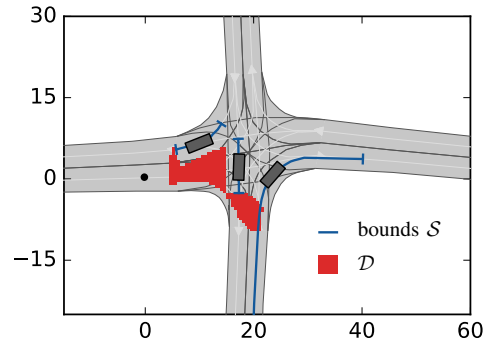
B. Scenario II: Highway scenario

The second scenario is a highway scenario taken from the NGSIM US 101 dataset[2]. This scenario can be found in the CommonRoad benchmark collection under ID USA_US101-14_1_T-1. The ego vehicle has an initial velocity of $14.85\,\mathrm{m\,s^{-1}}$ and is surrounded by similarly paced vehicles in the initial configuration. In order to simplify the scenario,

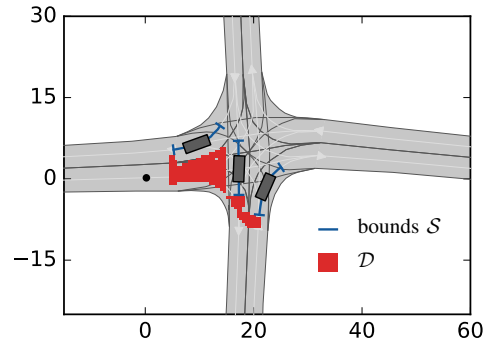
[1] https://commonroad.in.tum.de
[2] http://www.fhwa.dot.gov/publications/research/operations/07030/



(a) Initial configuration at $t = 2.5\,\mathrm{s}$ with initial bounds $\mathcal{S}$.



(b) Configuration after iteration step $l = 2$ at $t = 2.5\,\mathrm{s}$ with adapted bounds $\mathcal{S}$.



(c) Final optimized configuration at $t = 2.5\,\mathrm{s}$ with final bounds $\mathcal{S}$.

Fig. 3: Scenario I: initial configuration, intermediate result during optimization, and the final configuration at $t = 2.5\,\mathrm{s}$. Positional bounds $\mathcal{S}$ of respective iteration k are depicted in blue.

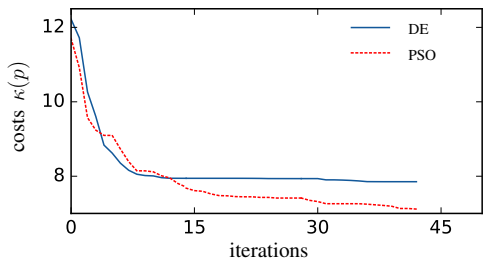


Fig. 4: Comparison of solver convergence for scenario I.

irrelevant vehicles are removed prior to the optimization. As a result, 13 vehicles with a total number of 39 parameters are optimized. This scenario is especially demanding with respect to collision constraints (4) and the high number of parameters in total.

All three algorithms are initialized with a population of 195 individuals and computed for 45 iterations, while parameter bounds are adapted and the repair algorithm is applied every 9 iterations. The computation time is 201.7 minutes. In Fig. 5, the resulting convergence of all solvers

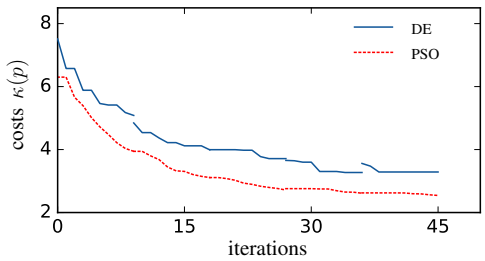


Fig. 5: Comparison of solver convergence for scenario II.

are compared. It shows that PSO has the highest convergence rate and yields the best solution. In comparison, DE exhibits premature convergence. The occasional increase of the cost function can be attributed to the repair algorithm in line 9 of Algorithm 1.

The result of the PSO algorithm is depicted in Fig. 6. While in the beginning the ego vehicle has enough space to maneuver, at $t = 2.6\,\mathrm{s}$ there is little space left due to a lane-changing vehicle and several closely navigating vehicles. For comparison, the initial scenario prior to the optimization, where the vehicle has considerably more space to maneuver, is shown. As this scenario also demonstrates, no vehicles collide, despite the crowded driving situation.

VI. Conclusions

In this work, we present an optimization-based approach to generate critical scenarios for complex traffic situations. Unlike previous works, our approach can handle complex road layouts and dynamics for a high number of involved traffic participants. We ensure that all relevant configurations of a scenario can be reached due to the automatic computation of relevant parameter intervals and evolutionary algorithms. We demonstrate that we can generate critical scenarios for urban scenarios and many involved traffic participants. The obtained scenarios can be used for testing arbitrary motion planners.

Acknowledgements

The authors gratefully acknowledge financial support by the Central Innovation Programme of the German Federal Government under grant ZF4086007BZ8.

References

[1] N. Kalra and S. M. Paddock, “How many miles of driving would it take to demonstrate autonomous vehicle reliability?” RAND Corporation, Santa Monica, CA, Tech. Rep., 2016. [Online]. Available: http://www.rand.org/pubs/research_reports/RR1478.html

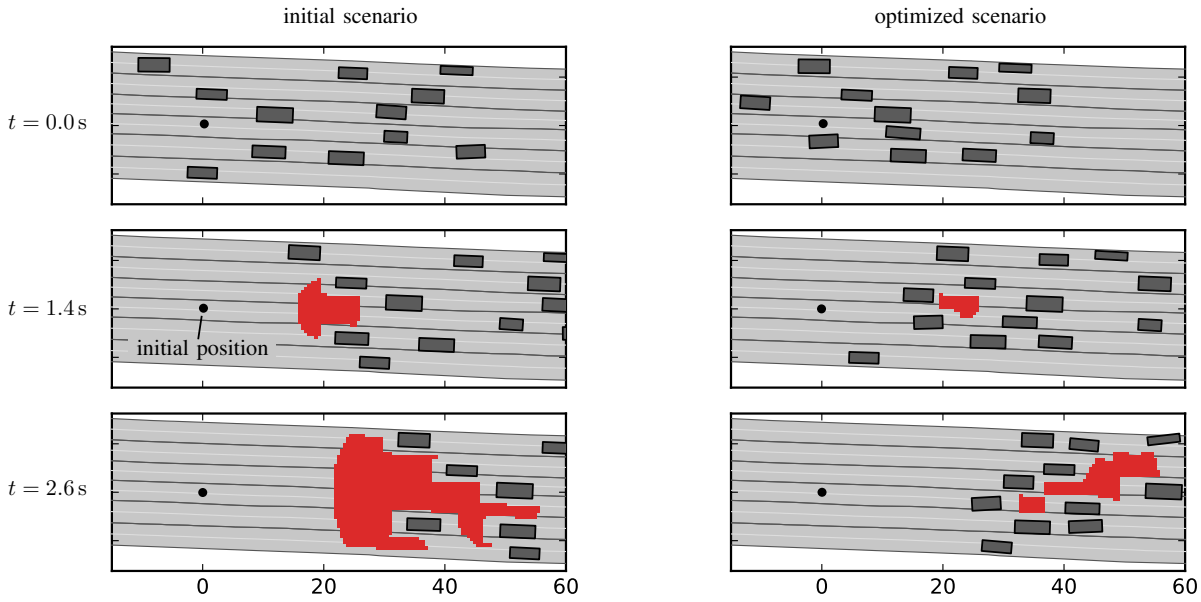


Fig. 6: Scenario II: optimized scenario compared to initial scenario at different times t.

[2] B. Kim, Y. Kashiba, S. Dai, and S. Shiraishi, "Testing autonomous vehicle software in the virtual prototyping environment," *IEEE Embedded Systems Letters*, vol. 9, no. 1, pp. 5–8, 2017.

[3] M. R. Zofka, S. Klemm, F. Kuhnt, T. Schamm, and J. M. Zöllner, "Testing and validating high level components for automated driving: Simulation framework for traffic scenarios," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2016, pp. 144–150.

[4] R. Math, A. Mahr, M. M. Moniri, and C. Müller, "OpenDS: A new open-source driving simulator for research," in *Proc. of Automotive meets Electronics*, 2013.

[5] A. Pütz, A. Zlocki, J. Küfen, J. Bock, and L. Eckstein, "Database approach for the sign-off process of highly automated vehicles," in *25th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration*, 2017.

[6] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. of the Congress on Evolutionary Computation*, vol. 2, 2004, pp. 1980–1987.

[7] G. E. Mullins, P. G. Stankiewicz, and S. K. Gupta, "Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2017, pp. 1443–1450.

[8] C. Wolschke, D. Rombach, P. Liggesmeyer, and T. Kuhn, "Mining test inputs for autonomous vehicles," in *Proc. of Commercial Vehicle Technology*, 2018, pp. 102–113.

[9] V. De Oliveira Neves, M. E. Delamaro, and P. C. Masiero, "An environment to support structural testing of autonomous vehicles," in *European Signal Processing Conference*, 2014, pp. 19–24.

[10] I. R. Jenkins, L. O. Gee, A. Knauss, H. Yin, and J. Schroeder, "Accident scenario generation with recurrent neural networks," in *Proc. of IEEE Conf. on Intelligent Transportation Systems*, 2018, pp. 3340–3345.

[11] F. Schuldt, F. Saust, B. Lichte, M. Maurer, and S. Scholz, "Effiziente systematische Testgenerierung für Fahrerassistenzsysteme in virtuellen Umgebungen," in *Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel*, 2013, pp. 114 – 134.

[12] G. Bagschik, T. Menzel, C. Körner, and M. Maurer, "Wissensbasierte Szenariengenerierung für Betriebsszenarien auf deutschen Autobahnen," in *Workshop Fahrerassistenzsysteme und automatisiertes Fahren. Bd*, vol. 12, 2018.

[13] F. Hauer and B. Holzmüller, "Szenario-Optimierung für die Absicherung von automatisierten und autonomen Fahrsystemen," in *FKFS AutoTest Fachkonferenz*, 2018.

[14] H. Beglerovic, M. Stolz, and M. Horn, "Testing of autonomous vehicles using surrogate models and stochastic optimization," in *Proc. of the IEEE Conf. on Intelligent Transportation Systems*, 2018, pp. 1129–1134.

[15] O. Buehler and J. Wegener, "Evolutionary functional testing of an automated parking system," in *Proc. of the Int. Conf. on Computer, Communication and Control Technologies*, 2003, pp. 26–31.

[16] T. Hempen, S. Biank, W. Huber, and C. Diedrich, "Model based generation of driving scenarios," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, vol. 222, 2018, pp. 153–163.

[17] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing S-TaLiRo as an automatic test generation framework for autonomous vehicles," in *Proc. of the IEEE 19th International Conference on Intelligent Transportation Systems*, 2016, pp. 1470–1475.

[18] H. Abbas, M. E. O'Kelly, A. Rodionova, and R. Mangharam, "A driver's license test for driverless vehicles," *Mechanical Engineering*, vol. 139, no. 12, pp. 13–16, 2017.

[19] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1555–1562.

[20] M. Althoff and S. Lutz, "Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1326–1333.

[21] A. Rizaldi and M. Althoff, "Formalising traffic rules for accountability of autonomous vehicles," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2015, pp. 1658–1665.

[22] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Dissertation, Technische Universität München, 2010, http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20100715-963752-1-4.

[23] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1855–1866, 2018.

[24] J. Dattorro, *Convex optimization & Euclidean distance geometry*. USA: Meboo Publishing, 2011.

[25] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[26] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[27] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.

3.2 Scenario Factory: Creating Safety-Critical Traffic Scenarios for Automated Vehicles

Test scenarios for autonomous vehicles are often obtained through real-world data collection, which is not only costly but also difficult to scale to arbitrary locations. To automate the scenario generation using virtual methods only, we develop a scenario-generation pipeline that uses maps extracted from OpenStreetMap and enriches their level of detail. The map extraction is combined with a traffic simulator to obtain initial scenario candidates. Since scenarios generated by a traffic simulation are typically uncritical and do not provide behaviorally diverse scenarios, we finally apply our approach introduced in Sec. 3.1 for increasing the criticality of the scenarios. The map collection of our dataset is complemented with an anomaly detection algorithm that yields more diverse intersections that only appear rarely in the maps.

For the first time, we evaluate the approach for optimizing the criticality of traffic scenarios using the drivable area on a large dataset of initial scenario candidates. Since the dataset is composed of maps collected from random locations worldwide and due to the outlier detection, the dataset comprises a large variety of road layouts. Our evaluation shows, that our approach is able to reduce the drivable area considerably in the majority of traffic scenarios. Furthermore, we demonstrate that our approach is scalable and can handle a large variety of traffic scenarios. The generated scenarios are contributed to the CommonRoad benchmark suite.

Contributions M. A. initiated the idea for combining automatic map extraction from OpenStreetMap with the simulation and optimization of traffic scenarios. M. K. developed the concept for optimizing the scenarios using the drivable area, which is suitable for a large variety of maps. E. L. developed the approach for automatically extracting and clustering maps, which was implemented by F. H.. M. K. further developed the simulation interface and conducted the final evaluation of the combined approach. M. K. and E. L. wrote the paper together. M. A. led the research project, provided feedback, and helped to improve the manuscript.

Conference Article In this thesis, the accepted version submitted by the author is reprinted. The edited version is available under https://doi.org/10.1109/ITSC45102.2020.9294629.

Scenario Factory: Creating Safety-Critical Traffic Scenarios for Automated Vehicles

Conference Proceedings:
2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)

Author: Moritz Klischat

Publisher: IEEE

Date: 20 September 2020

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

Scenario Factory: Creating Safety-Critical Traffic Scenarios for Automated Vehicles

Moritz Klischat*, Edmond Irani Liu*, Fabian Höltke, and Matthias Althoff

***Abstract*— The safety validation of motion planning algorithms for automated vehicles requires a large amount of data for virtual testing. Currently, this data is often collected through real test drives, which is expensive and inefficient, given that only a minority of traffic scenarios pose challenges to motion planners. We present a workflow for generating a database of challenging and safety-critical test scenarios that is not dependent on recorded data. First, we extract a large variety of road networks across the globe from OpenStreetMap. Subsequently, we generate traffic scenarios for these road networks using the traffic simulator SUMO. In the last step, we increase the criticality of these scenarios using nonlinear optimization. Our generated scenarios are publicly available on the CommonRoad website.**

I. Introduction

Virtual testing is an important tool for validating the safety of automated vehicles, as it exposes potential defects of the algorithms under test. Having a large variety of challenging traffic scenarios is vital for effective and efficient testing of motion planning algorithms. While carrying out simulations using data recorded from test drives provides us with realistic scenarios, the required data collection is often overly expensive and time-consuming [1]. Even though the number of publicly-available datasets has increased over the last few years, e.g., [2]–[5], they usually feature only a small number of maps and require much effort to record.

Our framework generates a database of safety-critical scenarios for scenario-based testing [6] of motion planing algorithms for automated vehicles. It consists of

1) Extracting interesting road intersections worldwide from OpenStreetMap (OSM) [7] by using our Globetrotter tool (see Sec. III).
2) Generating safety-critical test scenarios by first populating the extracted intersections with traffic participants through the traffic simulator SUMO [8].
3) Optimizing the criticality of the obtained scenarios by using a generalizable criticality criterion (see Sec. IV).

A. Related Work

Below, we concisely review related works on approaches towards automatically creating virtual representations of road networks and generating critical test scenarios for automated vehicles.

*The first two authors have contributed equally to this work.

All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.

{moritz.klischat, edmond.irani, fabian.hoeltke, althoff}@tum.de

1) Creating road networks: Generative approaches construct road networks, e.g., based on abstract specifications [9]. Moreover, a suite of road networks with a defined coverage of road curvatures is generated in [10] using satisfiability modulo theories. Road networks that lead to the failure of lane-keeping assistants are generated procedurally in [11] through mutating road networks using genetic algorithms.

Alternatively, road networks can also be created from external sources. In [12]–[16], the authors extract road networks from aerial and satellite images with the help of computer vision techniques. These works are capable of extracting high-level geometric information of road networks; however, lane-level information concerning motion planners of automated vehicles is not reconstructed. Promising works towards the reconstruction of road networks with lane-level detail from aerial images can be seen in [17], [18]. Similarly, while creating road networks for single lanes from OSM data is straightforward [19], creating those with lane-level detail is a more challenging problem [20].

2) Generating critical test scenarios: Creating test scenarios for automated vehicles using traffic simulators is proposed, e.g., in [21], [22]. Realistic scenarios can be obtained by calibrating their simulations through real-world measurements. By using criticality metrics, safety-critical scenarios are filtered [22]; however, these situations occur only rarely, in the main.

To efficiently obtain critical scenarios, importance sampling from large databases of recorded traffic data is proposed [23]. Criticality metrics are combined with the occurrence rates to efficiently sample critical scenarios representative of real-world driving conditions [24]. Other approaches use optimization to create critical scenarios based on these metrics [25], [26]. Similarly, falsification methods can detect scenarios that falsify a motion planner with respect to a given safety specification [27], [28]. Parameter regions for critical scenarios based on constraint satisfaction are computed in [29]. In our previous work, we presented an optimization-based method to increase the criticality of initially uncritical traffic scenarios [30], [31] by decreasing the space of possible solutions for the vehicle under test, called the *drivable area*.

B. Contributions

In contrast to previous work on generating test scenarios through simulation, our approach is particularly efficient since we combine the automatic extraction of road networks from OSM with a traffic simulation followed by an increase of its criticality. By exploiting the large variety of road networks around the world, we create complex, yet realistic

road networks which currently no procedural map generator is capable of producing. In contrast to existing work, our approach does not rely on test drives nor other real-world traffic data, thus it is able to efficiently create many traffic scenarios at low costs. The resulting scenarios are independent of the vehicle under test, due to our criticality metric based on the drivable area.

II. Overview

An overview of our approach is presented in Fig. 1. In the following subsections, we introduce each component.

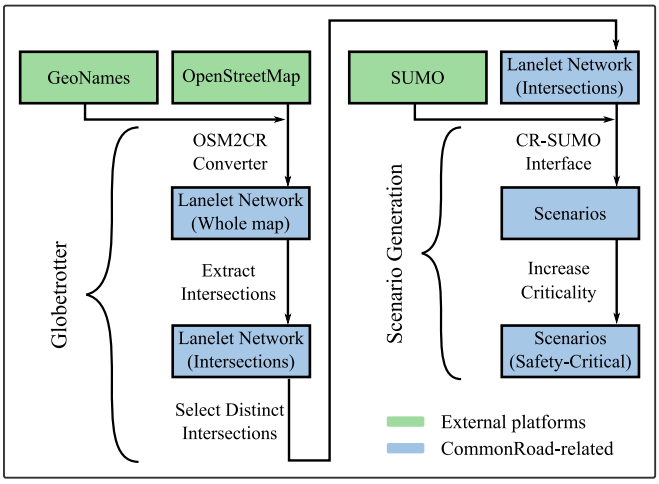


Fig. 1. Our pipeline for generating safety-critical scenarios.

A. Platforms

1) CommonRoad: The CommonRoad (CR) benchmark suite[1] is an open-source framework that provides a collection of traffic scenarios for motion planning algorithms. Scenarios in CommonRoad consist of *road networks*, *static obstacles*, and *dynamic obstacles* that represent all possible types of traffic participants. In this work, we focus on cars, trucks, and bicycles. *Road networks* in CommonRoad are described by lanelets [32] (see Fig. 2). Lanelets are defined by their left and right bounds, which are modeled by polylines. Furthermore, lanelets are connected through successor-predecessor and lateral-adjacency relations and contain additional information such as the speed limit. Additionally, we define *forking points* as the points on the centerlines of lanelets where lanelets split or multiple lanelets merge.

2) OpenStreetMap: OSM is an open-source project that provides geographic data worldwide. The main structure of OSM data is defined by three elements: *nodes*, *ways*, and *relations*. *Nodes* are geographic points defined by their latitude and longitude; *ways* are tuples of *nodes* and represent elements such as roads and boundaries of areas; *relations* are groups of *nodes*, *ways* and other *relations*. Fig. 3a shows a map taken from OpenStreetMap.

[1]https://commonroad.in.tum.de/

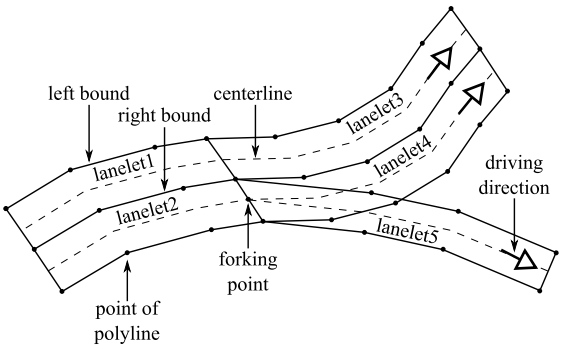


Fig. 2. Lanelet network representation.

3) GeoNames: GeoNames[2] is a free geographic database which covers all countries and contains over 11 million placenames of cities from all over the world. The provided geographical information includes global coordinates, postal codes, population, etc.

4) SUMO: This open-source microscopic traffic-simulation package is designed to handle large road networks. SUMO models individual vehicles and their interactions using models for car-following, lane-changing, and intersection behavior.

B. Converters

Since most of the software platforms mentioned above have individual formats and map representations, we use different converters and interfaces to bridge these platforms.

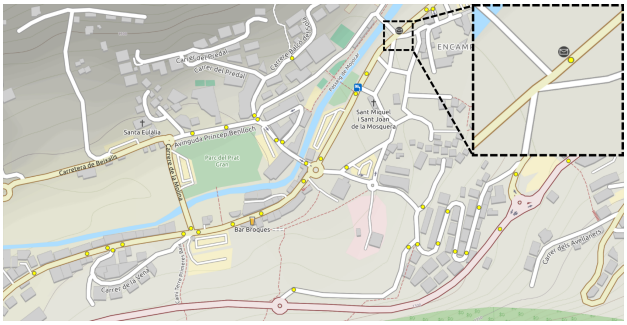
1) OSM2CR converter: It converts OSM maps to CommonRoad lanelet networks. While OSM provides map data for almost any place in the world, their level of detail is not yet suited for automated vehicles: The motion planners of automated vehicles and traffic simulators typically require lane-level information. To resolve this issue, in the first step, the topology of the lanelet network, i.e., the connections at intersections, needs to be estimated. Next, spatial information of individual lanes is deducted accordingly. Fig. 3b shows a converted lanelet network via this converter.

2) CR-SUMO interface: It enables the communication between CommonRoad and SUMO by a) converting the CR road network to SUMO format, b) generating configuration files for the simulation, and c) converting simulated vehicle trajectories to the CR format. We refer the interested reader to [33] for more details regarding this interface.

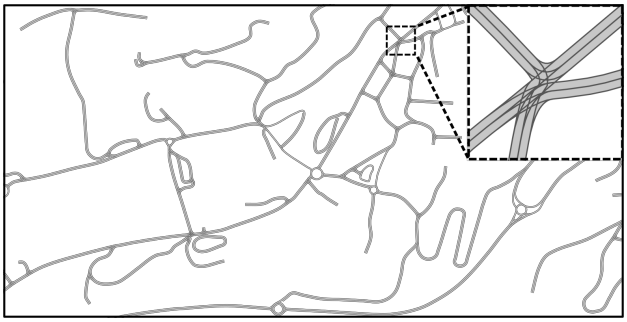
III. Globetrotter

To automatically extract interesting road networks from all over the world, we have developed the Globetrotter tool, which takes the road network data from OSM as its underlying input. As we want to create scenarios on distinct road networks, we mainly focus on extracting intersections. Below, we explain the major steps for extracting the intersections from OSM.

[2]https://www.geonames.org/



(a)



(b)

Fig. 3. (a) Map of Encamp, Andorra taken from OSM. (b) Conversion result into CommonRoad lanelet network via the OSM2CR converter.

A. Retrieving Candidate Regions

Clearly, there are intersections all over the globe, and they mostly vary according to region. Given that only 29% of the Earth's surface is covered by land[3], and that 10% of these regions accommodate 95% of the human population[4], sampling the Earth's surface with random coordinates is not very efficient. Assuming that intersections mostly occur near populated areas, we retrieve these populated candidate regions from GeoNames. To speed up the processing in the next steps, we can also divide the region into smaller subregions if the area of the region exceeds a certain value. The retrieved candidate (sub)regions are converted into lanelet networks via the OSM2CR converter.

B. Extracting Intersections

We denote the n-th forking point and the tuple of all forking points in a lanelet network as P_n and $\mathcal{P}$, respectively. For a given lanelet network, it is usually difficult to determine beforehand the number of intersections to be extracted. For this reason, instead of k-means-like algorithms [34], we apply the hierarchical agglomerative clustering (HAC) algorithm [35] to $\mathcal{P}$. HAC only requires a distance threshold d_{th} to limit the distances between clusters: a higher d_{th} entails larger intersections. Alg. 1 describes how the intersections are extracted from $\mathcal{P}$.

1) Clustering forking points (Alg. 1, lines 2-4): Initially, each forking point forms a cluster C_n with it being the only member: $C_n = \{P_n\}$. We denote the tuple of clusters by $\mathcal{C} := \langle C_1, C_2, \dots \rangle$, and the distance between two clusters C_i, C_j with single-linkage setting [35] by $d_{i,j}$:

$$d_{i,j} = \min\{\text{dist}(a,b) | a \in C_i, b \in C_j\},$$

where the operator $\text{dist}(\cdot)$ returns the Euclidean distance between two given forking points. In each iteration, the two clusters C_i, C_j with the minimum distance $d_{i,j} < d_{\text{th}}$ are merged into a new cluster $C' = C_i \cup C_j$. This process is repeated until no more clusters can be merged. Fig. 4a-4b show the dendrogram for clustering an exemplary lanelet network and the clustered forking points.

2) Creating intersections (Alg. 1, lines 5-10): For each remaining cluster $C' \in \mathcal{C}$, we determine the minimum radius r_{min} of a circle enclosing all forking points within the cluster. We enlarge this radius by a user-defined margin r_{mgn} to span a region of interest. We cut out all lanelets from this region, resulting in an intersection I, and additionally apply the following steps:

1) Lanelets that are not within the region of interest are removed.
2) Due to removed lanelets, we update the successor-predecessor and lateral-adjacency relations.

Fig. 4c shows the extracted intersections $\mathcal{I}$.

Algorithm 1 Extracting Intersections

Inputs: forking points $\mathcal{P}$, distance threshold d_{th}
Output: extracted intersections $\mathcal{I}$

1: $\mathcal{I} \leftarrow \emptyset$
2: ▷ Clustering forking points
3: $\mathcal{C} \leftarrow$ INITIALIZE($\mathcal{P}$)
4: $\mathcal{C} \leftarrow$ HAC($\mathcal{C}$, d_{th})
5: ▷ Creating intersections
6: **for** $C' \in \mathcal{C}$ **do**
7: $I \leftarrow$ CUTLANELETS(C')
8: $I \leftarrow$ POSTPROCESS(I)
9: $\mathcal{I} \leftarrow \mathcal{I} \cup \{I\}$
10: **end for**
11: **return** $\mathcal{I}$

C. Selecting Interesting Intersections

Given the intersections $\mathcal{I}$ extracted from a lanelet network, we only keep those that are particularly interesting or distinct according to the following features:

- number of forking points;
- number of lanelets;
- number of crossing lanelets;
- number of predecessors and successors;
- area of lanelets;
- density of lanelets;
- angle between lanelets; and
- mean distance between forking points and their centroid.

We associate the interestingness of intersections with the dissimilarity between their features and those of other

[3]https://www.noaa.gov/
[4]https://ec.europa.eu/jrc/en

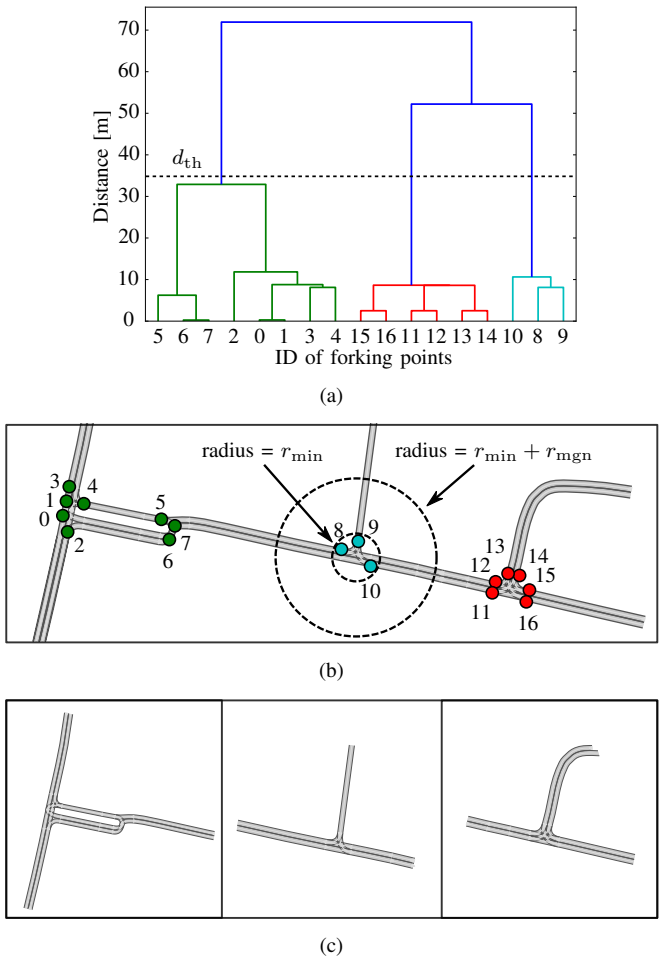


Fig. 4. (a) Dendrogram of the clustering result. Three clusters are generated with d_{th} set to 35 meters. (b) Forking points within one cluster have the same color. r_{mgn} is set to 15 meters. (c) Intersections extracted from the input lanelet network.

intersections. By doing so, we turn the selection of interesting intersections into a multivariate outlier (anomaly) detection problem. To solve this problem, we use the isolation forest (iForest) algorithm [36], since it is unsupervised, capable of efficiently handling multiple dimensions of features, and requires limited effort to hand-tune its parameters. In the training phase, a total of k isolation trees (iTrees) are trained with sets of randomly-selected intersections; in the detection phase, an anomaly score $s \in [0, 1]$ is assigned to each intersection by the iTrees [36], where an intersection with a score above a threshold s_{th} is considered an outlier. Fig. 5 presents a collection of distinct intersections. It should be recalled that we divide the candidate region into subregions if it is overly large, thus rendering the adopted iForest algorithm computationally tractable.

IV. Generation of Safety-Critical Scenarios

On the extracted maps, we simulate traffic participants using our previously-introduced CR-SUMO interface next. Since scenarios simulated with SUMO often yield uncritical, easy-to-solve motion planning problems, we subsequently increase their criticality using our approach [30], [31] for reducing the solution space. We first parametrize the initially obtained trajectories of other traffic participants in Sec. IV-B and, following that, formulate a nonlinear optimization problem with a criticality criterion specified in Sec. IV-D.

A. Motion Planning Problem

The system dynamics of the ego vehicle is defined by

$$\dot{x}_{\text{e}}(t) = f(x_{\text{e}}(t), u(t)),$$

where $x_{\text{e}}(t) \in \mathbb{R}^n$ is the state vector and $u(t) \in \mathcal{U}$ is the input vector with the set of admissible inputs $\mathcal{U} \in \mathbb{R}^m$.

The trajectories of n_{tp} other traffic participants are given by $x_i(t; p)$, $i \in \{1, ..., n_{\text{tp}}\}$ with parameters $p \in \mathbb{R}^{n_{\text{p}}}$, initial time t_0, and final time t_f. Initial candidates for these trajectories are obtained from SUMO; their parametrization is explained in more detail in Sec. IV-B. The occupied space $\mathcal{O}_i(t; p) \subset \mathbb{R}^2$ of a traffic participant is obtained through the $\text{occ}(\cdot)$ operator, i.e., $\mathcal{O}_i(t; p) = \text{occ}(x_i(t; p))$. We define the motion planning problem for the ego vehicle as a classical reach-avoid problem: given an initial state $x_{\text{e},0} = x_{\text{e}}(t_0)$, an input trajectory $u(t)$ has to be found to steer the ego vehicle into a goal region while not leaving the road surface $\mathcal{W}_{\text{lanes}} \in \mathbb{R}^2$ and avoiding the space $\mathcal{O}(t; p)$ occupied by all obstacles, i.e.,

$$\forall t \in [t_0, t_f] : \text{occ}(x_{\text{e}}(t)) \subseteq \mathcal{W}_{\text{lanes}} \backslash \mathcal{O}(t; p). \tag{1}$$

We obtain a motion planning problem by deleting a selected vehicle in a scenario simulated with SUMO and storing the initial state $x_{\text{e},0}$ of this vehicle. Vehicles with interesting maneuvers are automatically selected by using thresholds on the velocity and acceleration profiles or by identifying lane changes, turns or vehicles driving nearby.

B. Scenario Parametrization

In order to optimize the criticality of scenarios, we parametrize trajectories by the parameter vector p. We describe trajectories in lane-based coordinate systems, in which a state is defined as $x = [s_\xi, \dot{s}_\xi, s_\eta, \dot{s}_\eta]^T$. The subscripts ξ and η denote the longitudinal and lateral coordinates with respect to the centerline, respectively (see Fig. 2).

For the n_{tp} traffic participants, we only parametrize the longitudinal trajectory using translations $p^{\text{s}} \in \mathbb{R}^{n_{\text{tp}}}$, initial velocity variations $p^{\text{v}} \in \mathbb{R}^{n_{\text{tp}}}$, and acceleration variations $p^{\text{a}} \in \mathbb{R}^{n_{\text{tp}}}$, yielding $p = \left[p^{\text{s}}, p^{\text{v}}, p^{\text{a}}\right]^T$. The parametrized longitudinal position trajectory is given by

$$s_{\xi,i}(t; p_i) = \hat{s}_{\xi,i}(t) + p_i^{\text{s}} + p_i^{\text{v}} t + \frac{1}{2} p_i^{\text{a}} t^2. \tag{2}$$

From (2), the centerlines, and the dimensions of the vehicle, we obtain the occupied space $\mathcal{O}_i(t, p)$ of each traffic participant.

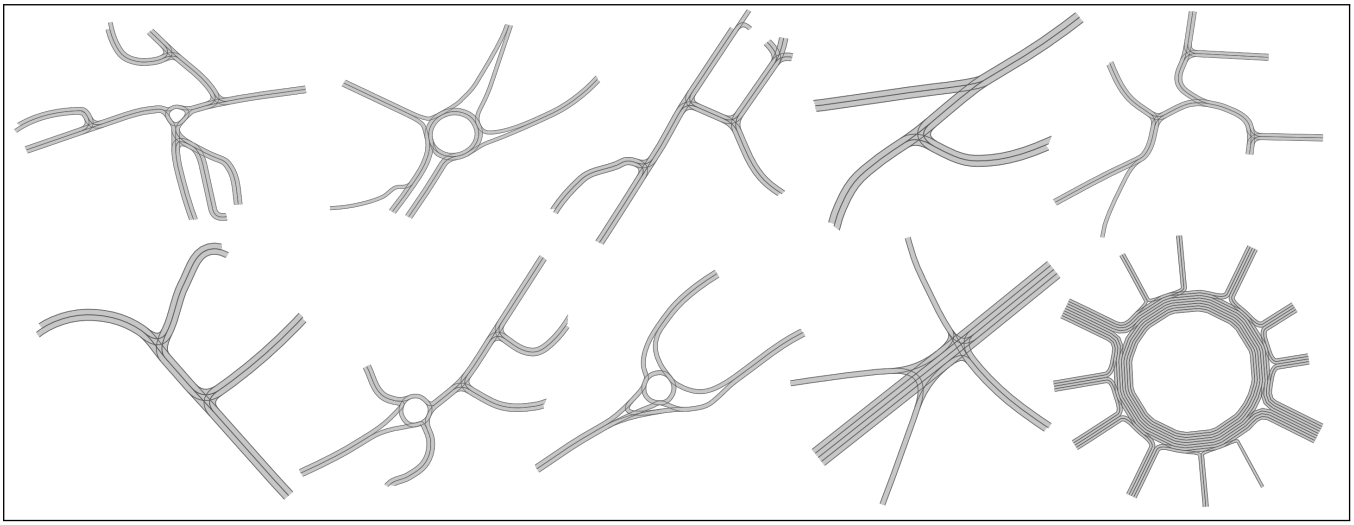


Fig. 5. Selected road intersections generated by Globetrotter ($s_{\mathrm{th}} = 0.9$).

C. Drivable Area

We denote a feasible solution to the motion planning problem defined in Sec. IV-A as $\chi(t; x_0, u(\cdot))$, where $u(\cdot)$ refers to the entire trajectory instead of a particular value $u(t)$ at time t. To quantify the criticality of a scenario, we use the solution space, which corresponds to the set of reachable states for $t \in [t_0, t_f]$ without collisions:

$$\mathcal{R}(t; x_0, \mathcal{O}(\,\cdot\,; p)) = \Big\{ \chi(t; x_0, u(\cdot)) \;\Big|\; \forall \tau \in [t_0, t_f] : u(\tau) \in \mathcal{U}, \; \mathrm{occ}\big(\chi(\tau; x_0, u(\cdot))\big) \subseteq \mathcal{W}_{\mathrm{lanes}} \backslash \mathcal{O}(\tau; p) \Big\}.$$

By applying the projection operator $\mathrm{proj}(x) : \mathbb{R}^n \to \mathbb{R}^2$, which projects the state space to the position domain, we obtain the *drivable area*

$$\mathcal{D}(t; x_0, \mathcal{O}(\,\cdot\,; p)) = \bigcup_{x \in \mathcal{R}(t; x_0, \mathcal{O}(\,\cdot\,; p))} \mathrm{proj}(x). \tag{3}$$

An example for the drivable area in presence of an obstacle is depicted in Fig. 6.

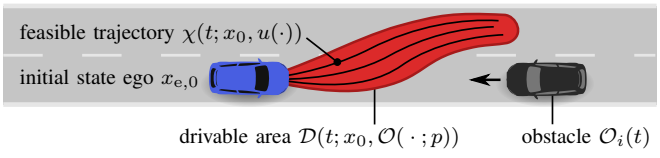


Fig. 6. Example of a drivable area for the time interval $[t_0, t_f]$.

To quantify the solution space, we introduce the function $\mathrm{area}(\mathcal{X})$ returning the area of a set. We write

$$A(t; p) \coloneqq \mathrm{area}\big(\mathcal{D}(t; x_0, \mathcal{O}(\,\cdot\,; p))\big)$$

to obtain the area profile of the drivable area over time. We compute the drivable area using our approach as in [37].

D. Optimization Problem

For increasing the criticality of the motion planning problem, we optimize the parameter vector p to obtain a desired, critical area profile $A_{\mathrm{crit}}(t)$:

$$\underset{p}{\mathrm{argmin}}\, \kappa(p), \quad \kappa(p) = \int_0^{t_f} \big(A(t; p) - A_{\mathrm{crit}}(t)\big)^2 \tag{4}$$

$$\text{subject to} \quad \forall t, \forall i, \forall j \neq i : \quad \mathcal{O}_i(t; p) \cap \mathcal{O}_j(t; p) = \emptyset. \tag{5}$$

The constraint in (5) ensures that no traffic participants collide with each other. In this work, we use the drivable area computed without any traffic participants and the scalar $\gamma \in \,]0, 1[$ which quantifies the reduction of the drivable area: $A_{\mathrm{crit}}(t) = \gamma \cdot \mathrm{area}(\mathcal{D}(t; x_0, \emptyset))$.

Since the drivable area is highly nonlinear with respect to the trajectories of other traffic participants and possibly subjected to local minima, we use particle swarm optimization [38] as in our previous work [30]. Furthermore, we implement a repair algorithm that enforces the collision constraint (5). To that end, we formulate the collision constraints as linear inequality constraints and correct infeasible solutions by computing the closest feasible solution using linear programming. A more efficient optimization is ensured by an a priori computation of relevant parameter intervals as presented in [30].

V. Evaluation

We demonstrate our approach by generating scenarios on a large variety of road networks from various places across the world. First, we obtain 576 road networks from 8 countries and 46 cities from Globetrotter, for each of which we simulate multiple scenarios using our CR-SUMO interface. After selecting interesting ego vehicles, we obtain 1402 scenarios for which we optimize the criticality. The resulting scenarios are added to our website[5].

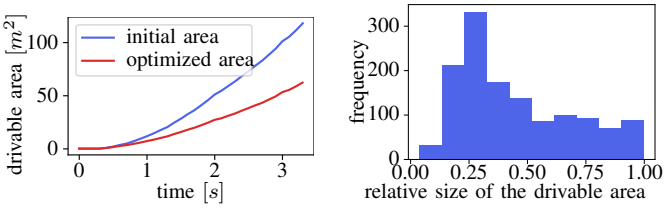
[5]https://commonroad.in.tum.de/

In Fig. 7a we compare the area profiles $A(t;p)$ of the drivable area in the optimized scenarios against the initial scenario obtained from SUMO: our approach is able to significantly decrease the drivable area, and it thus increases the criticality. Fig. 7b shows the distribution of the achieved reduction of the critical area. For most of the optimized scenarios, the drivable area ranges between 0.2 – 0.3 of its initial size.

TABLE I

PARAMETERS FOR CRITICALITY OPTIMIZATION

Drivable area computation	
max. acceleration ego vehicle $\lvert a_{max} \rvert$	$5.0\,\mathrm{m/s^2}$
time step size Δt	$0.1\,\mathrm{s}$
time horizon t_f	$3.4\,\mathrm{s}$
Constraints for optimization	
initial velocity variation	$[-3, 3]\,\mathrm{m/s}$
acceleration variation	$[-5, 2]\,\mathrm{m/s^2}$



(a) (b)

Fig. 7. (a) Size of the drivable area $A(t;p)$ over time, averaged over all scenarios. (b) Histogram of the size of the drivable area in the optimized scenarios relative to the initial scenarios.

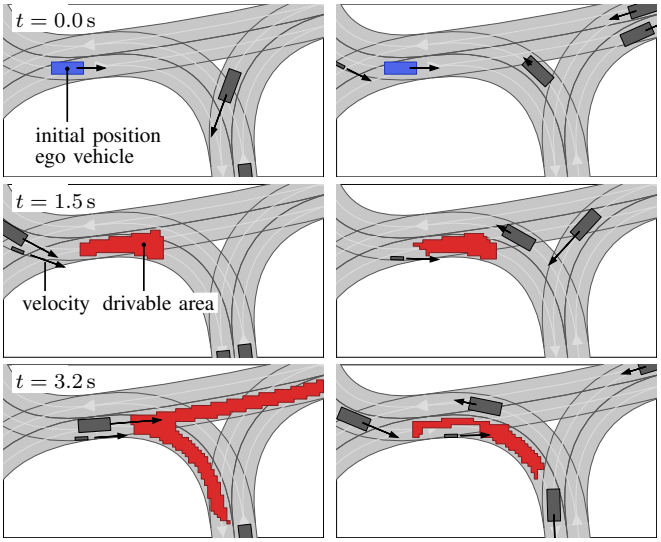
Let us present some concrete examples for demonstrating our algorithm. The first example is an intersection from the town Pula, Croatia. In Fig. 8, we compare the drivable area of the initial scenario obtained from SUMO with the optimized scenario. Note that we restrict the allowed road surface $\mathcal{W}_{\text{lanes}}$ to lanelets that the ego vehicle is allowed to drive in. In the initial scenario, the ego vehicle could either turn freely to the right or drive straight. However, after the optimization, two turning vehicles and a bicycle restrict possible maneuvers of the ego vehicle.

The second example is a four-way intersection from the town Putte, Belgium. In the optimized scenario, the ego vehicle must either respect an oncoming vehicle when turning left or a bicycle when driving straight. As a result, the drivable area is split into two parts, as shown in Fig. 9.

VI. CONCLUSIONS

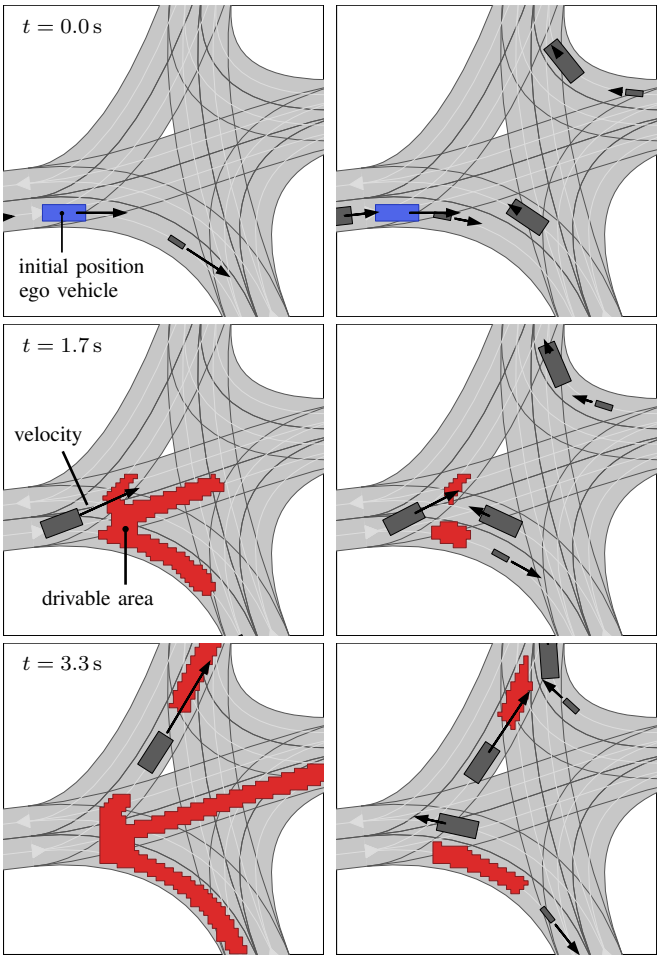
We present an approach to automatically generate a large number of test scenarios for automated vehicles. Our results show that we are able to extract a large number of distinct road networks from OpenStreetMaps, for which we simulate traffic scenarios using the traffic simulator SUMO. Our approach subsequently yields challenging scenarios by decreasing the solution space for motion planning algorithms.

The generated, publicly-available scenarios render the virtual testing of motion-planning algorithms in challenging situations easier. In the future, the explicit consideration of traffic rules during the generation of our critical scenarios will further improve our test cases.



(a) Initial scenario from simulation. (b) Optimized, more critical scenario.

Fig. 8. Example 1: Comparison of the drivable areas at different times.



(a) Initial scenario from simulation. (b) Optimized, more critical scenario.

Fig. 9. Example 2: Comparison of the drivable areas at different times.

ACKNOWLEDGMENTS

The authors would like to thank Chuxuan Li for her work on the SUMO interface and Maximilian Rieger for his work on the OSM2CR converter. We gratefully acknowledge the financial support from the Central Innovation Programme of the German Federal Government under grant no. ZF4086007BZ8 and the German Research Foundation (DFG) within the Priority Programme SPP 1835 *Cooperative Interacting Automobiles* under grant no. AL 1185/4-2.

REFERENCES

[1] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transp. Res. Part A: Policy Pract.*, vol. 94, pp. 182–193, 2016.

[2] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, "INTERACTION dataset: An INTERnational, Adversarial and Cooperative moTION dataset in interactive driving scenarios with semantic maps," *arXiv:1910.03088*, 2019.

[3] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Lyft Level 5 AV dataset 2019," https://level5.lyft.com/dataset/, 2019.

[4] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo Open dataset," *arXiv:1912.04838*, 2019.

[5] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 2118–2125.

[6] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, 2020.

[7] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, 2008.

[8] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO–simulation of urban mobility: an overview," in *Proc. of Int. Conf. Adv. Syst. Simul.*, 2011, pp. 63–68.

[9] C. Campos, J. M. Leitão, J. P. Pereira, A. Ribas, and A. F. Coelho, "Procedural generation of topologic road networks for driving simulation," in *Iberian Conf. Inf. Syst. Technol.*, 2015, pp. 1–6.

[10] B. Kim, A. Jarandikar, J. Shum, S. Shiraishi, and M. Yamaura, "The SMT-based automatic road network generation in vehicle simulation environment," in *Proc. of the ACM Int. Conf. Embed. Softw.*, 2016, pp. 1–10.

[11] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proc. of the 28th ACM SIGSOFT Int. Symposium on Software Testing and Analysis*, 2019, pp. 318–328.

[12] G. Máttyus, W. Luo, and R. Urtasun, "DeepRoadMapper: Extracting road topology from aerial images," in *Proc. of the IEEE Int. Conf. Comput. Vision*, 2017, pp. 3438–3446.

[13] M. Maboudi, J. Amini, M. Hahn, and M. Saati, "Road network extraction from VHR satellite images using context aware object feature integration and tensor voting," *Remote Sens.*, vol. 8, no. 8, 2016.

[14] P. Li, Y. Zang, C. Wang, J. Li, M. Cheng, L. Luo, and Y. Yu, "Road network extraction via deep learning and line integral convolution," in *Proc. of the Int. Geosci. Remote Sens. Symp.*, 2016, pp. 1599–1602.

[15] Y. Zang, C. Wang, Y. Yu, L. Luo, K. Yang, and J. Li, "Joint enhancing filtering for road network extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 3, pp. 1511–1525, 2016.

[16] Y. Y. Chiang and C. A. Knoblock, "Automatic extraction of road intersection position, connectivity, and orientations from raster maps," in *Proc. of the ACM Int. Symp. Adv. Geogr. Inf. Syst.*, 2008, pp. 183–192.

[17] P. Fischer, S. M. Azimi, R. Roschlaub, and T. Krauß, "Towards HD maps from aerial imagery: Robust lane marking segmentation using country-scale imagery," *Int. J. Geo-Inf.*, vol. 7, no. 12, 2018.

[18] A. Zang, Z. Li, R. Xu, and D. Doria, "Lane boundary extraction from satellite imagery," in *Proc. of the ACM SIGSPATIAL Workshop High-Precis. Maps Intell. Appl. Auton. Veh.*, 2017, pp. 1–8.

[19] A. Artunedo, J. Godoy, and J. Villagra, "Smooth path planning for urban autonomous driving using OpenStreetMaps," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 837–842.

[20] D. Krajzewicz, G. Hertkorn, and J. Ringel, "Preparation of digital maps for traffic simulation; part 1: approach and algorithms," in *Proc. Ind. Simul. Conf.*, 2005, pp. 285–290.

[21] D. Nalic, A. Eichberger, G. Hanzl, M. Fellendorf, and B. Rogic, "Development of a co-simulation framework for systematic generation of scenarios for testing and validation of automated driving systems," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2019, pp. 1895–1901.

[22] P. Riegl, A. Gaull, and M. Beitelschmidt, "A tool chain for generating critical traffic situations for testing vehicle safety functions," in *IEEE Int. Conf. on Vehicular Electronics and Safety*, 2019, pp. 1–6.

[23] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 595–607, 2017.

[24] S. Feng, Y. Feng, C. Yu, Y. Zhang, and H. X. Liu, "Testing scenario library generation for connected and automated vehicles, part I: Methodology," *arXiv:1905.03419*, 2020.

[25] F. Hauer, A. Pretschner, and B. Holzmüller, "Fitness functions for testing automated and autonomous driving systems," in *Proc. of the Int. Conf. Comput. Safety, Rel., Security*, 2019, pp. 69–84.

[26] H. Beglerovic, M. Stolz, and M. Horn, "Testing of autonomous vehicles using surrogate models and stochastic optimization," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1129–1134.

[27] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing S-TaLiRo as an automatic test generation framework for autonomous vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2016, pp. 1470–1475.

[28] M. Koschi, C. Pek, S. Maierhofer, and M. Althoff, "Computationally efficient safety falsification of adaptive cruise control systems," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2019, pp. 2879–2886.

[29] A. Nonnengart, M. Klusch, and M. Christian, "CriSGen : Constraint-based generation of critical scenarios for autonomous vehicles," in *Proc. of the Int. Workshop on Formal Methods for Autonomous Systems*, 2019.

[30] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *Proc. of the IEEE Intell. Veh. Symp.*, 2019, pp. 2352–2358.

[31] M. Althoff and S. Lutz, "Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2018, pp. 1326–1333.

[32] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intell. Veh. Symp.*, 2014, pp. 420–425.

[33] M. Klischat, O. Dragoi, M. Eissa, and M. Althoff, "Coupling SUMO with a motion planning framework for automated vehicles," in *SUMO: Simulating Connected Urban Mobility*, 2019.

[34] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[35] F. Murtagh and P. Legendre, "Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion?" *J. Classif.*, vol. 31, no. 3, pp. 274–295, 2014.

[36] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. of the IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.

[37] M. Klischat and M. Althoff, "A multi-step approach to accelerate the computation of reachable sets for road vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2020.

[38] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. of the Int. Symp. Micro Mach. Human Sci.*, 1995, pp. 39–43.

3.3 A Multi-Step Approach to Accelerate the Computation of Reachable Sets for Road Vehicles

When using the drivable area of the SUT as a criticality metric in the numerical optimization for solving Problem 1, we require an efficient method for computing the drivable area. Later, this enables the generation of safety-critical scenarios on a larger scale. Hence, we introduce in this work a new method for computing the drivable area for road vehicles in dynamic traffic scenarios that is more efficient than previous methods. By dividing the reachability analysis into an online and an offline computation, we avoid computationally expensive operations during runtime by exploiting invariances of the reachability analysis that allow reusing precomputed results: In the offline computation, we discretize the drivable area at each time step and compute a graph representing the reachability of the discretized subsets from previous time steps. During the online computation, subsets occupied by obstacles are removed from the drivable area and the graph is used to propagate the reachability to subsequent time steps. Crucial for limiting the over-approximation is our method that utilizes the reachability edges from multiple preceding time steps at once for each propagation. This reduces the discretization error that is added at each time step, which could sum up over time resulting in a so-called "wrapping effect".

Our evaluation using a dataset of more than 300 scenarios from urban and highway environments demonstrates that this method is significantly faster than a method without precomputations while providing comparable accuracy. Furthermore, the evaluation shows that the proposed multi-step propagation can significantly decrease the degree of overapproximation.

Contributions M. K. initiated the idea and developed the concept for dividing the reachability analysis into offline and online computations and for the multi-step propagation. M. K. conducted the evaluation and wrote the article. M. A. provided feedback and helped to improve the article.

Conference Article In this thesis, the accepted version submitted by the author is reprinted. The edited version is available under https://doi.org/10.1109/ITSC45102.2020.9294328.

A Multi-Step Approach to Accelerate the Computation of Reachable Sets for Road Vehicles

Conference Proceedings:
2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)

Author: Moritz Klischat; Matthias Althoff

Publisher: IEEE

Date: 20-23 Sept. 2020

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

A Multi-Step Approach to Accelerate the Computation of Reachable Sets for Road Vehicles

Moritz Klischat and Matthias Althoff

***Abstract*— We propose an approach for the fast computation of reachable sets of road vehicles while considering dynamic obstacles. The obtained reachable sets contain all possible behaviors of vehicles and can be used for motion planning, verification, and criticality assessment. The proposed approach precomputes computationally expensive parts of the reachability analysis. Further, we partition the reachable set into cells and construct a directed graph storing which cells are reachable from which cells at preceding time steps. Using this approach, considering obstacles reduces to deleting nodes from the directed graph. Although this simple idea ensures an efficient computation, the discretization can introduce considerable over-approximations. Thus, the main novelty of this paper is to reduce the over-approximations by intersecting reachable sets propagated from multiple points in time. We demonstrate our approach on a large range of scenarios for automated vehicles showing a faster computation time compared to previous approaches while providing the same level of accuracy.**

I. Introduction

Reachability analysis is considered a powerful tool to ensure safety for safety-critical applications such as self-driving vehicles. Although reachability analysis is a well-researched topic with continuous improvements in terms of scalability and/or tightness [1]–[3], most approaches do not consider time-varying forbidden regions originating from static or dynamic obstacles.

However, reachable sets excluding forbidden regions are especially useful for motion planning, e.g., for restricting the search space to safe regions [4]. Also, the size of reachable sets can be used to assess the criticality of traffic scenarios for generating safety-critical test cases for motion planners [5]. Another application of reachable sets is cooperative path planning for multiple agents [6], [7] or the computation of the time-to-react (TTR), i.e., the last point in time to avoid a collision [8]. Most of the above-mentioned applications are used in real time and require a fast computation. In this paper, we propose a novel method that computes reachable sets excluding forbidden regions more efficiently compared to previous work.

A. Related Work

General approaches which are based on Hamilton–Jacobi–Bellmann (HJB) equations are proposed in [9]–[11]. However, for real-time applications this method is computationally too expensive. An early work computing reachable sets for trajectory planning of vehicles is [12]. In [13], reachable sets are computed offline for parametrized trajectories with constant inputs. During run time, parameters of the collision-free reachable sets are selected to determine the set of safe inputs for the subsequent trajectory optimization. The method in [14] approximates the reachable set of automated vehicles using HJB equations. However, the method is restricted to rectangular obstacles and simple road configurations.

A related topic where obstacles are considered in reachability analysis is the computation of inevitable collision states (ICS), which eventually lead to a collision irrespective of the chosen input [15]. Works that compute ICS in dynamic environments using reachability analysis can be found in [9], [16], [17].

To emphasize that we are ultimately interested in reachable sets avoiding forbidden region projected to the road surface, we use the term *drivable area* henceforth. In our previous work [18], we also computed the drivable area, but that approach requires recomputing similar computations, which unnecessarily consumes computational resources.

B. Contributions

We propose a novel method for the graph-based computation of the drivable area. A similar graph-based method was used in [19]; however, the reachability was only approximated and not based on system dynamics. Although a spatio-temporal decomposition of the state space for reachability analysis was used, e.g., in [20]–[22], the novelty of our work is that

- it can limit the discretization error by considering multiple preceding time steps at every iteration;
- our algorithm can handle arbitrary obstacle shapes and road networks;
- ICS can be considered less conservatively compared with our previous work [18];
- we provide extensive testing and benchmarking on a large number of traffic scenarios.

The rest of the paper is organized as follows. First, we introduce the problem statement in Sec. II, before describing the proposed method in Sec. III, which is divided into offline and online computation. Finally, we evaluate our approach in Sec. IV using multiple numerical examples and compare it to related works.

II. Definitions and Problem Statement

Let the dynamics of a model M be given by $\dot{x}(t) = f(x(t), u(t)$ with inputs $u(t) \in \mathcal{U}$ bounded by the input set $\mathcal{U} \subset \mathbb{R}^m$. A solution originating from the initial state

All authors are with the Technische Universität München, Fakultät für Informatik, Lehrstuhl für Robotik und Echtzeitsysteme, Boltzmannstraße 3, 85748, Garching, Germany. {moritz.klischat, althoff}@tum.de

$x_0 \in \mathbb{R}^n$ is

$$x(t; u(\cdot), x_0) = x_0 + \int_{t_0}^{t} f(x(\tau), u(\tau)) d\tau \,, \tag{1}$$

where $x(t; u(\cdot), x_0) \in \mathbb{R}^n$ and $u(\cdot)$ denotes an input trajectory in contrast to points in time t. Because we often require the projection of states to the two-dimensional position domain, we define the projection operator $\mathrm{proj}(x) : \mathbb{R}^n \to \mathbb{R}^2$.

From the perspective of the vehicle, the possible future occupancy of obstacles and regions outside of the road surface are a set of forbidden states $\mathcal{F}(t) \subset \mathbb{R}^2$ dependent on time t. The anticipated reachable set is defined as the set of reachable states starting from an initial state x_0 while avoiding a set of forbidden states $\mathcal{F}(t)$ during time interval $t \in [t_0, t_h]$ [18]:

$$\begin{aligned} &\operatorname*{reach}_{t}(\mathcal{X}_0, \mathcal{U}, \mathcal{F}(t)) \coloneqq \{x(t; u(\cdot), x_0) \mid x_0 \in \mathcal{X}_0, \\ &\forall \tau \in [t_0, t_h] \colon\ u(\tau) \in \mathcal{U},\ \mathrm{proj}(x(\tau; u(\cdot), x_0)) \notin \mathcal{F}(\tau)\} \,. \end{aligned} \tag{2}$$

The time-dependent reachable set is computed iteratively for time increments $\Delta t \in \mathbb{R}^+$. Hence, we denote a set at time t_k by a subscript k. At each point in time $t_k = k \cdot \Delta t$ with $k \in \mathbb{N}$, we denote the reachable set by

$$\mathcal{R}_{k+1} \coloneqq \operatorname*{reach}_{t_{k+1}}(\mathcal{R}_k, \mathcal{U}, \mathcal{F}(t)), \quad \mathcal{R}_0 = \mathcal{X}_0. \tag{3}$$

When no exclusion of forbidden sets is considered ($\mathcal{F}(t) = \emptyset$) we write $\hat{\square}$, e.g., $\hat{\mathcal{R}}_k$. Since an exact solution of the drivable area $\mathrm{proj}(\mathcal{R}_k)$ cannot be computed for the general case [23], we compute it over-approximately. Moreover, we model road vehicles by a point-mass model M_a because it abstracts any high-fidelity model M for a road vehicle whose dynamics are bounded by the friction circle $\ddot{x}_\zeta^2 + \ddot{x}_\eta^2 \leq a_\mathrm{max}^2$ [12]:

$$\begin{pmatrix} \dot{x}_\zeta(t) \\ \ddot{x}_\zeta(t) \\ \dot{x}_\eta(t) \\ \ddot{x}_\eta(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_\zeta(t) \\ \dot{x}_\zeta(t) \\ x_\eta(t) \\ \dot{x}_\eta(t) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_\zeta(t) \\ u_\eta(t) \end{pmatrix} \tag{4}$$

$$u(t) \in \mathcal{U},\ \mathcal{U} = \left\{u \mid u_\zeta^2 + u_\eta^2 \leq a_\mathrm{max}^2\right\}. \tag{5}$$

This abstraction guarantees that the drivable area of the high-fidelity model $\mathrm{proj}(\mathcal{R}_k^M)$ is always a subset of the abstracted model $\mathrm{proj}(\mathcal{R}_k^{M_a})$, i.e., $\mathrm{proj}(\mathcal{R}_k^M) \subseteq \mathrm{proj}(\mathcal{R}_k^{M_a})$.

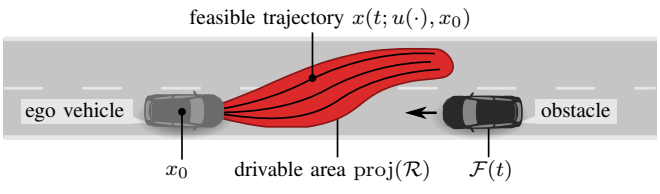


Fig. 1: The drivable area for a complete prediction horizon that represents the set of all feasible trajectories for the ego vehicle.

III. Concept of Offline and Online Computation

We divide the computation of the drivable area into two parts: offline and online. During the offline computation, the drivable area is computed without forbidden sets $\mathcal{F}(t)$, which are only known during online execution. The result is partitioned using a uniform grid of disjoint axis-aligned cells in the position domain

$$\mathcal{C}_k^{(i)} = \left[\underline{c}^{(i)}, \overline{c}^{(i)}\right] \subset \mathbb{R}^2, \quad \bigcup_{i=0}^{n_\mathrm{c}} \mathcal{C}_k^{(i)} \supseteq \mathrm{proj}(\hat{\mathcal{R}}_k), \tag{6}$$

where the index i refers to the i^th cell. We define a directed graph as a tuple $\mathcal{G} = (V, E)$ where nodes $v_{k,i} \in V$ correspond to the cells $\mathcal{C}_k^{(i)}$ of the drivable area and edges $(v_{k,i}, v_{k+1,j}) \in E$ express that a trajectory from one cell to another cell exists.

Thus, forbidden sets can be excluded during the online computation by deleting occupied nodes and their outgoing edges from the graph as illustrated in Fig. 2. Since only the position domain in $\mathbb{R}^2$ is discretized, the number of nodes only scales quadratically with the number of segments in each dimension. We compensate over-approximations from the discretization by adding edges between nodes spanning multiple time steps, as explained in Sec. III-C.

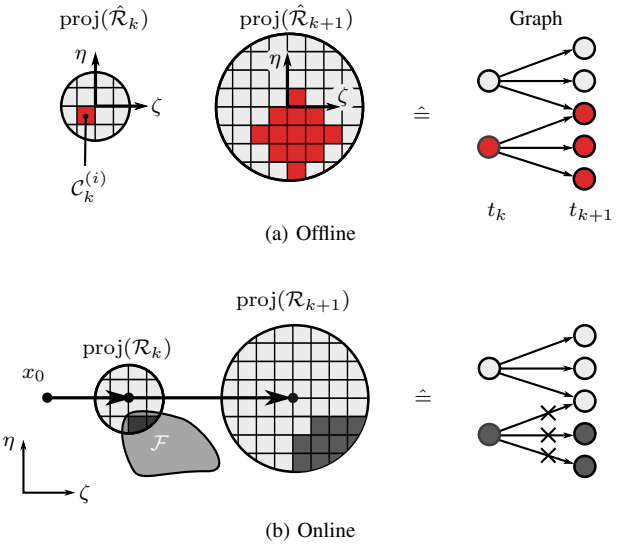


Fig. 2: (a) Offline computation of reachable cells $\mathcal{C}_{k+1}$ at t_{k+1} which is encoded in a graph. (b) Online: deleting nodes (represented in dark gray) that correspond to cells which intersect with forbidden sets $\mathcal{F}(t_k)$ or do not have a predecessor.

A. Offline Reachability Analysis

The offline computation consists of two steps:

1) *Propagation* of the reachable set $\hat{\mathcal{R}}_k$ to obtain $\hat{\mathcal{R}}_{k+1}$.
2) *Discretization* of the reachable set $\hat{\mathcal{R}}_{k+1}$ to $\hat{\mathfrak{D}}_{k+1}$ and construction of the graph.

1) Propagation: Since the vehicle model in (4) is linear, the superposition principle can be applied: The reachable set $\hat{\mathcal{R}}_k$ is obtained by adding the homogeneous solution $x(t_k; u{=}\mathbf{0}, x_0{\neq}\mathbf{0})$ resulting from the initial state x_0 and the inhomogeneous solution $x(t_k; u{\neq}\mathbf{0}, x_0{=}\mathbf{0})$. Since

$\mathcal{U}$ is a constant set, the set of inhomogeneous solutions $\mathrm{reach}_{t_k}(\mathbf{0}, \mathcal{U}, \emptyset)$ can be computed offline for an initial state located at the origin $\mathbf{0}$. For further derivation let us introduce the Minkowski sum of a singelton a and a set $\mathcal{Y}$ as $a \oplus \mathcal{Y} := \{a + y \mid y \in \mathcal{Y}\}$. During online execution, the reachable sets for an arbitrary initial state x_0 are obtained as

$$\underbrace{\underset{t_k}{\mathrm{reach}}(x_0, \mathcal{U}, \emptyset)}_{\hat{\mathcal{R}}_k} = \underbrace{\underset{t_k}{\mathrm{reach}}(x_0, \mathbf{0}, \emptyset)}_{x(t_k;0,x_0)\ :\ \text{online}} \oplus \underbrace{\underset{t_k}{\mathrm{reach}}(\mathbf{0}, \mathcal{U}, \emptyset)}_{\hat{\mathcal{R}}_k^{\mathrm{ori}}\ :\ \text{offline}}. \quad (7)$$

Since no forbidden set is excluded during the offline computation, the propagation step

$$\hat{\mathcal{R}}_{k+1}^{\mathrm{ori}} = \underset{t_{k+1}}{\mathrm{reach}}(\hat{\mathcal{R}}_k^{\mathrm{ori}}, \mathcal{U}, \emptyset) \quad (8)$$

can be computed with standard tools for reachability analysis; a non-exhaustive list is given by Flow* [24], SpaceEx [25], C2E2 [26], JuliaReach [27], and CORA [28].

2) Discretization of $\hat{\mathcal{R}}_{k+1}$ and Construction of the Graph: For creating the graph $\mathcal{G}$, we first partition the reachable set $\hat{\mathcal{R}}_{k+1}^{\mathrm{ori}}$ into disjoint subsets $\mathcal{B}_{k+1}^{(j)}$ using the cells in the position domain (6) to compute the Cartesian product

$$\forall i \in \mathcal{I}\colon \mathcal{B}_k^{(i)} = \mathcal{C}^{(i)} \times [\underline{\dot{x}}_{k,\zeta}^{(i)}, \overline{\dot{x}}_{k,\zeta}^{(i)}] \times [\underline{\dot{x}}_{k,\eta}^{(i)}, \overline{\dot{x}}_{k,\eta}^{(i)}],$$
$$\mathcal{I} = \left\{ i \mid \hat{\mathcal{R}}_{k+1}^{\mathrm{ori}} \cap \mathcal{C}^{(i)} \neq \emptyset \right\}. \quad (9)$$

The intervals $[\underline{\dot{x}}_{k,\zeta}^{(i)}, \overline{\dot{x}}_{k,\zeta}^{(i)}]$ and $[\underline{\dot{x}}_{k,\eta}^{(i)}, \overline{\dot{x}}_{k,\eta}^{(i)}]$ bound the velocities of the reachable set $\hat{\mathcal{R}}_{k+1}^{\mathrm{ori}}$ for the i^{th} cell. To obtain a discretized representation of a reachable set, we introduce the discretization operator:

$$\hat{\mathfrak{D}}_k^{\mathrm{ori}} = \mathrm{discr}(\hat{\mathcal{R}}_k^{ori}) = \bigcup_{i=0}^{n_b} \mathcal{B}_k^{(i)} \supseteq \hat{\mathcal{R}}_k, \quad (10)$$

which analogously yields $\mathfrak{D}_k = \mathrm{discr}(\mathcal{R}_k)$. An edge $(v_{k,i}, v_{k+1,j})$ is added to $\mathcal{G}$ if

$$\underset{t_{k+1}}{\mathrm{reach}}(\mathcal{B}_k^{(i)}, \mathcal{U}, \emptyset) \cap \mathcal{B}_{k+1}^{(j)} \neq \emptyset. \quad (11)$$

For efficient implementation, we represent the edges by adjacency matrices $P_k^{k+1} \in \mathbb{R}^{q_k \times q_{k+1}}$ with q_k being the number of cells at the respective time. Each element p_{ji} of P_k^{k+1} is a Boolean value

$$p_{ji} = \begin{cases} 1 & \text{if } \mathrm{reach}_{t_{k+1}}(\mathcal{B}_k^{(i)}, \mathcal{U}, \emptyset) \cap \mathcal{B}_{k+1}^{(j)} \neq \emptyset \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

The adjacency matrices P_k^{k+1} are the main result of the offline computation and can be stored compactly as sparse matrices.

B. Online Computations

The objective of the online computation is to exclude forbidden sets $\mathcal{F}_k$ from the drivable area. Therefore, at every iteration step the drivable area is propagated and cells intersecting with $\mathcal{F}_k$ are excluded.

Algorithm 1 Offline Reachability Analysis

Require: input set $\mathcal{U}$

1: **for** $k = 0$ to n **do**
2: $\quad \hat{\mathcal{R}}_{k+1}^0 \leftarrow \mathrm{reach}_{t_{k+1}}(\hat{\mathcal{R}}_k^0, \mathcal{U})$
3: $\quad \hat{\mathfrak{D}}_{k+1} \leftarrow \mathrm{discr}(\hat{\mathcal{R}}_{k+1}^0)$ ▷ see (10)
4: $\quad \{v_i, v_j\} \leftarrow$ ReachableCells($\hat{\mathfrak{D}}_k, \hat{\mathfrak{D}}_{k+1}$) ▷ see (11)
5: $\quad P_k^{k+1} \leftarrow$ ConstructGraph($\{(\mathcal{B}_k^{(i)}, \mathcal{B}_{k+1}^{(j)})\}$) ▷ see (12)
6: **end for**
7: **return** $P_k^{k+1}, \mathfrak{D}_k$

1) Propagation: To propagate the drivable area using the adjacency matrices P_k^{k+1}, we introduce the Boolean vector $r_k \in \{0,1\}^q$ that denotes for each cell whether it is part of the drivable area:

$$r_k^{(i)} = \begin{cases} 1 & \text{if } \left(x(t_k;0,x_0) \oplus \mathcal{C}_k^{(i)}\right) \cap \mathrm{proj}(\mathfrak{D}_k) \neq \emptyset \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

Using (12), we write the graph-based propagation as

$$\hat{r}_{k+1} = P_k^{k+1} r_k. \quad (14)$$

2) Discretization of Obstacles: To exclude $\mathcal{F}_{k+1}$ from the Boolean representation $\hat{r}_{k+1}$, we discretize $\mathcal{F}_{k+1}$. As commonly done in motion planning, we consider the shape of the ego vehicle by dilating the occupied space of obstacles with a disk of radius ρ that under-approximates the shape of the ego vehicle [29]. The under-approximation is required to consistently over-approximate the drivable area when excluding $\mathcal{F}_{k+1}$. The resulting occupied space $\mathcal{F}_{k+1}$ is represented analogously to r_k by the occupancy vector o_{k+1} with elements

$$o_{k+1}^{(i)} = \begin{cases} 1 & \text{if } x(t_k;0,x_0) \oplus \mathcal{C}_{k+1}^{(i)} \subseteq \mathcal{F}_{k+1} \\ 0 & \text{otherwise} \end{cases}. \quad (15)$$

$\mathcal{F}_{k+1}$ is typically not connected and thus, occupancies from multiple obstacles need to be discretized individually. For efficiency, the discretization of an obstacle is only conducted if it intersects with the bounding box of the drivable area. From (13) and (15) follows that the exclusion of the occupied states is equivalent to

$$r_{k+1} = \hat{r}_{k+1} \wedge \neg o_{k+1}$$

with logical operators $\neg$ and $\wedge$ being performed elementwise. Thus, we can write the propagation of the reachable set and exclusion of forbidden sets as

$$\mathfrak{D}_{k+1} = \mathrm{discr}\left(\underset{t_{k+1}}{\mathrm{reach}}(\mathfrak{D}_k, \mathcal{U}, \emptyset)\right) \setminus \mathrm{discr}(\mathcal{F}(t_{k+1})),$$
$$\mathfrak{D}_0 = x_0, \quad (16)$$

which using (14) simplifies to

$$r_{k+1} = \left(P_k^{k+1} r_{t_k}\right) \wedge \neg o_{k+1}. \quad (17)$$

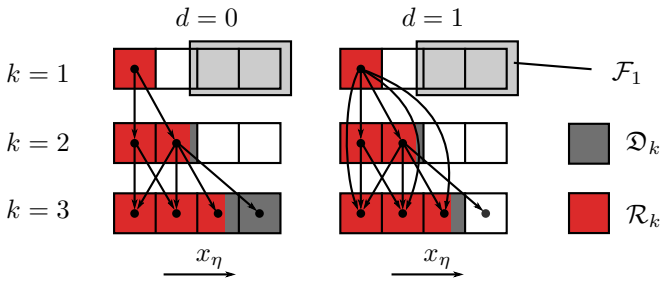


Fig. 3: One-dimensional example for the intersection of multiple propagated sets to reduce the discretization error of $\mathfrak{D}_3$. For $d = 0$, only cells at subsequent time steps are connected in the graph; for $d = 1$, edges from $k = 1$ to $k = 3$ are also considered and only those cells with an edge to $k = 1$ are reachable at $k = 3$. The undiscretized drivable area $\mathcal{R}$ is shown for comparison.

C. Compensating Discretization Errors with Multiple Propagations

When propagating the drivable area $\mathrm{proj}(\mathfrak{D}_k)$, the discretization accumulates as shown in Fig. 3. After multiple time steps, these errors quickly lead to an undesirably over-approximated drivable area. Furthermore, we over-approximate the velocities so that we only have to discretize the position domain.

To counteract the discretization errors, we add edges between cells of the drivable area that span multiple time steps. Using these edges, drivable areas from multiple preceding points in time are propagated up to time t_{k+1} and the resulting sets are intersected. Since the edges spanning multiple time steps are also computed offline using the same principle as in Sec. III-A, the discretization error is only added once, instead of aggregating the discretization errors from every intermediate time step as illustrated in Fig. 3.

Let us first formalize the multi-step approach using the set-based representation. We define the refined reachable set resulting from propagations from d points in time t_l, $l \in \{k-d, ..., k\}$ to time step t_{k+1} as

$$\overline{\mathfrak{D}}^d_{k+1} := \Big(\bigcap_{l=k-d}^{k} \mathrm{discr}\left(\underset{t_{k+1}}{\mathrm{reach}}(\overline{\mathfrak{D}}_l, \mathcal{U}, \emptyset) \right) \Big) \setminus \mathrm{discr}(\mathcal{F}(t_{k+1})),$$
$$\overline{\mathfrak{D}}_0 = x_0. \tag{18}$$

Since $\overline{\mathfrak{D}}^0_{k+1}$ is always among the intersected sets in (18), $\overline{\mathfrak{D}}^d_{k+1} \subseteq \mathfrak{D}_{k+1}$ is true for any $d > 0$ showing that multiple propagations obviously provide tighter results. The propagation steps $\mathrm{discr}(\mathrm{reach}_{t_{k+1}}(\overline{\mathfrak{D}}^0_l, \mathcal{U}, \emptyset)))$ can also be computed offline and represented by propagation matrices P^{k+1}_l using the same approach as in Sec. III-A. Thus, we can write the multi-step online propagation from (18) in matrix notation as

$$\overline{r}^d_{k+1} = \Big(\bigwedge_{l=k-d}^{k} P^{k+1}_l \overline{r}_l \Big) \wedge \neg o_{k+1}\,. \tag{19}$$

Since (19) can be implemented efficiently, the runtime is mainly dominated by the discretization of obstacles; even multiple propagations do not impact run time considerably, as shown in Sec. IV.

D. Exclusion of Inevitable Collision States

By utilizing the graph, we can efficiently exclude ICS from the previously computed drivable area. Even though the principle was formulated before in [18], it becomes especially effective when combined with our approach that uses a fine discretization of the whole drivable area. When no path in the graph $\mathcal{G}$ from a cell $\mathcal{C}^{(i)}_k$ to a cell at the final time step exists, all states in $\mathcal{C}^{(i)}_k$ eventually lead to a collision. Thus, we exclude these cells from the drivable area by iterating backward from the final set $\overline{r}^d_h$ and deleting nodes with no reachable set at a preceding time step. Using the propagation matrices and the multi-step propagation as in (19), this is computed at each time step as

$$\underline{r}^d_{k-1} = \Big(\bigwedge_{l=k}^{k+d} {P^l_{k-1}}^T \underline{r}_l \Big) \wedge \neg o_{k-1}, \quad \underline{r}_h = \overline{r}^d_h, \tag{20}$$

where the transpose of the propagation matrix follows from its definition in (12). The complete forward-backward algorithm of the online reachability analysis is summarized in Algorithm 2.

Algorithm 2 Online Reachability Analysis

Require: graph represented by propagation matrices P^{k+1}_k, forbidden set $\mathcal{F}(t)$, number of time steps h, and number d of considered time steps for propagation

1: **for** $k = 0$ to h **do**
2: $x(t_k;,0,x_0) \leftarrow$ HOMOGENEOUSSOLUTION(x_0)
3: $o_{k+1} \leftarrow$ OCCUPANCYGRID$(\mathcal{F}_{k+1}, \mathcal{C}_{k+1}, x(t_k;,0,x_0))$ ▷ see (15)
4: $\overline{r}^d_{k+1} \leftarrow$ PROPAGATE$(P^{k+1}_k, \{r_{k-d}, ..., r_k\}, o_{k+1})$ ▷ see (19)
5: **end for**
6: $\underline{r}_h \leftarrow \overline{r}^d_h$
7: **for** $k = h$ to 1 **do**
8: $\underline{r}^d_{k-1} \leftarrow$ EXCULDEICS$(P^k_{k-1}, \underline{r}^d_k)$ ▷ see (20)
9: **end for**
10: **return** $\{\underline{r}^d_k \mid k \in \{0, \ldots, h\}\}$

IV. Evaluation

We evaluate our approach using recorded traffic from the CommonRoad benchmark suite[1] [30] and use [31] to create obstacles representing road boundaries to detect leaving the road. Further, we compare our results with those of [18] using identical parameters for all scenarios as listed in Table I. Both methods are implemented in Python using C++ for computationally expensive operations. The computation times were measured on a laptop with an Intel i7-8650U 1.90 GHz processor and 16 GB of RAM.

[1] https://commonroad.in.tum.de/

TABLE I: Parameters used in the evaluation.

Parameter	Value
maximal acceleration $a_{\max}$	$5.0\ \mathrm{m/s^2}$
cell size dx_ζ, dx_η	$0.5\ \mathrm{m}$
number of time steps d	$\{0, 1, 7\}$
time step Δt	$0.1\mathrm{s}$
radius ρ	$1.25\mathrm{m}$

A. Computation Time

To compare the online computation time with the polytope-based approach in [18], we compute the drivable area for 339 scenarios from the CommonRoad benchmark suite. In this comparison, $d = 7$ time steps are propagated simultaneously. Fig. 4 shows the median and the ranges of the computation times over the number of computed time steps. The proposed algorithm requires $0.037\,\mathrm{s}$ for 34 time steps as the median computation time, compared with $0.170\,\mathrm{s}$ with the polytope-based approach. In particular, for longer time horizons and larger drivable areas, our method outperforms the polytope-based approach. The maximum computation time is also lower with $0.25\,\mathrm{s}$ compared with $0.34\,\mathrm{s}$.

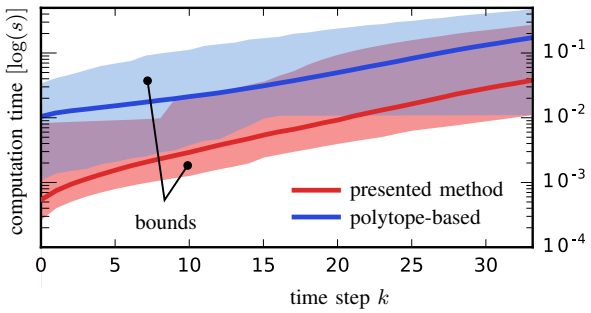
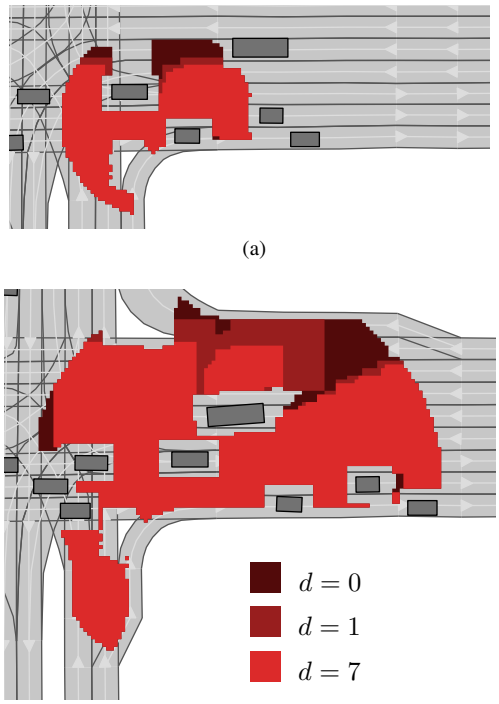


Fig. 4: Median computation times and min/max computation times depending on time steps k compared to polytope-based computation [18].

B. Scenario A

To illustrate the accuracy of our approach, we present results for two out of the considered 339 scenarios. The first scenario is an intersection from the NGSIM Lankershim Dataset[2] which can be found under ID USA_Lanker-1_1_T-1 in the CommonRoad benchmark suite. In Fig. 5, the evolution of the drivable area is depicted at different points in time and for a different number of additionally considered time steps d (see (19)). Thus, it shows that incorporating multiple time steps during the propagation has a noticeable effect even for $d = 1$, which is especially evident in the upper right region that is cut off by other vehicles. Comparing the results in Fig. 6 with those obtained by the algorithm from [18], it suggests that our computed area is slightly less over-approximative, which results from the box constraint for the input set $\mathcal{U}$ over-approximating the friction circle in [18]; in contrast, we directly use the friction circle in (5). Nevertheless, there are regions where our approach is more over-approximative, which indicates that our method over-approximates the velocity in these regions more.



(a)

(b)

Fig. 5: Scenario A: Effect of multiple propagations for different numbers of involved preceding time steps d at (a) $t = 2.4$ s and (b) $t = 3.4$ s.

C. Scenario B

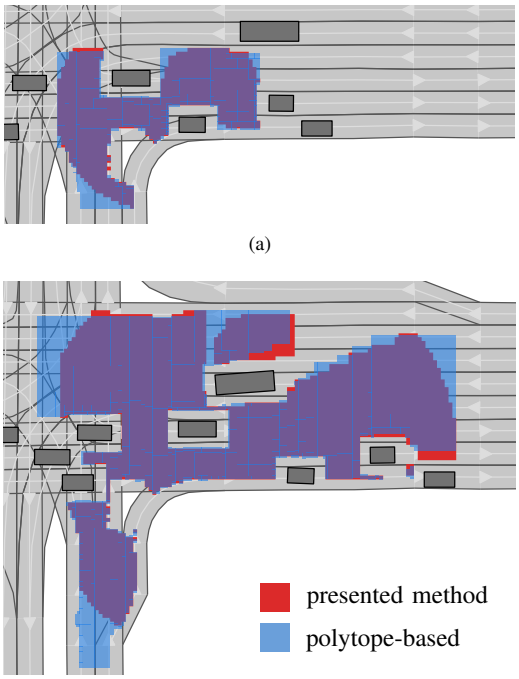
The second scenario is a highway scenario from the US 101 database[3], which can be found under ID USA_US101-27_1_T-1 in CommonRoad. The ego vehicle has an initial velocity of $13.88\,\mathrm{m/s}$. As in Scenario A, the effect of d is shown in Fig. 7 and the comparison to [18] is shown in Fig. 8. At $t = 2.4\,\mathrm{s}$ a smaller over-approximation can be observed due to the constraint of the friction circle that is only considered by our method. In contrast, the drivable area from our approach is larger at $t = 3.4\,\mathrm{s}$ in the upper right region since we tend to over-approximate the velocity.

D. Discussion

Online computation can be divided into discretization (15) and propagation (17). The discretization is the dominating part, which contributes $56\,\%$ of the overall run-time on average. In the worst case, when every obstacle needs to be discretized once every time step due to an intersection with the bounding box of the drivable area (see Sec. III-B.2), the complexity for the discretization is linear with respect to the number of obstacles n_o. In contrast, the computation time for the propagation in (19) does not depend on the number of obstacles, but only linearly on the number of nodes, the number of edges for every node and the number of considered time steps d.

[2]https://www.fhwa.dot.gov/publications/research/operations/07029/

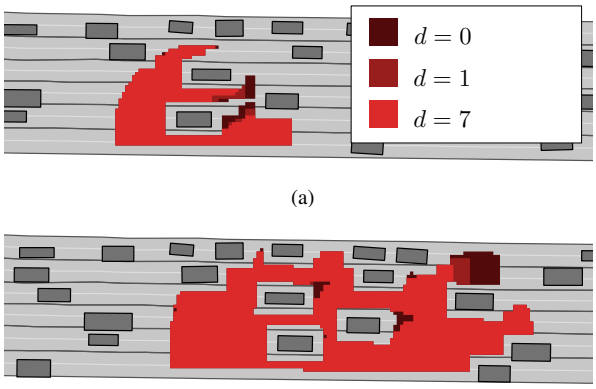
[3]https://www.fhwa.dot.gov/publications/research/operations/07030/



(a)

(b)

Fig. 6: Scenario A: Comparison of the approach with $d = 7$ to results from [18] at (a) $t = 2.4$ s and (b) $t = 3.4$ s.



(a)

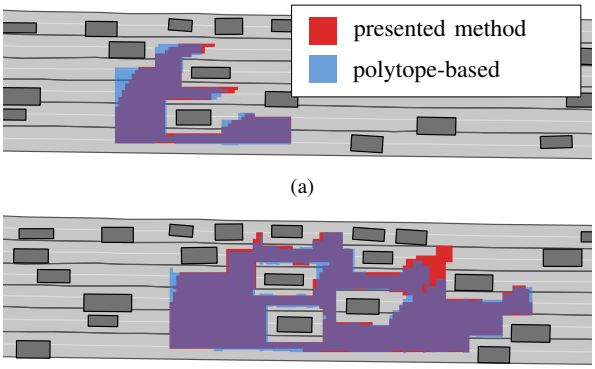
(b)

Fig. 7: Scenario B: Effect of multiple propagations for different numbers of involved preceding time steps d at (a) $t = 2.4$ s and (b) $t = 3.4$ s.

The limiting factor for our approach is the required memory for storing the offline computed matrices P_k^{k+1} in Sec. III-A. However, for 34 time steps the resulting file still has a reasonable size of 173 MB and takes 134 minutes to compute using the tool CORA [28].

V. Conclusions

We present a method for the fast computation of drivable areas considering dynamic obstacles. In hundreds of scenarios, we show that our method results in a faster computation time compared with our previous approach while providing a comparable accuracy due to our multi-step approach. Compared to related work [12]–[14], our method can compute the drivable area more efficiently and is also able to handle more complex traffic situations and road layouts. The resulting graph from our method can be used subsequently by a trajectory planner for extracting a driving corridor efficiently or finding an initial solution with graph-based methods.



(a)

(b)

Fig. 8: Scenario B: Comparison of the approach with $d = 7$ to results from [18] at (a) $t = 2.4$ s and (b) $t = 3.4$ s.

Acknowledgments

The authors gratefully acknowledge financial support by the Central Innovation Programme of the German Federal Government under grant ZF4086007BZ8.

References

[1] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Hybrid Systems: Computation and Control*, 2005, pp. 291–305.

[2] Z. Han and B. H. Krogh, "Reachability analysis of nonlinear systems using trajectory piecewise linearized models," in *Proc. of the American Control Conference*, 2006, pp. 1505–1510.

[3] M. Althoff, O. Stursberg, and M. Buss, "Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization," in *Proc. of the 47th IEEE Conference on Decision and Control*, 2008, pp. 4042–4048.

[4] S. Dixit, U. Montanaro, S. Fallah, M. Dianati, D. Oxtoby, T. Mizutani, and A. Mouzakitis, "Trajectory planning for autonomous high-speed overtaking using MPC with terminal set constraints," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2018, pp. 1061–1068.

[5] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 2352–2358.

[6] M. Chen, J. F. Fisac, S. Sastry, and C. J. Tomlin, "Safe sequential path planning of multi-vehicle systems via double-obstacle Hamilton-Jacobi-Isaacs variational inequality," in *Proc. of the European Control Conference*, 2015, pp. 3304–3309.

[7] S. Manzinger and M. Althoff, "Tactical decision making for cooperative vehicles using reachable sets," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2018.

[8] S. Söntges, M. Koschi, and M. Althoff, "Worst-case analysis of the time-to-react using reachable sets," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1891–1897.

[9] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control*, 2015, pp. 11–20.

[10] K. Margellos and J. Lygeros, "Hamilton-Jacobi formulation for reach-avoid problems with an application to air traffic management," in *Proc. of the American Control Conference*, 2010, pp. 3045–3050.

[11] O. Bokanowski, N. Forcadel, and H. Zidani, "Reachability and minimal times for state constrained nonlinear problems without any controllability assumption," *SIAM Journal on Control and Optimization*, vol. 48, no. 7, pp. 4292–4316, Jan. 2010.

[12] C. Schmidt, F. Oechsle, and W. Branz, "Research on trajectory planning in emergency situations with multiple objects," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2006, pp. 988–992.

[13] S. Kousik, S. Vaskov, M. Johnson-Roberson, and R. Vasudevan, "Safe trajectory synthesis for autonomous driving in unforeseen environments," in *Proc. of the ASME Dynamic Systems and Control Conf.*, Art. No. V001T44A005, 2017.

[14] I. Xausa, R. Baier, O. Bokanowski, and M. Gerdts, "Computation of avoidance regions for driver assistance systems by using a Hamilton-Jacobi approach," Art. No. OCA.2565, 2020.

[15] A. Lawitzky, A. Nicklas, D. Wollherr, and M. Buss, "Determining states of inevitable collision using reachability analysis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4142–4147.

[16] J. Nilsson, J. Fredriksson, and A. C. E. Ödblom, "Verification of collision avoidance systems using reachability analysis," *Proc. of the IFAC World Congress*, pp. 10 676–10 681, 2014.

[17] P. Falcone, M. Ali, and J. Sjoberg, "Predictive threat assessment via reachability analysis and set invariance theory," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1352–1361, 2011.

[18] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1855–1866, 2018.

[19] M. Ono, G. Droge, H. Grip, O. Toupet, C. Scrapper, and A. Rahmani, "Road-following formation control of autonomous ground vehicles," in *Proc. of the IEEE Conference on Decision and Control*, 2015, pp. 4714–4721.

[20] J. Lunze, "A timed discrete-event abstraction of continuous-variable systems," *International Journal of Control*, vol. 72, no. 13, pp. 1147–1164, 1999.

[21] M. Zamani, G. Pola, M. Mazo, and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1804–1809, 2012.

[22] G. Frehse, "PHAVer: Algorithmic verification of hybrid systems past HyTech." in *Hybrid Systems: Computation and Control*, 2005, pp. 258–273.

[23] A. Platzer and E. M. E. Clarke, "Formal verification of curved flight collision avoidance maneuvers: A case study," in *Proc. of the 16th International Symposium on Formal Methods*, 2009, pp. 547–562.

[24] X. Chen, S. Sankaranarayanan, and E. Abraham, "Flow* 1.2: More effective to play with hybrid systems," in *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, vol. 34, 2015, pp. 152–159.

[25] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *Proc. of the 23rd International Conference on Computer Aided Verification*, 2011, pp. 379–395.

[26] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2E2: A verification tool for stateflow models," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2015, pp. 68–82.

[27] S. Bogomolov, M. Forets, G. Frehse, F. Viry, A. Podelski, and C. Schilling, "Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices," in *Proc. of the 21st International Conference on Hybrid Systems: Computation and Control*, 2018, pp. 41–50.

[28] M. Althoff, "An introduction to CORA 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015, p. 120151.

[29] J.-C. Latombe, *Robot Motion Planning*. Norwell: Kluwer Academic Publishers, 1991.

[30] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.

[31] A. Zhu, S. Manzinger, and M. Althoff, "Evaluating Location Compliance Approaches for Automated Road Vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 642–649.

Chapter 4

Falsification Using Concrete Scenarios Synthesized from Abstractly Specified Scenarios

In this chapter, we propose methods for synthesizing concrete scenarios from abstract scenario specifications, which we combine with falsification methods that provide a solution to Problem 2. This chapter is structured into two parts: first, we present a format to represent scenarios abstractly and enable synthesizing concrete scenarios. In the second part, we propose a falsification algorithm that utilizes MCTS and the scenario representation format and a scenario synthesis based on reachability analysis.

4.1 Synthesizing Traffic Scenarios from Formal Specifications for Testing Automated Vehicles

Efficient methods for synthesizing concrete scenarios from abstract scenario specifications are an important step stone toward the automated and scalable testing and falsification of autonomous vehicles when using a scenario-based approach. Especially for complex specifications that involve multiple traffic participants, finding a concrete scenario that realizes the specification can be challenging. Furthermore, for existing scenario representations and concretization approaches it is not always possible to assess whether a scenario concretization is feasible.

In this work, we first propose a scenario format that enables the spatiotemporal specification for the behavior of all traffic participants. The scenario is divided into successive scenes, where for each scene the spatial relations between traffic participants and their associated lanes are defined on a semantic level. Based on this scenario format, we propose a constraint-based representation that we integrate with a mixed-integer optimization problem to synthesize concrete trajectories of the traffic participants. We show how the behavior of traffic participants at intersections, in lane-changing or overtaking scenarios can be coordinated jointly. Additionally, the mixed-integer representation can be used to check the executability of an abstract scenario, i.e., whether there exists any concrete scenario that complies with the specification.

We evaluate our method using specifications for merging lanes and a more complex intersection scenario with multiple traffic participants each. To assess how reliable our method can synthesize concrete scenarios, we develop an algorithm to generate more than 600 variations of abstract scenarios and measure the rate of feasible scenarios and the required computation times. The evaluation demonstrates that our approach can synthesize for the majority of scenarios with computation times between 0.21 and 2.29 s on average. The constraint-based representation provides the basis for our subsequent work on falsification.

Contributions M. K. initiated the idea for the constraint-based scenario representation and the formulation of the scenario concretization as a mixed-integer optimization problem. He implemented the method and conducted the evaluation. M. A. led the research project and provided feedback to improve the manuscript.

Conference Article In this thesis, the accepted version submitted by the author is reprinted. The edited version is available under https://doi.org/10.1109/IV47402.2020.9304617.

IEEE Requesting permission to reuse content from an IEEE publication

Synthesizing Traffic Scenarios from Formal Specifications for Testing Automated Vehicles

Conference Proceedings: 2020 IEEE Intelligent [::Vehicles::] Symposium (IV)

Author: Moritz Klischat; Matthias Althoff

Publisher: IEEE

Date: 19 Oct.-13 Nov. 2020

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

Synthesizing Traffic Scenarios from Formal Specifications for Testing Automated Vehicles

Moritz Klischat and Matthias Althoff

Abstract— **Virtual testing plays an important role in the validation and verification of automated vehicles. State-of-the-art approaches first generate a huge amount of test scenarios through simulations or test drives, which are later filtered to obtain relevant scenarios for a given set of specifications. However, only few works exist on synthesizing scenarios directly from specifications. In this work, we present an optimization-based approach to synthesize scenarios only from formal specifications and a given map. To concretize the specifications, we formulate predicates, which are subsequently converted to a mixed-integer quadratic optimization problem. We demonstrate how our method can generate scenarios for maps featuring merging lanes and intersections given a variety of specifications.**

I. Introduction

Proving the safety of automated vehicles is still a major challenge due to the variety of situations that can be possibly encountered in the real world. To cope with this variety, virtual testing is essential in the development phase. Especially in industry, virtual testing is oftentimes still based on data recorded from real test drives or variations of it[1]. Not only is this expensive and time-consuming, but also the availability of large datasets to suppliers or public research institutions is limited.

In scenario-based testing, automated vehicles are virtually subjected to short sequences of traffic data, i.e., *scenarios*, which are representative of real-world traffic. For the verification of automated vehicles, test engineers are additionally interested in 1) using scenarios to test the software against selected requirements and 2) a large variety of traffic scenarios where some are only rarely found in datasets recorded from real test drives.

Requirements can include formal specifications of a scenario, i.e., the behavior of surrounding vehicles. These specifications can also be utilized to set a large variety of scenarios in order to create a diverse test suite. In this paper, we present an optimization-based algorithm that synthesizes concrete scenarios that fulfill a given formal specification.

A. Related Work

A commonly-used method to generate test scenarios is the parametrization of scenarios and combinations of all possible values within defined parameter ranges[1]. In particular, for complex scenarios with many parameters, the exploding number of parameter combinations quickly results in an unmanageable computational effort. At the same time, many parameter combinations might be unrealizable.

Approaches focusing on parameterizing and modifying trajectories from real traffic data are presented, e.g., in [1], [2]. The derived variations of a given scenario still resemble the high-level characteristics of the original data. While these approaches consider only one scenario at a time, the stochastic properties of the traffic behavior from an entire database of scenarios can be considered when sampling new scenarios from a Bayesian network whose parameters are learned from that database [3]. This method has been further extended from highways to complex intersections in [4]. With the application of importance sampling, scenarios with interesting, rarely occurring behavior can be generated more efficiently [5], [6].

In verification, software components are typically tested against their functional requirements in scenarios defined by formal specifications. Works on falsification aim at falsifying planning algorithms of automated vehicles against a given logic formula, e.g., formulated in signal temporal logic (STL) [7], [8]. However, these works do not answer the question of how to ensure that the behavior of surrounding vehicles conforms to assumptions the test specification is based on.

While the above approaches for scenario generation can be used to easily generate a large number of scenarios, they cannot explicitly consider specifications for the scenarios. Instead, classification techniques, e.g. [9], [10], would have to be applied subsequently to find scenarios that conform with a desired specification. This process is proposed in [11], [12]. However, these data-driven approaches require large amounts of data to find scenarios that conform with a specification. In particular, for rarely occurring specifications, the required amount of scenarios, and correspondingly the computational effort, increases disproportionately [11]. A more efficient approach is to generate scenarios directly from specifications. The resulting amount of scenarios scales only linearly with respect to the number of specifications.

In [13], specifications of scenarios are first derived from police reports on crashes using natural language processing from which waypoints connected by trajectory planners are generated. While this is demonstrated for scenarios with two vehicles, it remains unclear how the motion of more vehicles could be coordinated reliably. In the context of search-based scenario generation [14], the implicit conformance with a specification can be obtained through dedicated cost functions [15]. With these functions, the search is directed to regions in the space of scenario parameters where the

All authors are with the Technische Universität München, Fakultät für Informatik, Lehrstuhl für Robotik, Künstliche Intelligenz und Echtzeitsysteme, Boltzmannstraße 3, 85748, Garching, Germany. {moritz.klischat, althoff}@in.tum.de

[1]https://waymo.com/safety

specification is fulfilled.

Another representation of specifications is given by ontologies, which can also be generated automatically [16]. Highway scenario descriptions conforming with such ontologies are generated in [17] and combinatorial test generation from ontologies focusing on road infrastructure is proposed in [18].

A formal representation of specifications is provided by temporal logic. In [19], [20], control problems for linear systems subjected to linear temporal logic (LTL) specifications are solved using mixed-integer formulations.

Lately, two scenario description languages and respective scenario generators have been developed, which focus on a realistic visualization of the environment for vision-based algorithms [21], [22]. Another high-level description of traffic scenarios is presented in [9]; so-called traffic patterns describe the traffic flow at intersections, and an algorithm to match scenarios to the patterns is presented.

For the formalization of specifications with a focus on motion planning, a specification language based on constraints is proposed [23]. Valid parameter ranges that satisfy the specifications can be obtained using SAT solvers [24].

In summary, existing works do not provide methods to efficiently generate concrete scenarios from complex specifications in terms of the road network or the coordination of motion of surrounding vehicles.

B. Overview and Contributions

In this work, we present an optimization-based method to synthesize traffic scenarios that conform to a scenario specification. Our specification language is formally defined in Section II: We extend a high-level description similar to [9] by a detailed formal description of spatio-temporal relations of vehicles. The main part of our work is subsequently presented in Section III, where we use for the first time a framework based on mixed-integer quadratic optimization (MIQP) that enables the generation of concrete trajectories satisfying a scenario specification. To consider the specification in the optimization, we formulate them as mixed-integer convex constraints. In Section IV, we demonstrate how our approach can be used to efficiently generate diverse traffic scenarios by applying it to a large set of specifications. The following are the main contributions of our work:

- We present a general scenario specification suited for complex road network topologies.
- Our approach is the first based on MIQP for synthesizing concrete traffic scenarios that explicitly consider formal specifications.
- By including a feasibility check for specifications, our scenarios are guaranteed to be executable.

The advantage of our optimization-based scenario synthesis is that it does not rely on recorded or simulated data and we can generate scenarios based on a road network and an abstract specification only. Furthermore, it opens the possibility of incorporating additional objectives, such as criticality measures, into the cost function to obtain scenarios with desired properties.

II. Scenario Specification

In this work, we use discretized time $t_k = k\Delta t$ with time steps Δt and k as the time index. To simplify the notation, we denote the time-step sequence $(\underline{k}, \underline{k}+1, ..., \overline{k})$ by $[\![\underline{k}, \overline{k}]\!]$. We generate traffic scenarios, which are defined by vehicles V_i following trajectories $x_i(k) \in \mathbb{R}^n, k \in [\![0, h]\!]$, where $i \in \{1, \dots, n_{\text{veh}}\}$ refers to the index of the vehicle, and $h \in \mathbb{N}$ represents the considered time horizon.

Subsequently, we introduce a two-level scenario specification consisting of route patterns and scene sequences. The route patterns are defined in Section II-A and scene sequences in Section II-C.

A. Route Patterns

Before introducing the route patterns, let us define road networks consisting of lanelets [25].

Definition 1 (Lanelet): A lanelet L_{id} with an identifier id is composed of right and left borders defined by polylines and attributes describing its spatial relations to other lanelets: *successors*, *predecessors*, *adjacent_right*, and *adjacent_left*.□

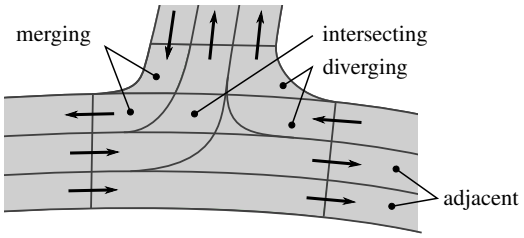


Fig. 1: Example of relations between lanelets.

A lanelet is said to be *adjacent_right* to a lanelet L, when its left lanelet border intersects with L across its entire length. Compared to the original work [25], we additionally use the relations *merging*, *diverging*, and *intersecting*. *Merging (diverging)* lanelets share the same successors (predecessors). *Intersecting* lanelets comprise all pairs of lanelets where the borders intersect and no adjacency or merging relation can be found. Fig. 1 shows an example of these relations.

To further structure the lanelet network, we introduce lanelet sections C_{i_c} combining lanelets that are coupled laterally through the relations *adjacent_left* and *adjacent_right*, as illustrated in Fig. 2. The ID of such a section is denoted by $i_\text{c} \in \mathbb{N}$. For convenience, we denote the IDs of the lanelets by tuples $id = (i_\text{c}, i_l)$, where $i_l \in \mathbb{N}$ enumerates the lanelets of a section in the lateral direction from right to left. Using lanelet sections, we define routes in the lanelet network.

Definition 2 (Route): A route R_κ with route index κ is defined as a tuple of connected lanelet sections. □

For instance, the route of vehicle V_i in Fig. 2 is given by $R_0 = (C_0, C_1)$. The trajectory of each vehicle V_i can be mapped to a route. For this mapping, we introduce the operator $\text{route}(V_i)$, which maps a vehicle to a route R_κ.

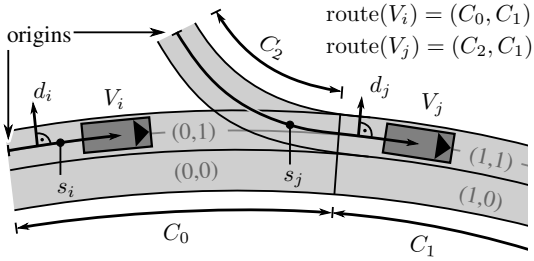
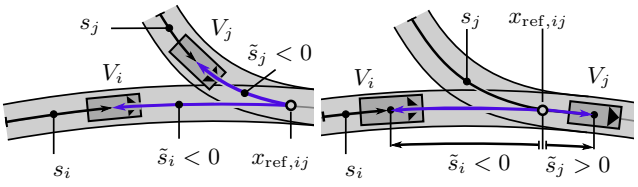


Fig. 2: Example of lanelet sections and lane-based coordinate systems..



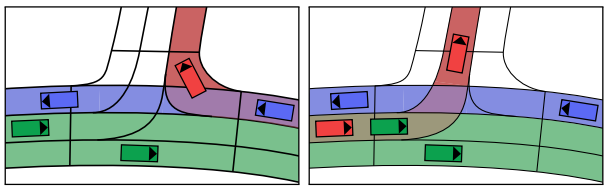
(a) Before merging. (b) After merging.

Fig. 4: Coupling of lane-based coordinate systems of two vehicles V_i, V_j using $x_{\mathrm{ref},ij}$ for computing the distance $\mathrm{dist}(V_i, V_j) = \tilde{s}_j - \tilde{s}_i$.

Definition 3 (Route Pattern): A route pattern of a scenario is the union of the routes of all vehicles in the scenario:

$$T_{\mathrm{route}} = \bigcup_{i=1}^{n_{\mathrm{veh}}} \mathrm{route}(V_i). \qquad \square$$

With these patterns, a scenario can be described on an abstract level. Fig. 3 shows an example of two different route patterns for the same intersection.



(a) (b)

Fig. 3: Examples of two route patterns at an intersection, vehicles are colored based on their assigned routes.

B. Lane-based Coordinate Systems

Henceforth, we consider trajectories in lane-based coordinate systems that are aligned to a reference path. The reference path is represented by a polyline and constructed for each route $R_\kappa \in T_{\mathrm{route}}$ by concatenating the center lines of consecutive lanelets from each section of the route. Since every vehicle is assigned to a route, every vehicle is also assigned to a lane-based coordinate system. The longitudinal state of a vehicle V_i in the coordinate system of $\mathrm{route}(V_i)$ is given by $x_{\mathrm{s},i} = [s_i, \dot{s}_i, \ddot{s}_i]$, where s_i is the longitudinal position, and the lateral state is given by $x_{\mathrm{d},i} = [d_i, \dot{d}_i, \ddot{d}_i]$ with d_i being the lateral position (see Fig. 2). We define a projection operator $\mathrm{lon}_i(x)$ to project a Cartesian coordinate to a longitudinal position of $\mathrm{route}(V_i)$.

For two vehicles V_i and V_j, our specification can involve the longitudinal distance between these vehicles from possibly different coordinate systems. As in our previous work [26], we couple the coordinates s_i and s_j of a pair of vehicles by computing a common reference point $x_{\mathrm{ref},ij}$ at the first intersection of their reference paths. We introduce auxiliary coordinates $\tilde{s}_i = s_i - \mathrm{lon}_i(x_{\mathrm{ref},ij})$ and $\tilde{s}_j = s_j - \mathrm{lon}_j(x_{\mathrm{ref},ij})$ relative to the reference point. Using these coordinates, we define the operator $\mathrm{dist}(V_i, V_j) \coloneqq \tilde{s}_j - \tilde{s}_i$ for computing the longitudinal distance between V_i and V_j. As shown in Fig. 4, this distance is also defined for vehicles before their lanelets merge.

C. Scene Specification

To further specify the behavior of vehicles, we introduce predicates. We focus on essential predicates that can already express a large variety of traffic scenarios. Nevertheless, any other predicate that can be formulated as a mixed-integer convex constraint in the MIQP, as in Section III, can be modeled as well. The predicates are introduced as follows:

Definition 4 (onLanelet): Let a set of lanelets $\mathcal{L}(i_c, [\![\underline{i}_l, \overline{i}_l]\!])$ be defined by the section ID i_c and a sequence of lateral IDs $[\![\underline{i}_l, \overline{i}_l]\!]$. Computing the longitudinal bounds of all lanelets in $\mathcal{L}(i_c, [\![\underline{i}_l, \overline{i}_l]\!])$ in the coordinate system of a vehicle V_i yields the interval $[\underline{s}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}, \overline{s}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}]$ and the lateral bounds at a longitudinal position s_i within that interval are determined as $[\underline{d}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}(s_i), \overline{d}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}(s_i)]$. Then the predicate `onLanelet` is defined as

$$\begin{aligned}\texttt{onLanelet}(V_i, i_c, [\![\underline{i}_l, \overline{i}_l]\!]) \iff \\ \underline{s}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])} \le s_i \le \overline{s}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])} \\ \underline{d}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}(s_i) \le d_i \le \overline{d}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}(s_i). \square\end{aligned}$$

Note, that our definition of $\texttt{onLanelet}(V_i, i_c, [\![\underline{i}_l, \overline{i}_l]\!])$ considers the center of the vehicle and not the whole occupancy of the vehicle. By specifying multiple lanelets, we make lane changes possible.

Definition 5 (isBehind): If V_i and V_j are specified to move on the same lanelet or on merging, diverging, or succeeding lanelets, we define that vehicle V_i is behind vehicle V_j with a safety margin $r \in \mathbb{R}^+$ through the predicate

$$\texttt{isBehind}(V_i, V_j) \iff \mathrm{dist}(V_i, V_j) > r. \qquad (1)$$

$\square$

For a vehicle, which is specified to move on a lanelet L_i intersecting with another lanelet L_j, we use additional predicates to specify the crossing behavior. For formulating these predicates, the conflicting area of both vehicles needs to be defined first. To maximize the flexibility for generating scenarios, we aim at computing a small conflicting area. Hence, we use the intersecting area of both lanelets to avoid collisions. When traffic rules should be considered, one could alternatively use stop lines at intersections.

To determine the intersecting area of two lanelets L_i and L_j in lane-based coordinates, we first compute the intersecting points $\xi_z(L_i, L_j) \in \mathbb{R}^2$ with $z \in \{1, \dots, 4\}$ of the lanelet borders through a sweep-line algorithm [27]. After computing the longitudinal coordinates $s_{\xi,z}(L_i, L_j)$ of

each point $\xi_z(L_i, L_j)$, we can determine the longitudinal interval $[\underline{s}^{\text{ca}}_{i,j}, \overline{s}^{\text{ca}}_{i,j}]$ that bounds all $s_{\xi,z}(L_i, L_j)$ as illustrated in Fig. 5. Using this interval, we divide the lanelet L_i into three sections and introduce the corresponding predicates below. These predicates are also illustrated in Fig. 5.

Definition 6 (Conflict Area Predicates): The position of a vehicle V_i on a lanelet intersecting with the lanelet of another vehicle V_j is evaluated by the predicates

$$\begin{aligned}\texttt{beforeCA}(V_i, V_j) &\iff s_i < \underline{s}^{\text{ca}}_{i,j} - r,\\ \texttt{inCA}(V_i, V_j) &\iff \underline{s}^{\text{ca}}_{i,j} - r \leq s_i \leq \overline{s}^{\text{ca}}_{i,j} + r,\\ \texttt{behindCA}(V_i, V_j) &\iff s_i > \overline{s}^{\text{ca}}_{i,j} + r\end{aligned}$$

using a safety margin $r > l_i/2$ with l_i being the length of V_i. □

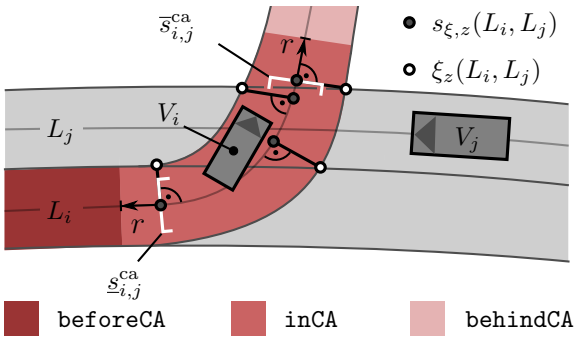


Fig. 5: Construction of longitudinal bounds $[\underline{s}^{\text{ca}}_{i,j}, \overline{s}^{\text{ca}}_{i,j}]$ of the conflict area for vehicle V_i crossing the lanelet of V_j. Furthermore, the intervals corresponding to each intersection predicate of V_i are shown.

To further structure the specification, we divide the set of predicates into scenes as illustrated for an example shown in Fig. 6.

Definition 7 (Scene Sequence): We define a scene S_l by the tuple $(\mathcal{P}_l, \underline{k}_l)$, which encodes the set $\mathcal{P}_l$ of predicates that hold true starting at the switching time $\underline{k}_l \in [\![0, h]\!]$ until the switching time of the subsequent scene S_{l+1}. Thus the complete scenario is specified by the sequence of n_{sc} scenes

$$T_{\text{sc}} = (S_0, \ldots, S_l, \ldots, S_{n_{\text{sc}}}).$$ □

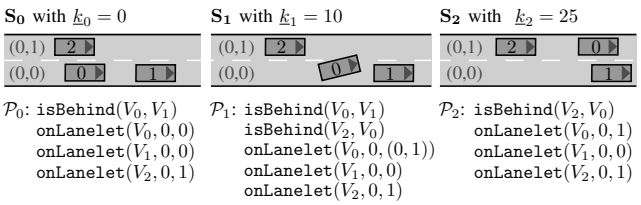


Fig. 6: Example of a lane-change maneuver defined by three scenes.

Instead of specifying the exact switching times, it is more convenient to set lower and upper bounds for the duration δ_l of each scene, i.e., $h_{\min} \leq \delta_l \leq h_{\max}$, $h_{\max} \in \mathbb{N}$, $\forall l \in [\![0, n_{\text{sc}}]\!]$. This also reduces the number of specifications, because scenarios with the same predicates $\mathcal{P}_l$, but different switching times $\underline{x}_l$ can be described by the same specification. The exact duration is determined in the scenario synthesis presented in the next section.

Our definition of a *scene* can be considered as a formalized version of the frequently-used definition in [28]. When creating the specification, collision-free scenarios can be specified by respecting simple rules. For instance, collisions on intersections are avoided when the specification satisfies

$$\begin{aligned}\forall i, j, l\colon i \neq j \wedge \neg(\texttt{inCA}(V_i, V_j) \in \mathcal{P}_l \wedge \texttt{inCA}(V_j, V_i) \in \mathcal{P}_l)\\ \vee \neg(\texttt{inCA}(V_i, V_j) \wedge \texttt{inCA}(V_j, V_i)).\end{aligned} \quad (2)$$

Test cases for automated vehicles can be formulated using our specification by including an ego vehicle in the specification and finally deleting it from the synthesized scenario.

III. Synthesis of Trajectories through Optimization

To synthesize concrete trajectories that comply with a given scenario template, we formulate a mixed logical dynamical (MLD) system and generate a combined MIQP whose solution yields the trajectories for all vehicles. As commonly done in motion planning for road vehicles, we solve the longitudinal and lateral planning problems separately [29].

Similar approaches are proposed for the control of general MLD systems [30], cooperative planning of trajectories [31]–[33] or for trajectory planning of lane changes for single vehicles [34]. Unlike in previous work, our optimization problem combines planning of longitudinal and lateral motion for multiple vehicles while handling intersections.

A. Representation of Switching Times

Initially, the switching times $\underline{k}_l$ of each scene are unknown and need to be determined as part of the optimization problem. To facilitate formulating the optimization problem, we encode each $\underline{k}_l$ through a binary vector $\beta_l \in \{0,1\}^h$, $\forall l \in [\![0, n_{\text{sc}}]\!]$ that switches at $\underline{k}_l$

$$\beta_l(k) = \begin{cases} 0, & k < \underline{k}_l \\ 1, & k \geq \underline{k}_l \end{cases}.$$

The temporal order of scenes is ensured through the linear constraints

$$\begin{aligned}&\beta_0(0) = 1 \\ \forall k \in [\![1, h]\!], \forall l \in [\![0, n_{\text{sc}} - 1]\!]\colon\quad &\beta_l(k) \leq \beta_{l+1}(k).\end{aligned} \quad (3)$$

For constraining the duration δ_l of each scene, we need to determine whether a scene is active, i.e., whether $\underline{k}_l \leq k < \underline{k}_{l+1}$. For convenience, we denote this by auxiliary binary variables $\alpha_l \in \{0,1\}^h$ (see Fig. 7 for an example):

$$\forall k \in [\![0, h]\!]\colon \alpha_l(k) = \begin{cases} \beta_l(k) - \beta_{l+1}(k), & 0 \leq l < n_{\text{sc}} \\ \beta_l(k), & l = n_{\text{sc}} \end{cases}.$$

Hence, the durations are bounded through

$$\forall k \in [\![0, h]\!]\colon \quad h_{\min} \leq \sum_{k=0}^{h} \alpha_l(k) \leq h_{\max}. \quad (4)$$

	$\underline{k}_0 = 0$		$\underline{k}_1 = 2$			$\underline{k}_2 = 5$	
$\beta_0(k)$	1	1	1	1	1	1	1
$\beta_1(k)$	0	0	1	1	1	1	1
$\beta_2(k)$	0	0	0	0	0	1	1
$\alpha_0(k)$	1	1	0	0	0	0	0
$\alpha_1(k)$	0	0	1	1	1	0	0
$\alpha_2(k)$	0	0	0	0	0	1	1

Fig. 7: Example for binary variables $\beta_l(k)$ and $\alpha_l(k)$ with three scenes.

B. System Dynamics

We now define the linear system dynamics for the longitudinal (denoted by subset s) and lateral motions (denoted by subset d) of all vehicles using the state matrices $A_s, A_d \in \mathbb{R}^{p,p}$ and the input matrices $B_s, B_d \in \mathbb{R}^{p,q}$ by

$$x_\mathrm{s}(k+1) = A_\mathrm{s} x_\mathrm{s}(k) + B_s u_\mathrm{s}(k). \tag{5}$$

The lateral motion is defined by

$$x_\mathrm{d}(k+1) = A_\mathrm{d} x_\mathrm{d}(k) + B_d u_\mathrm{d}(k). \tag{6}$$

The state vector x_s comprises the concatenated states of all vehicles $x_\mathrm{s} = [x_{\mathrm{s},1}^T, \dots, x_{\mathrm{s},n_\mathrm{veh}}^T]^T$ and the input vector is given by the jerk of all vehicles to obtain continuous acceleration: $u_\mathrm{s} = [\dddot{s}_1, \dots, \dddot{s}_{n_\mathrm{veh}}]^T$ and $u_\mathrm{d} = [\dddot{d}_1, \dots, \dddot{d}_{n_\mathrm{veh}}]^T$. In combination with the binary variables $\beta_l(k)$ we obtain an MLD system.

C. Converting Predicates to Mixed-Integer Constraints

To consider the specification T_sc in the MIQP motion planning problem, it needs to be written as mixed-integer linear inequality constraints. The formulation of such constraints from predicates for temporal logic has been shown, e.g., for STL [35], [36].

Since our predicates are intentionally formulated as linear inequality constraints, see Definitions 4 to 6, we make use of the big-M method [37] to only activate them during their corresponding scene, i.e., when $\alpha_l(k) = 1$. In this well-known method for mixed-integer programming, a term involving a vector $M \in \mathbb{R}_+^m$, which has sufficiently large values and is of correct dimensions, is added to a linear inequality constraint to control its activation. In our case, adding the binary term $M(1-\alpha_l(k))$ to a constraint as in (7), ensures that a constraint is always fulfilled if $\alpha_l(k) = 0$.

We convert each predicate of each scene S_l, $l \in \{0, \dots, n_\mathrm{sc}\}$ separately and finally combine all constraints to two inequalities in the required big-M form

$$\begin{aligned} \underline{g}_l(x_\mathrm{s}, k) &> \underline{c}_l(k) - M(1-\alpha_l(k)) \\ \overline{g}_l(x_\mathrm{s}, k) &< \overline{c}_l(k) + M(1-\alpha_l(k)) \end{aligned} \tag{7}$$

for lower and upper bounds $\underline{c}_l(k), \overline{c}_l(k) \in \mathbb{R}^{n_\mathrm{pred}}$ using linear functions $\underline{g}_l(x_\mathrm{s}, k)$, $\overline{g}_l(x_\mathrm{s}, k)$.

To give an example, we can directly write (1) of $\texttt{isBehind}(V_i, V_j)$ as a constraint using the big-M method:

$$\underbrace{\tilde{s}_j(k) - \tilde{s}_i(k)}_{\underline{g}_l(x_\mathrm{s},k)} > \underbrace{r}_{\underline{c}_l(k)} - M(1-\alpha_l(k)). \tag{8}$$

D. Longitudinal Optimization Problem

We solve an optimization problem for a user-defined convex cost function $J_\mathrm{s}(x_s, u_s, w)$, which can incorporate terms for desired properties, such as efficiency or criticality. An example is given later for the evaluation in Section IV. By introducing weights $w \in \mathbb{R}^\rho$ for selected terms, we can utilize the cost functions for parameterizing the scenarios in order to obtain variations of the scenario. The complete optimization problem is given by

$$\underset{x_\mathrm{s}(0), u_\mathrm{s}}{\arg\min} \quad \sum_{k=0}^{h} J_\mathrm{s}(x_\mathrm{s}(k), u_\mathrm{s}(k), w) \tag{9}$$

subjected to dynamic constraints, $\forall k \in [\![0, h]\!]$:

$$\begin{aligned} x_\mathrm{s}(k+1) &= A_\mathrm{s} x_\mathrm{s}(k) + B_\mathrm{s} u_\mathrm{s}(k) \\ u_{\mathrm{s},\min} &\le u_\mathrm{s}(k) \le u_{\mathrm{s},\max} \\ x_{\mathrm{s},\min} &\le x_\mathrm{s}(k) \le x_{\mathrm{s},\max}, \end{aligned} \tag{10}$$

predicate constraints, $\forall l \in \{0, \dots, n_{st}\}$, $\forall k \in [\![0, h]\!]$:

$$\begin{aligned} \underline{g}_l(x_\mathrm{s}, k) &< \underline{c}(k) + M(1-\alpha_l(k)) \\ \overline{g}_l(x_\mathrm{s}, k) &> \overline{c}(k) - M(1-\alpha_l(k)), \end{aligned} \tag{11}$$

and logical constraints from (3) and (4):

$$\begin{aligned} \beta_0(0) &= 1 \\ \forall k \in [\![0, h]\!], \forall l \in [\![0, n_\mathrm{sc}-1]\!]&: \beta_l(k) \le \beta_{l+1}(k), \\ \forall k \in [\![0, h]\!]&: h_\mathrm{min} \le \sum_{k=0}^{h} \alpha_l(k) \le h_\mathrm{max}. \end{aligned} \tag{12}$$

E. Lateral Motion

After solving the longitudinal motion problem, we can compute the lateral trajectory of each vehicle. Since the switching times can be obtained from the previously computed $\beta_l(k)$, the active predicates at every time k are known. Hence, only a quadratic program without binary variables needs to be solved. At the longitudinal positions $s(k)$, the lateral lanelet bounds $[\underline{d}_L(s(k)), \overline{d}_L(s(k))]$ specified through `onLanelet` are extracted. Furthermore, the lateral reference path $d_\mathrm{ref}(k) \in \mathbb{R}$ is computed. The trajectory is obtained by solving

$$\underset{u_\mathrm{d}(\cdot)}{\arg\min} \quad \sum_{k=0}^{h} J_\mathrm{d}(x_\mathrm{d}(k), u_\mathrm{d}(k), d_\mathrm{ref}(k), w) \tag{13}$$

subjected to dynamic constraints, $\forall k \in [\![0, h]\!]$:

$$\begin{aligned} x_\mathrm{d}(k+1) &= A_\mathrm{d} x_\mathrm{d}(k) + B_\mathrm{d} u_\mathrm{d}(k) \\ u_{\mathrm{d},\min}(k) &\le u(k) \le u_{\mathrm{d},\max}(k) \\ x_{\mathrm{d},\min}(k) &\le x_d(k) \le x_{\mathrm{d},\max}(k) \end{aligned}$$

and lane constraints, $\forall k \in [\![0, h]\!]$:

$$\underline{d}_L(s(k)) \le x_d(k) \le \overline{d}_L(s(k)).$$

F. Feasibility Checking

Formulating the scenario synthesis as an MIQP problem enables us to check the satisfiability of a specification by the given system dynamics through checking the feasibility of the MIQP problem. For more insights into the cause of a possible infeasibility, the feasibility check can be performed in two steps:

1) The logical checking of the specifications T_{sc} detects a contradiction in the specifications in each scene: We introduce the feasible set $\mathcal{D}_{\mathrm{pred}} \subset \mathbb{R}^{\sum_l n_{\mathrm{pred},l}} \times \{0,1\}^{h n_{\mathrm{sc}}}$, denoting the mixed-integer set fulfilling predicate constraints (11) and binary constraints (12) of the longitudinal problem (9). An empty set $\mathcal{D}_{\mathrm{pred}}$ implies the contradiction of at least two specifications.
2) The satisfiability of the specification by the given system dynamics can be checked by additionally considering the dynamic constraints in (10) for the MIQP feasibility check. This problem can be solved by a branch and bound algorithm and the decidability of this problem has been proven already in [19] for general MLD systems.

IV. Evaluation

In this section, we demonstrate our approach by means of two maps: Map A features two merging lanes on a highway and map B shows an urban T-intersection. To test our approach with a large variety of specifications, we generate them through an exhaustive search, as described in Section IV-A. For the system matrices A_{s} and A_{d}, we use triple integrators for each vehicle and as cost functions we choose $J_{\mathrm{s}}(x_{\mathrm{s}}(k), u_{\mathrm{s}}(k), w) = x_{\mathrm{s}}(k)^T Q_{\mathrm{s}} x_{\mathrm{s}}(k) + u_{\mathrm{s}}(k)^T R_{\mathrm{s}} u_{\mathrm{s}}(k)$ and $J_{\mathrm{d}}(x_{\mathrm{d}}(k), u_{\mathrm{d}}(k), w) = x_{\mathrm{d}}(k)^T Q_{\mathrm{d}} x_{\mathrm{d}}(k) + u_{\mathrm{d}}(k)^T R_{\mathrm{d}} u_{\mathrm{d}}(k)$ to obtain efficient motions. Table I lists the chosen parameters of our algorithm.

Our code is written in Python, and the optimization problems are solved using the solver Gurobi[2]. The computation times are measured on a system with an Intel i7-8650U 1.90 GHz processor. We uploaded all generated scenarios to our website[3] with IDs ZAM_Zip and ZAM_Tjunction. A selection of scenarios can also be found in the video attachment at https://mediatum.ub.tum.de/1537464.

TABLE I: Parameters of our algorithm used for both the maps.

Parameter	Value
Δt [s]	0.25
t_h [s]	*map A:* 8.5, *map B:* 15.0
$t_{h_{\min}}$ [s]	1.5
$a_{s,\min}$ [m/s^2]	-7.0
$a_{s,\max}$ [m/s^2]	3.0
$\lvert a_{\max}\rvert$ [m/s^2]	-7.0
$Q_{\mathrm{s}}, Q_{\mathrm{d}}$	$\mathrm{diag}([5,1,1])$
$R_{\mathrm{s}}, R_{\mathrm{d}}$	0.5

[2]http://www.gurobi.com

[3]https://commonroad.in.tum.de/scenarios

A. Generation of Specifications

To generate a large number of specifications, we use a simple exhaustive search. A more distinct automation of formulating specifications is a subject of future work, and the utilized approach satisfies mainly the purpose of evaluating our synthesis approach. To generate specifications of the scenes, we manually define an initial scene, which is subsequently evolved through an exhaustive tree search, until a defined number of scenes is reached. For map A, we evolve a given scene to the next scene by letting up to two vehicles either change to the adjacent lanelet or to the succeeding lanelet. When switching lanes or merging into one lane, we add nodes in the search tree for each possible order of vehicles in the target lane. For map B, we let one vehicle switch to the following intersection predicate in the order $\big(\texttt{beforeCA}(V_i, V_j), \texttt{inCA}(V_i, V_j), \texttt{behindCA}(V_i, V_j)\big)$. To avoid collisions in the conflict area, we additionally disregard all the specifications that violate the condition in (2). Furthermore, we only consider specifications, wherein at least three vehicles cross the intersection.

B. Map A: Merging Lanes

The highway map features a road with merging lanes as depicted in Fig. 8. According to our definition, this map contains one route and accordingly one lane-based coordinate system. As depicted in Table II, the initial scene S_0 consists of two vehicles driving behind each other on each lane. Using the previously introduced approach for generating four scenes, we obtain 120 specifications in total for which we subsequently synthesize scenarios using our presented approach.

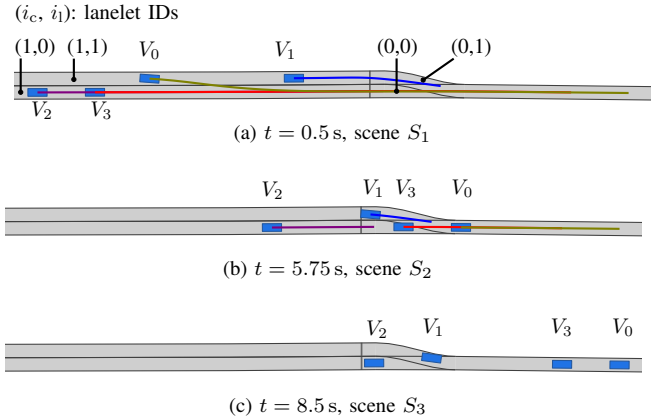
TABLE II: Specification of a scene sequence for Map A.

Scene	Predicates	Vehicle V_i: V_0,	V_1	V_2	V_3
S_0	$\texttt{onLanelet}(V_i, i_c, i_l)$	V_0,1,1	V_1,1,1	V_2,1,0	V_3,1,0
	$\texttt{isBehind}(V_i, V_j)$	V_0, V_1	-	V_2, V_3	-
S_1	$\texttt{onLanelet}(V_i, i_c, i_l)$	V_0,1,{0,1}	V_1,1,1	V_2,1,0	V_3,1,0
	$\texttt{isBehind}(V_i, V_j)$	V_0,V_1	-	V_2, V_3	V_3, V_0
S_2	$\texttt{onLanelet}(V_i, i_c, i_l)$	V_0,0,0	V_1,1,1	V_2,1,0	V_3,0,0
	$\texttt{isBehind}(V_i, V_j)$	-	V_1, V_3	V_2, V_3	-
S_3	$\texttt{onLanelet}(V_i, i_c, i_l)$	V_0,0,0	V_1,0,1	V_2,0,0	V_3,0,0
	$\texttt{isBehind}(V_i, V_j)$	V_0, V_1	V_1, V_3	V_2, V_1	V_3, V_0

Out of all specifications, 71 scenarios can be generated, taking on average 0.21 s. For the remaining specifications, the satisfiability checking failed. These specifications mainly contained overtaking maneuvers, which could not be completed in the prescribed time horizon t_h. In Table II we show an exemplary specification and Fig. 8 depicts three frames of the synthesized scenario.

C. Map B: Intersection

We use a T-intersection (Map B) to demonstrate intersection-related predicates. To obtain complex scenarios, we generate specifications for three routes that each intersect with the two other routes, as depicted in Fig. 9. In the



(a) $t = 0.5\,\mathrm{s}$, scene S_1

(b) $t = 5.75\,\mathrm{s}$, scene S_2

(c) $t = 8.5\,\mathrm{s}$, scene S_3

Fig. 8: Synthesized scenario for Map A at different times t.

initial configuration S_0, we place two vehicles on the lanelets approaching the intersection. By generating specifications as described in Section IV-A with six scenes, we obtain in total 557 different specifications T_{sc}. Our algorithm could generate scenarios for all the specifications. In Table III we show an example of a specification and the resulting scenario is depicted in Fig. 10. The computation takes on average 2.29 s.

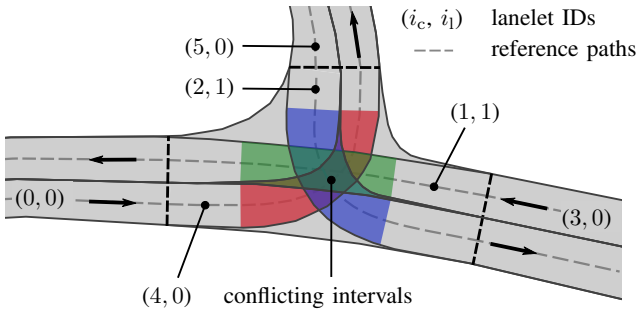


Fig. 9: Map B: reference paths of each route and conflicting intervals used in the experiments. Only lanelet IDs used in the specifications are shown.

V. Conclusions and Future Work

We presented a method for synthesizing traffic scenarios from specifications. Using this approach, scenarios can be generated efficiently instead of searching large databases for scenarios with matching specifications. By tailoring dedicated specifications, one could generate scenarios in which vehicles obey or disregard certain traffic rules. Our method can be further extended to a reactive environment for testing of automated vehicles, where the specification-conforming motion of surrounding vehicles is adapted to the action of the ego vehicle. Future work also includes the automatic generation of specifications and the design of dedicated cost functions for the creation of critical scenarios.

Acknowledgements

The authors gratefully acknowledge financial support by the Central Innovation Programme of the German Federal Government under grant ZF4086007BZ8.

TABLE III: Specification of a scene sequence for Map B.

	Predicates	Vehicle V_i V_0	V_1	V_2	V_3	V_4	V_5
S_0	onLanelet(V_i, i_c, i_l)	V_0,3,0	V_1,3,0	V_2,0,0	V_3,0,0	V_4,5,0	V_5,5,0
	isBehind(V_i, V_j)	V_0, V_1	-	V_2, V_3	-	V_4, V_5	-
S_1	onLanelet(V_i, i_c, i_l)	V_0,3,0	V_1,3,0	V_2,0,0	V_3,0,0	V_4,5,0	V_5,2,1
	isBehind(V_i, V_j)	V_0, V_1	-	V_2, V_3	-	V_4, V_5	-
	conflict area predicate	-	-	-	-	-	before
S_2	onLanelet(V_i, i_c, i_l)	V_0,1,1	V_1,1,1	V_2,0,0	V_3,4,0	V_4,5,0	V_5,2,1
	isBehind(V_i, V_j)	V_0, V_1	-	V_2, V_3	-	V_4, V_5	-
	conflict area predicate	-	before		before	-	in
S_3	onLanelet(V_i, i_c, i_l)	V_0,1,1	V_1,1,1	V_2,0,0	V_3,4,0	V_4,2,1	V_5,2,1
	isBehind(V_i, V_j)	V_0, V_1	-	V_2, V_3	-	V_4, V_5	-
	conflict area predicate	before	in	-	before	before	behind
S_4	onLanelet(V_i, i_c, i_l)	V_0,1,1	V_1,1,1	V_2,0,0	V_3,4,0	V_4,2,1	V_5,2,1
	isBehind(V_i, V_j)	V_0, V_1	-	V_2, V_3	-	V_4, V_5	-
	conflict area predicate	before	in	-	in	before	behind
S_5	onLanelet(V_i, i_c, i_l)	V_0,1,1	V_1,1,1	V_2,4,0	V_3,4,0	V_4,2,1	V_5,2,1
	isBehind(V_i, V_j)	V_0, V_1		V_2, V_3	-	V_4, V_5	-
	conflict area predicate	before	behind	before	behind	before	behind

References

[1] M. R. Zofka, F. Kuhnt, R. Kohlhaas, C. Rist, T. Schamm, and J. M. Zollner, "Data-driven simulation and parametrization of traffic scenarios for the development of advanced driver assistance systems," in *18th Int. Conf. on Information Fusion*, 2015, pp. 1422–1428.

[2] S. Wagner, K. Groh, T. Kuhbeck, and A. Knoll, "Towards cross-verification and use of simulation in the assessment of automated driving," in *Proc. of the IEEE Intelligent Vehicles Symposium*. IEEE, 2019, pp. 1589–1596.

[3] T. A. Wheeler, M. J. Kochenderfer, and P. Robbel, "Initial Scene Configurations for Highway Traffic Propagation," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2015, pp. 279–284.

[4] S. Jesenski, J. E. Stellet, F. Schiegg, and J. M. Zollner, "Generation of scenes in intersections for the validation of highly automated driving functions," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 502–509.

[5] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated Evaluation of Automated Vehicles Safety in Lane-Change Scenarios Based on Importance Sampling Techniques," *IEEE Trans. on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 595–607, 2017.

[6] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," in *Proc. of the 32nd Int. Conf. on Neural Information Processing Systems*, 2018, pp. 9849–9860.

[7] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1555–1562.

[8] C. Gladisch, T. Heinz, C. Heinzemann, J. Oehlerking, A. V. Vietinghoff, and T. Pfitzer, "Experience paper : search-based Testing in automated driving control Applications," in *Proc. of the 34th IEEE/ACM Int. Conf. on Automated Software Engineering*, 2019, pp. 26–37.

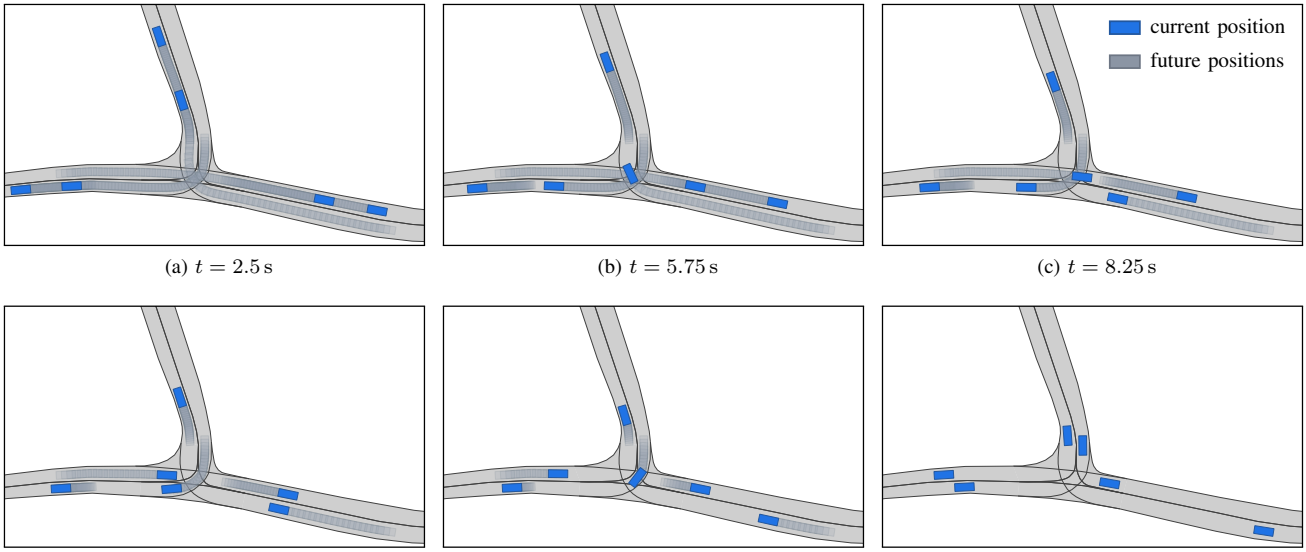
[9] H. Zhang, A. Geiger, and R. Urtasun, "Understanding high-level semantics by modeling traffic patterns," in *Proc. of the IEEE Int. Conf. on Computer Vision*, 2013, pp. 3056–3063.

[10] F. Kruber, J. Wurst, and M. Botsch, "An Unsupervised Random Forest Clustering Technique for Automatic Traffic Scenario Categorization," in *Proc. of the IEEE Conf. on Intelligent Transportation Systems*, 2018, pp. 2811–2818.

[11] J. Bach, J. Langner, S. Otten, E. Sax, and M. Holzapfel, "Test scenario selection for system-level verification and validation of geolocation-dependent automotive control systems," in *23rd Int. Conf. on Engineering, Technology and Innovation*, 2018, pp. 203–210.

[12] C. Sippl, F. Bock, D. Wittmann, H. Altinger, and R. German, "From simulation data to test cases for fully automated driving and ADAS," in *Lecture Notes in Computer Science*, 2016, pp. 191–206.

[13] A. Gambi, T. Huynh, and G. Fraser, "Generating effective test cases for self-driving cars from police reports," in *Proc. of the 27th ACM Joint*



(a) $t = 2.5\,\mathrm{s}$ (b) $t = 5.75\,\mathrm{s}$ (c) $t = 8.25\,\mathrm{s}$

(d) $t = 10.0\,\mathrm{s}$ (e) $t = 12.25\,\mathrm{s}$ (f) $t = 14.75\,\mathrm{s}$

Fig. 10: Synthesized scenario for Map B at different points in time t.

Meeting on European Software Engineering Conf. and Symposium on the Foundations of Software Engineering, 2019, pp. 257–267.

[14] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, "A systematic review of the application and empirical investigation of search-based test case generation," *IEEE Transactions on Software Engineering*, vol. 36, no. 6, pp. 742–762, 2010.

[15] F. Hauer, A. Pretschner, and B. Holzmüller, "Fitness functions for testing automated and autonomous driving systems," in *SAFECOMP 2019: Computer Safety, Reliability, and Security*, pp. 69–84.

[16] G. Bagschik, T. Menzel, and M. Maurer, "Ontology based Scene Creation for the Development of Automated Vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1813–1820.

[17] T. Menzel, G. Bagschik, L. Isensee, A. Schomburg, and M. Maurer, "From Functional to Logical Scenarios: Detailing a Keyword-Based Scenario Description for Execution in a Simulation Environment," *Proc. of the IEEE Intelligent Vehicles Symposium*, pp. 2383–2390, 2019.

[18] Y. Li, J. Tao, and F. Wotawa, "Ontology-based test generation for automated and autonomous driving functions," *Information and Software Technology*, vol. 117, Art. no. 106200, 2020.

[19] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with Linear Temporal Logic specifications," in *Proc. of the IEEE Conf. on Decision and Control*, 2008, pp. 2117–2122.

[20] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 5319–5325.

[21] D. J. Fremont, X. Yue, T. Dreossi, A. L. Sangiovanni-Vincentelli, S. Ghosh, and S. A. Seshia, "Scenic: A language for scenario specification and scene generation," in *Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation*, 2019, pp. 63–78.

[22] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, and D. Zufferey, "PARACOSM: A language and tool for testing autonomous driving systems," 2019, arXiv:1902.01084.

[23] A. Eggers, M. Stasch, T. Teige, T. Bienmüller, and U. Brockmeyer, "Constraint Systems from Traffic Scenarios for the Validation of Autonomous Driving," in *Third International Workshop on Satisfiability Checking and Symbolic Computation, Part of FLOC 2018*, pp. 95–109.

[24] K. Scheibler, A. Eggers, T. Teige, M. Walz, T. Bienmüller, and U. Brockmeyer, "Solving Constraint Systems from Traffic Scenarios for the Validation of Autonomous Driving," in *Proc. of the 4th SC-square Workshop*, 2019, pp. 2–12.

[25] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: efficient map representation for autonomous driving," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 420–425.

[26] M. Klischat and M. Althoff, "Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 2352–2358.

[27] M. I. Shamos and D. Hoey, "Geometric intersection problems," in *Proc. of the 17th Annual Symposium on Foundations of Computer Science*, 1976, pp. 208–215.

[28] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving," in *Proc. of the IEEE Conf. on Intelligent Transportation Systems*, 2015, pp. 982–988.

[29] C. Katrakazas, M. Quddus, W. H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

[30] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[31] J. Peng and S. Akella, "Coordinating Multiple Robots with Kinodynamic Constraints Along Specified Paths," *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, 2005.

[32] F. Altché, X. Qian, and A. De La Fortelle, "Time-optimal coordination of mobile robots along specified paths," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2016, pp. 5020–5026.

[33] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *European Control Conference*, 2001, pp. 2603–2608.

[34] C. Miller, C. Pek, and M. Althoff, "Efficient Mixed-Integer Programming for Longitudinal and Lateral Motion Planning of Autonomous Vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1954–1961.

[35] V. Raman, M. Maasoumy, A. Donze, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. of the IEEE Conf. on Decision and Control*, 2014, pp. 81–87.

[36] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *Proc. of the 53rd Annual Allerton Conference on Communication, Control, and Computing*, 2016, pp. 772–779.

[37] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization*. SIAM, 2009.

4.2 Falsifying Motion Plans of Autonomous Vehicles with Abstractly Specified Traffic Scenarios

To solve Problem 2, we integrate a method to integrate the synthesis of scenarios from abstract specifications with search algorithms that find a concrete scenario falsifying a provided SUT. For search algorithms it can become challenging to identify feasible solutions from the solution space of an abstract scenario that involves multiple traffic participants and/or complex behavior specifications. To guide the search to feasible solutions and enable a faster falsification, we propose a method that computes the solution space of an abstract scenario using reachability analysis and subsequently prunes irrelevant scenarios from the search space.

To compute the reachable sets, we combine reachability analysis with an over-approximative bounding operation balancing accuracy with efficiency to tackle the computational complexity that arises from the curse of dimensionality. From the reachable sets, we compute input constraints for the traffic participants of the scenario that are used to restrict the action space of each node during the MCTS. Thus, we are able to ensure that the search stays within the space of specification-compliant scenarios. Since we can also specify the failure of interest in the scenario specification, the scenario constraints also ensure that the MCTS selects only actions that can potentially result in failures at future time steps.

We evaluate the performance and efficiency of our approach using a motion planning algorithm as an SUT and several abstractly defined scenarios. We show that the approach is able to find failures in most cases and that the reachability-based action constraints increase the success rate significantly. Furthermore, the approach results in tighter action constraints the more complex the scenario specification becomes, which makes the MCTS more efficient.

Contributions M. K. initiated the idea to use reachability analysis for computing the solution space of abstract traffic scenarios and developed the method for using the reachability analysis to guide a search-based falsification algorithm. M. A. led the research project and provided feedback to improve the manuscript.

Journal Article In this thesis, the accepted version submitted by the author is reprinted. The edited version is available under https://doi.org/10.1109/TIV.2022.3191179.

Falsifying Motion Plans of Autonomous Vehicles With Abstractly Specified Traffic Scenarios

Author: Moritz Klischat; Matthias Althoff

Publication: IEEE Transactions on Intelligent Vehicles

Publisher: IEEE

Date: Feb. 2023

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2022.3191179

Falsifying Motion Plans of Autonomous Vehicles with Abstractly Specified Traffic Scenarios

Moritz Klischat and Matthias Althoff, *Member, IEEE*

***Abstract*—Verifying the safety of autonomous vehicles is one of the major challenges towards their deployment on public roads due to the vast number of possible situations that can occur in traffic. Scenario-based testing has been proposed to reduce the number of required tests using catalogs of abstractly defined scenarios. However, when specifying test scenarios abstractly, there is still an infinite number of possible concrete scenarios that can be derived from a specification. Available computational resources can thus be easily exceeded when using inefficient strategies for deriving concrete scenarios. In this work, we propose a novel method that synthesizes concrete scenarios complying with abstract scenario specifications. We compute the set of compliant trajectories for surrounding traffic participants and only sample trajectories within those sets. Our synthesis integrates a falsification algorithm that searches for specified failures of the vehicle under test. Compared to existing work, we can efficiently find scenario concretizations, especially for complex maneuver specifications. By directly considering failure specifications during the scenario synthesis, we avoid executing irrelevant simulations that cannot possibly result in failures. Our approach is demonstrated in several scenarios using Monte Carlo tree search as a search algorithm.**

***Index Terms*—Falsification, motion planning, reachability analysis, automated vehicles, scenario-based testing, Monte Carlo tree search**

I. Introduction

AUTONOMOUS vehicles are expected to safely handle a vast number of situations in the real world. To test the safety of such systems in sufficiently many situations, scenario-based testing is proposed to define representative scenarios [1]. These scenarios are typically structured into several abstraction levels. On the most abstract level, the semantics of the maneuvers of other traffic participants are defined, while concrete scenarios on the lowest level define the exact motion of each traffic participant as illustrated in Fig. 1. Thus, abstractly defined maneuvers serve as a specification for concrete scenarios to which further specifications such as traffic rules or the expected behavior of the autonomous vehicle can be added. When deriving concrete scenarios complying with the specifications, existing approaches often depend either on restrictive behavior models for other traffic participants or use sampling methods for which it becomes challenging or inefficient to obtain specification-compliant motions once the scenario specifications become more complex.

In contrast, we propose a scenario synthesis method that generates scenarios complying with spatio-temporal specifications of multiple traffic participants using reachability analysis.

Moritz Klischat and Matthias Althoff are with the Department of Computer Science, Technical University of Munich, D-85748 Garching, Germany (e-mail: {moritz.klischat,althoff}@tum.de).

Manuscript received April 30, 2022.

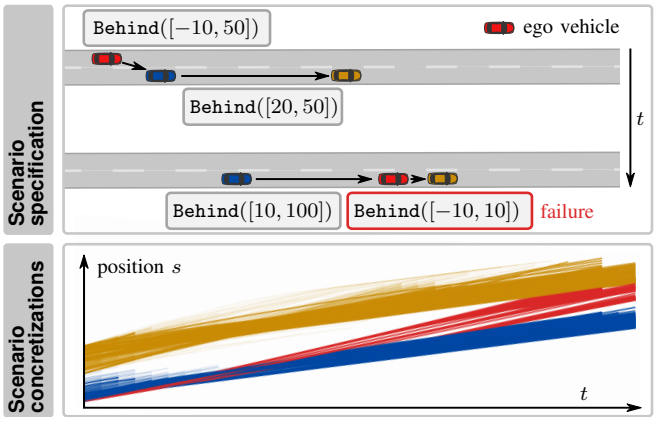


Figure 1. A scenario specification abstractly describes the scenario and a failure of the ego vehicle, e.g., using a predicate Behind($[a, b]$) that specifies the allowed interval of the distance between two vehicles. Our approach efficiently finds concrete failures by computing the set of specification-compliant trajectories and using search algorithms that sample trajectories from this set.

We first obtain the solution space of all trajectories complying with the specification and subsequently synthesize trajectories within this solution space. Because we can also formulate in the specification the failures we are interested in, we only obtain relevant scenarios that can possibly result in these failures. Our synthesis approach integrates with search algorithms and enables testing in a closed loop with the system under test (SUT) to adapt to its behavior for falsifying given traffic rules. Because we can compute the solution space, we are able to guide the scenario sampling process towards hardly reachable edge cases, that existing sampling methods might miss or require many rejections of samples.

A. Literature Review

Recent surveys provide a comprehensive overview of the safety validation of autonomous systems in general [2] and of autonomous vehicles, specifically [1], [3]. In the following review, we focus on the most recent and relevant research with respect to our approach.

1) Data-driven Scenario Classification and Sampling: A standard approach is to create test scenarios from real-world data that are subsequently classified into abstract scenario representations, see e.g., [4]–[10]. However, even for large datasets, it is not ensured that they contain the abstract scenarios of interest or enough variations of the same abstract scenario. Unseen scenarios can be sampled from probabilistic models that are learned from a database [11]–[13]. Importance sampling is proposed to increase the efficiency of sampling

methods and to focus on rarely occurring behavior, is proposed [14], [15]. Using extreme value theory, the collision frequency of an automated vehicle is estimated based on the distribution of a criticality measure evaluated on a dataset [16].

Other works parameterize trajectories from real traffic data and derive variations of the original scenarios [17], [18]. Another line of work focuses on generating diverse variations of scenarios using machine learning [19]–[21].

2) Falsification-based Scenario Generation: When verifying software components, they are typically evaluated against their functional requirements provided as formal specifications. Falsification algorithms aim at falsifying the SUT against these specifications, often formulated using temporal logic [22], [23]. In contrast to the previously reviewed methods for generating scenarios, falsification methods adapt the scenario in a closed loop incorporating the behavior of the SUT to find failures. While falsification algorithms have already been applied to highly nonlinear or hybrid systems [24], a specific challenge for autonomous vehicles is the complex behavior of the environment, see, e.g., [25]–[27].

Extensions to falsification problems incorporate the likelihood of scenarios and frame the problem as a Markov decision process to find the most likely failures [28], [29]. Similarly, adversarial trajectories of the surrounding traffic participants are generated using learning methods [30]–[32], optimization [33]–[36], or search methods [37], [38]. Apart from temporal logic, falsification methods often use criticality measures as heuristics to guide the search towards failures. An extensive overview of criticality metrics is provided in [39], [40].

3) Specification-compliant Scenario Generation: While the approaches reviewed so far can generate a large number of scenarios, they often do not explicitly consider specified behavior of other traffic participants or objects of the environment. To specify traffic scenarios, domain-specific languages have been developed [41]–[43]. The languages Scenic and Paracosm are integrated with testing frameworks capable of executing scenarios formulated using these languages [43], [44]. Another domain-specific language is OpenScenario 2.0[1]. Ontologies are proposed in [45], [46] for describing scenarios.

Conformance to a scenario specification can be considered in search-based methods through dedicated cost functions [33], [47]. With these heuristics, the search is directed to regions in the space of scenario parameters where the specification is fulfilled. However, this approach does not provide any guarantees for complying with the specification. Traffic rules for the surrounding agents are encoded using temporal logic in [48] and incorporated in the reward of a reinforcement learning problem for falsifying the SUT. A work that proposes efficient sampling of trajectories from Gaussian distributions complying with linear constraints is proposed in [49]. Valid instantiations compliant with a domain-specific language and considering vehicle dynamics can be obtained using a dedicated SMT solver [50]. In previous work, we generated specification-compliant scenarios using mixed-integer optimization [51].

[1]http://www.asam.net

4) Coverage-focused Scenario Generation: The continuous domains of many scenario parameters entail infinitely many scenarios. Even after discretizing the uncountable parameter space, efficient testing strategies are crucial due to the combinatorial explosion of discretized parameter values. Therefore, coverage criteria are incorporated into the scenario generation: In [52], [53] methods to achieve coverage of high-level scenarios are proposed, whereas [54]–[57] consider a low-level combinatorial coverage of parameter values by utilizing the k-wise combination of values within provided parameter intervals.

B. Contributions

In summary, there exist methods for concretizing abstractly defined scenarios and falsification procedures for autonomous vehicles. However, little focus has been on methods enabling both jointly, i.e., the efficient closed-loop falsification for complex abstract scenarios. However, this is crucial when utilizing such methods at scale.

We address this research gap by presenting a method for synthesizing concrete trajectories for multiple adversarial traffic participants reacting to the SUT while complying with an abstract scenario specification.

Our method provides the following features:

1) Finding falsifying scenario concretizations complying with specifications that define semantic and temporal relations for multiple traffic participants.
2) More efficient sampling compared to rejection-sampling techniques by only considering the solution space of feasible scenario parameters with respect to scenario specifications and vehicle dynamics through the combination of search algorithms with reachability analysis.
3) As a consequence of 2), our scenario synthesis algorithm becomes the more efficient the more complex the scenario becomes due to the shrinking solution space.

C. Organization

This paper is organized as follows: We introduce the preliminaries for the scenario representation in Section II and the problem definition in Section III. The abstract scenario representation is defined in Section IV from which we derive constraints for the reachability analysis in Section V. The reachable sets are used to obtain input constraints for the scenario sampling strategy. In Section VI, we show how the scenario synthesis can be integrated with falsification algorithms using Monte Carlo tree search as a scenario sampling strategy. The approach is evaluated in Section VII, and we finish with the conclusions in Section VIII.

II. Preliminaries

In this work, we use discretized time $t_k = k\Delta t$, where Δt is the time step and k is the time index. We generate traffic scenarios, which are defined by traffic participants V_i following trajectories $x_i(k) \in \mathbb{R}^n, k \in [0, k_\mathrm{f}]$, where $i \in \{1, \dots, n_\mathrm{p}\}$ is the index of the traffic participant, and $k_\mathrm{f} \in \mathbb{N}$ denotes the considered time horizon. The index ego identifies the ego vehicle. To denote a trajectory in the time interval $[k_0, k_\mathrm{f}]$, we use the notation $x_i([k_0, k])$.

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2022.3191179

A. Road Representation

We model roads by a network of lanelets [58].

Definition 1 (Lanelet):
A lanelet L_l with index l is composed of a left and right border defined by polylines and attributes describing its spatial relations to other lanelets: *successors*, *predecessors*, *adjacent_right*, and *adjacent_left*.

A lanelet is said to be *adjacent_right* to a lanelet L_l, when its left lanelet border intersects with L_l across its entire length. The relation *adjacent_left* is defined analogously. To further structure the lanelet network, we introduce lanelet sections C_c combining lanelets coupled laterally through the relations *adjacent_left* and *adjacent_right* as illustrated in Fig. 2. We define routes R_ϱ with route index ϱ as a tuple of connected lanelet sections ordered in the longitudinal direction. The trajectory of each traffic participant V_i is assigned to a route. For this mapping, we introduce the operator $\mathsf{route}(V_i)$, which maps a traffic participant to a route R_ϱ.

B. Curvilinear Coordinate Systems

We use curvilinear coordinate systems defined by a reference path, which is constructed for a route R_ϱ by concatenating the center lines of consecutive lanelets from each section of the route. The longitudinal state of a traffic participant V_i in the coordinate system of $\mathsf{route}(V_i)$ is given by $x_{\xi,i} = [s_i, \dot{s}_i]^T$, where $s_i \in \mathbb{R}$ is the longitudinal position, and the lateral state is given by $x_{\eta,i} = [d_i, \dot{d}_i]^T$ with $d_i \in \mathbb{R}$ being the lateral position (see Fig. 2). Together, they define the state of a traffic participant $x_i = [x_{\xi,i}, x_{\eta,i}]^T$. Subsequently, we denote variables related to the longitudinal direction by the subscript ξ and correspondingly variables related to the lateral direction by the subscript η.

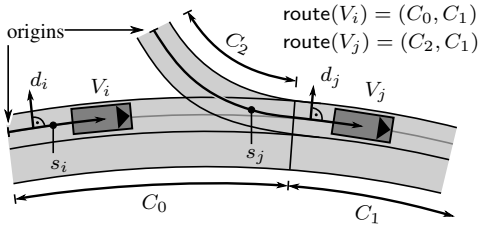


Figure 2. Example of lanelet sections C and curvilinear coordinate systems.

C. Vehicle Dynamics and Reachable Sets

Let us assume a scenario specification ϕ_s, which we will define in more detail in Section IV. We say a scenario state $x^\mathrm{S}(k) = [x_1(k), \dots, x_i(k), \dots, x_{n_\mathrm{p}}(k)]^T \in \mathcal{X}^\mathrm{S}$ is compliant with ϕ_s if the states of all traffic participants fulfill the scenario specification at time t_k, written as $x^\mathrm{S}(k) \models \phi_\mathrm{s}$. We use the projection operator $\mathsf{proj}_{x_i} : \mathcal{X}^\mathrm{S} \to \mathcal{X}_i$ to extract the set of state variables corresponding to the i-th traffic participant from the combined state space $\mathcal{X}^\mathrm{S}$. We define the set operator $\mathsf{box}(\mathcal{X})$ returning the tightest interval enclosing a set $\mathcal{X}$.

We further introduce the vector $u^\mathrm{S}(k) = [u_1(k), \dots, u_i(k), \dots, u_{n_\mathrm{p}}(k)]^T \in \mathcal{U}^\mathrm{S}$ as the scenario input with $u_i = [u_{\xi,i}, u_{\eta,i}]^T$ consisting of inputs, which are bounded by intervals $u_{\xi,i} \in \mathcal{U}_\xi$, $\mathcal{U}_\xi = [\underline{u}_{\min,\xi}, \overline{u}_{\max,\xi}] \subset \mathbb{R}$ in the longitudinal direction and correspondingly in the lateral direction. To permit efficient computations, we synthesize the trajectories using decoupled, linear system dynamics in the longitudinal and lateral direction: $\ddot{s}_i = u_{\xi,i}$ and $\ddot{d}_i = u_{\eta,i}$. The coupling of both dynamics is achieved later in the synthesis through additional constraints of $u_{\eta,i}(k)$ that account for the friction circle $u^2_{\xi,i}(k) + u^2_{\eta,i}(k) \leq a^2_{\max}$. As a consequence of the introduced discrete time, we have discretized dynamics

$$x_{\xi,i}(k+1) = \underbrace{\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}}_{A} x_{\xi,i}(k) + \underbrace{\begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix}}_{b} u_{\xi,i}(k) \quad (1)$$

for each traffic participant in the longitudinal direction and likewise in the lateral direction. For the SUT, we assume a black-box model.

Definition 2 (System Under Test):
The system under test (SUT) is represented by a black-box model that returns the state of the ego vehicle V_ego based on the current state of the scenario: $x_\mathrm{ego}(k) = M_\mathrm{SUT}(x^\mathrm{S}(k))$.

Let us further define $\chi\big(k, u^\mathrm{S}([0,k]), x^\mathrm{S}(0)\big)$, which denotes the scenario state of all traffic participants at time k resulting from the initial state $x^\mathrm{S}(0)$ and the input trajectory $u^\mathrm{S}([0,k])$. Finally, we define reachable sets, which later help us finding compliant trajectories:

Definition 3 (Compliant Reachable Set):
Starting from an initial set $\overline{\mathcal{R}}_0 \subset \mathcal{X}^\mathrm{S}$, we define

$$\overline{\mathcal{R}}(k, \phi_\mathrm{s}, \mathcal{U}^\mathrm{S}, \overline{\mathcal{R}}_0) = \Big\{ \chi\big(k, u^\mathrm{S}([0,k]), x^\mathrm{S}(0)\big) \;\Big|\; x^\mathrm{S}(0) \in \overline{\mathcal{R}}_0, \; \forall \kappa \in [0, k_\mathrm{f}] : u^\mathrm{S}(\kappa) \in \mathcal{U}^\mathrm{S}, \chi\big(\kappa, u^\mathrm{S}([0,\kappa]), x^\mathrm{S}(0)\big) \models \phi_\mathrm{s} \Big\}$$

as the compliant reachable set, which contains all states at the k-th time step that comply with a specification ϕ_s for the time interval $[0, k_\mathrm{f}]$ and consider the system dynamics.

To simplify the notation, we will only write $\overline{\mathcal{R}}(k)$ in the remaining paper. Furthermore, we will need to compute $\overline{\mathcal{R}}(k)$ over-approximatively, which we denote by $\mathcal{R}(k) \supseteq \overline{\mathcal{R}}(k)$. In Table I we also summarize all the subsequently introduced notations for reachable sets. For computing the reachable set, we later require the one-step reachable set that is computed using the Minkowski sum $\oplus$ defined as $\mathcal{A} \oplus \mathcal{B} := \{\; a + b \;|\; \forall a \in \mathcal{A}, \forall b \in \mathcal{B} \;\}$.

Definition 4 (One-Step Reachable Set):
When only considering system dynamics provided by A, b, and the reachable set $\mathcal{R}_{\xi,i}(k)$, the one-step reachable set at the subsequent time step $k+1$ is obtained using a set-based evaluation of (1):

$$\overrightarrow{\mathsf{reach}}\big(\mathcal{R}_{\xi,i}(k)\big) := A\mathcal{R}_{\xi,i}(k) \oplus b\mathcal{U}_\xi. \quad (2)$$

D. Monte Carlo Tree Search

As a search algorithm for solving the falsification problem, we use Monte Carlo tree search, which is widely used as a policy for Markov decision processes but also for solving

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2022.3191179

deterministic games with a high branching factor resulting from a large space of possible actions like in our case [59]. Previous works used it for falsification of cyber-physical systems [60]. Another advantage is that Monte Carlo tree search does not require a search heuristic because it uses playouts to estimate action values. For defining the game, we introduce general states $\sigma \in \mathcal{S}$ and actions $\alpha \in \mathcal{A}'(\sigma) \subseteq \mathcal{A}$ which are not limited to the state space $\mathcal{X}^{\mathrm{S}}$ and input space $\mathcal{U}^{\mathrm{S}}$ but additionally include high-level scenario parameters as explained later in Section VI. In our case, the action space $\mathcal{A}'(\sigma)$ will be constrained depending on the state. The actions α result in transitions $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Once the search reaches a terminal state $\sigma_{\mathrm{T}} \in \mathcal{S}_{\mathrm{T}} \subseteq \mathcal{S}$ it is evaluated using a reward function $R : \mathcal{S}_{\mathrm{T}} \to \mathbb{R}^{+}$. During the search, a search tree is grown where nodes represent states σ and edges represent actions α. The aim is to find a sequence of actions that yield a desired terminal state. During the search, the state-action values are estimated using a heuristic $Q(\sigma, \alpha)$ explained in Appendix A. Starting at the root node, each iteration in Monte Carlo tree search comprises four main steps [59] illustrated in Fig. 3:

1) *Selection*: Starting from the root node, the search tree is traversed by successively choosing actions based on the tree policy $\pi_{\mathrm{T}}(\sigma, \mathcal{A}'(\sigma)) = \mathrm{argmax}_{\alpha \in \mathcal{A}'(\sigma)} Q(\sigma, \alpha)$ until encountering a node without child nodes, i.e., a leaf node σ_{L}.
2) *Expansion*: The leaf node is expanded, i.e., new nodes are added to the tree for each action $\alpha \in \mathcal{A}'(\sigma_{\mathrm{L}})$ and one of them is selected randomly.
3) *Playout*: Starting from the obtained node, a playout policy $\pi_{\mathrm{P}}(\sigma, \mathcal{A}'(\sigma)) : \mathcal{S} \times P(\mathcal{A}) \to \mathcal{A}'(\sigma)$, with $P()$ being the power set, successively selects actions until reaching a terminal state σ_{T}.
4) *Backpropagation*: The reward $R(\sigma_{\mathrm{T}})$ of the playout is saved for each node along the path selected in step 1) as the average reward of a node is used to compute $Q(\sigma, \alpha)$ in subsequent iterations.

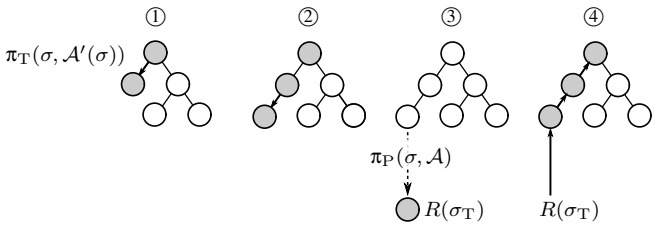


Figure 3. Main steps of an iteration in Monte Carlo tree search.

III. Problem Statement and Solution Concept

A. Problem Statement

We are provided with a black-box model $M_{\mathrm{SUT}}(x^{\mathrm{S}}(k))$ of the SUT and a scenario specification

$$\phi_{\mathrm{s}} = \phi_{\mathrm{e}} \wedge \phi_{\mathrm{f}}$$

comprising a maneuver specification ϕ_{e} describing the behavior of all traffic participants and a failure specification ϕ_{f} describing, e.g., a collision or a traffic rule violation. We consider the problem of finding a concrete scenario represented by a trajectory $x^{\mathrm{S}}([0, k_{\mathrm{f}}])$ that complies with ϕ_{s}, i.e., $x^{S}([0, k_{\mathrm{f}}]) \models \phi_{s}$.

B. Solution Concept

Our solution concept is depicted in Fig. 4: The derivation of concrete scenarios is divided into two steps: At each time step, a *scenario synthesizer* first translates the specification into input bounds for the other traffic participants to ensure specification-compliant trajectories considering their dynamical models and the previous scenario state $x^{\mathrm{S}}(k-1)$. To achieve this, we compute for all traffic participants the reachable sets compliant with the specification and determine input bounds based on these sets. A *scenario sampling strategy* subsequently selects the inputs for the other traffic participants and updates the scenario state $x^{\mathrm{S}}(k)$. The SUT updates its state based on $x^{\mathrm{S}}(k)$ and the loop is closed to obtain a reactive behavior of the environment.

The objective of the synthesizer is to provide wide input bounds for allowing the sampler to generate a large variety of scenarios, including unusual edge cases that trigger failures of the SUT. Yet, the bounds should be chosen such that we avoid generating trajectories that result in incompliant scenarios at a later point in time, which wastes computational resources by rejecting those. The scenario sampling strategy can be an arbitrary algorithm that is capable of generating values from a bounded interval, which comprises, e.g., samplers using probabilistic distributions obtained from naturalistic driving data or in our case, search algorithms, such as Monte Carlo tree search. In Fig. 5 we depict an example for the longitudinal

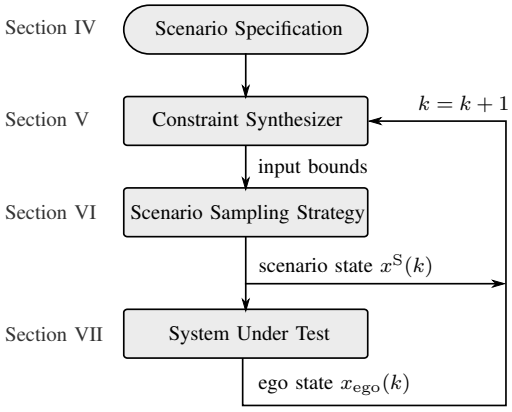


Figure 4. Illustration of the solution concept.

reachable sets of three traffic participants and a concrete scenario that is sampled for demonstration purposes randomly within the sets.

IV. Scenario Specification

We formalize the specification ϕ_{s} using an intentionally simple specification format so that it can be expressed using domain-specific languages reviewed in Section I-A3.

Definition 5 (Scene):
A scene S_q with scene index q is defined by the lower

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2022.3191179

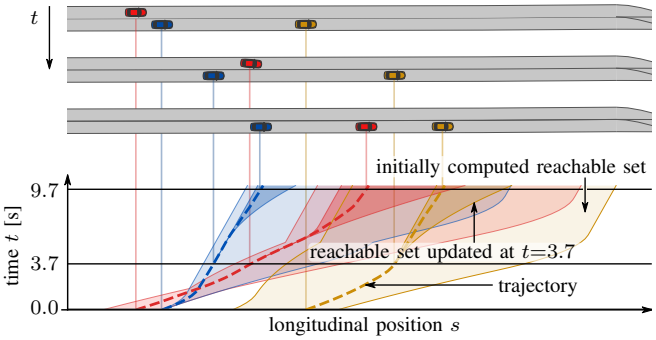


Figure 5. Example for concrete trajectories for an overtaking scenario. We first compute the reachable sets of each traffic participant which we plot projected to the t-s plane. The sets are continuously updated by the constraint synthesizer after each sampling step, for which we show an example at t=3.7. Snapshots of the scenario sampled within the reachable set are shown above.

and upper bounds of its duration $[\underline{\delta}_q, \overline{\delta}_q] \subset \mathbb{R}^+$ and a set of predicates $\mathcal{P}_q$ that hold true $\forall t_k \in [\tau_q, \tau_q + \delta_q]$, where $\delta_q \in [\underline{\delta}_q, \overline{\delta}_q]$ and the initial time τ_q of the scene follows from $\tau_q = \sum_{\varrho=1}^{q-1} \delta_\varrho$.

Definition 6 (Scenario Specification):
A scenario specification ϕ_s defines semantic relations between traffic participants using a sequence of consecutive scenes $(S_1, \dots, S_q, \dots, S_m)$.

At $k = 0$, $\mathcal{R}(0)$ provides the set of possible initial states. The predicates specify properties of traffic participants or relations between a pair of traffic participants or other scenario elements, such as lanelets. Fig. 6 depicts the example of a simple overtaking maneuver that is specified by three scenes using predicates that we define next.

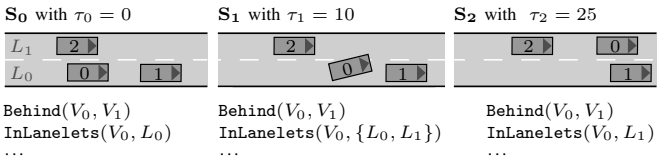


Figure 6. An example of a lane-change maneuver defined by three scenes.

As shown later, our approach requires the predicates to be formulated using linear inequality constraints of the form $a \leq \nu_j - \nu_i \leq b$, with $a, b \in \mathbb{R}$ and a state variable $\nu \in \{s, d, \dot{s}, \dot{d}\}$ yielding convex solution spaces for each traffic participant. For this work, we use the predicates defined next.

Definition 7 (InLanelets):
Let a set of lanelets $\mathcal{L}$ be given, which are either laterally or longitudinally adjacent. The longitudinal bounds of all lanelets in $\mathcal{L}$ in the coordinate system of the considered vehicle V_i with position $[s_i, d_i]^T$ are denoted as $[\underline{s}_\mathcal{L}, \overline{s}_\mathcal{L}]$ and the lateral bounds at a longitudinal position s_i are within the interval $[\underline{d}_\mathcal{L}(s_i), \overline{d}_\mathcal{L}(s_i)]$. Then the predicate is defined as

$$\texttt{InLanelets}(V_i, \mathcal{L}) \iff \underline{s}_\mathcal{L} \leq s_i \leq \overline{s}_\mathcal{L} \quad \wedge \\ \underline{d}_\mathcal{L}(s_i) \leq d_i \leq \overline{d}_\mathcal{L}(s_i).$$

To further specify the position within the lanelet, the following two predicates are introduced:

Definition 8 (LonPosition):
The longitudinal position of a traffic participant is constrained by

$$\texttt{LongitudinalPosition}(V_i, [a, b]) \iff a \leq s_i \leq b.$$

Definition 9 (LatPosition):
The lateral position of a traffic participant is constrained by

$$\texttt{LateralPosition}(V_i, [a, b]) \iff a \leq d_i \leq b.$$

To define speed limits or to model traffic situations such as congestions, we specify the velocity range.

Definition 10 (VelocityRange):
The velocity of a traffic participant is constrained by

$$\texttt{VelocityRange}(V_i, [a, b]) \iff a \leq \dot{s}_i \leq b.$$

For two traffic participants V_i and V_j, our specification can include the longitudinal distance between these traffic participants from possibly different coordinate systems for which we use an operator $\mathsf{dist}(s_i, s_j)$ that returns the longitudinal distance between V_i and V_j. Using the following predicates, relations between traffic participants are specified.

Definition 11 (Behind):
If V_i and V_j are specified to move on the same lanelet or on merging, forking, or succeeding lanelets, we define that vehicle V_i is behind vehicle V_j within a specified distance through the predicate

$$\texttt{Behind}(V_i, V_j, [a, b]) \iff a \leq \mathsf{dist}(V_i, V_j) \leq b.$$

Definition 12 (DrivesFaster):
The velocity difference between two traffic participants is defined by

$$\texttt{DrivesFaster}(V_i, V_j, [a, b]) \iff a \leq \dot{s}_i - \dot{s}_j \leq b.$$

With these predicates, one can model further high-level predicates relevant for, e.g., traffic rules [61]. For instance, the predicate `OnAccessRamp` can be expressed using $\texttt{InLanelets}(V_i, \{L_\mathrm{AR}\})$, where L_AR is a lanelet modeling an access ramp.

V. Constraint Synthesizer using Reachability Analysis

For obtaining input bounds for all traffic participants, we require the consistent reachable sets complying with ϕ_s. Because we formulated the predicates using convex constraints and use decoupled linear system dynamics in longitudinal and lateral direction (1), we can represent reachable sets at every time step and for each traffic participant independently for both directions using a two-dimensional convex polytope $\mathcal{R}_{\xi,i}(k)$ and $\mathcal{R}_{\eta,i}(k) \subset \mathbb{R}^2$, respectively (cf. Fig. 7). Hence, the longitudinal reachable set of all traffic participants $\mathcal{R}_\xi(k) \subset \mathcal{X}^\mathrm{S}$ is represented by the Cartesian product of the sets of all traffic participants: $\mathcal{R}_\xi(k) = \mathcal{R}_{\xi,0}(k) \times \dots \times \mathcal{R}_{\xi,i}(k) \times \dots \times \mathcal{R}_{\xi,n_\mathrm{p}}(k)$ and similarly for the lateral direction. We choose polytopes

because they are closed under the set operations we require, namely, intersection, Minkowski sum, and linear maps.

The reachability analysis we present in this section computes these sets iteratively starting from the initial time step $k = 0$ using the one-step propagation (2), followed by constraining after each time step the obtained reachable set in order to comply with the specification ϕ_s. Since ϕ_s generally considers not only the states $x^\mathrm{S}(k)$ at the current time step k, but also at future points in time $[k{+}1, k_\mathrm{f}]$, we face the problem that those reachable sets are not yet known. To resolve this conflict, we present a forward-backward approach, which first propagates reachable sets forward in time considering only compliance within $[0, k]$, and subsequently, we backpropagate the reachable sets starting at the final time k_f to incorporate specifications from the time interval $[k{+}1, k_\mathrm{f}]$.

In contrast to previous work [62], we simultaneously consider specifications for reachable sets of a set of traffic participants, and subsequently, we directly sample a variety of trajectories within the reachable sets instead of solving an optimal control problem [63]. The algorithm of our reachability-based constraint synthesizer integrated with the scenario sampling strategy is summarized in Algorithm 3, which we explain in the remainder of this section.

A. Considering Scene Predicates as Constraints

Let us assume that the reachable sets $\overline{\mathcal{R}}_{\xi,i}(k)$ with $i \in [1, n_\mathrm{p}]$ comply with the predicates $\forall \kappa \in [0, k]$. After the one-step propagation

$$\tilde{\mathcal{R}}_{\xi,i}(k{+}1) = \vec{\mathsf{reach}}\big(\overline{\mathcal{R}}_{\xi,i}(k)\big)$$

we obtain reachable sets $\tilde{\mathcal{R}}_{\xi,i}(k{+}1)$ that do not yet consider the predicates $\mathcal{P}_q$ at time $k{+}1$ and which we therefore constrain as shown next. Since we formulated the predicates in $\mathcal{P}_q$ using linear inequalities, which we summarize as $Kx^\mathrm{S} \leq \delta$ with $K \in \mathbb{R}^{n \times z}$ and $\delta \in \mathbb{R}^n$, $\mathcal{P}_q$ defines a polyhedron $\mathcal{I}_q \subseteq \mathcal{X}^\mathrm{S}$ for each scene so that complying with all predicates in $\mathcal{P}_q$ at time $k{+}1$ is ensured by computing the intersection

$$\overline{\mathcal{R}}_{\xi}(k{+}1) = \tilde{\mathcal{R}}_{\xi}(k{+}1) \cap \mathcal{I}_q. \tag{3}$$

However, since this operation needs to be performed in the combined state space $\mathcal{X}^\mathrm{S}$ of all traffic participants, the computation time becomes unmanageable for complex scenarios due to the unfavorable complexity of polytope intersections with respect to the number of dimensions [64]. To avoid this issue, we compute the intersection over-approximately and separately for each traffic participant by tightening the bounds of the set

$$\mathcal{R}_{\xi,i}(k{+}1) \;=\; \tilde{\mathcal{R}}_{\xi,i}(k{+}1) \cap \mathsf{bound}(\tilde{\mathcal{R}}_{\xi}(k{+}1), \mathcal{I}_q, i) \tag{4}$$

using the interval obtained by the operator $\mathsf{bound}(\tilde{\mathcal{R}}_{\xi}(k), \mathcal{I}_q, i)$, which yields the over-approximation $\mathcal{R}_{\xi,i}(k{+}1) \supseteq \overline{\mathcal{R}}_{\xi,i}(k{+}1)$. The $\mathsf{bound}()$ operator is defined by the optimization problems

$$\begin{aligned} \mathsf{bound}(\tilde{\mathcal{R}}_{\xi}(k), \mathcal{I}_q, i) \coloneqq \Big[\min x_{\xi,i}, \quad \max x_{\xi,i} \Big] \qquad (5) \\ \text{subject to} \quad \underline{x}^\mathrm{S} \leq x^\mathrm{S} \leq \overline{x}^\mathrm{S} \; (\Leftrightarrow x^S \in \mathsf{box}(\tilde{\mathcal{R}}_{\xi}(k))) \\ Kx^\mathrm{S} \leq \delta \qquad (\Leftrightarrow x^\mathrm{S} \in \mathcal{I}_q). \end{aligned}$$

The optimization problems in (5) can be solved using linear programming. As a trade-off for the increased efficiency, the over-approximative intersection can later result in sampling infeasible trajectories, decreasing the efficiency of the scenario sampling process, as explained in Section VI-B.

Because of the repeated bound tightening, we choose the vertex representation for polytopes because compared to the halfspace representation, where redundant constraints need to be eliminated, the intersections directly result in a decrease of the representation size, i.e., the number of vertices, and can still be computed efficiently in the two-dimensional state space. For the same reason, we do not use other set representations, such as constrained zonotopes [65], where a reduction of the representation size is computationally more expensive [66] for this low-dimensional setting.

B. Scene Durations

For each scene S_q, only the bounds $[\underline{\delta}_q, \overline{\delta}_q]$ of its duration are provided, but we require to know the active scene at each time step to apply the corresponding constraints. Hence, in Algorithm 3, l. 2–6, we start by selecting the durations δ_q. When selecting δ_q, we need to consider that for a transition to the subsequent scene S_{q+1}, the predicates $\mathcal{P}_{q+1}$ need to be fulfilled, which we check using the condition

$$\mathsf{bound}(\mathcal{R}_{\xi}(k), \mathcal{I}_{q+1}, i) \neq \emptyset. \tag{6}$$

Thus, we first need to determine the admissible duration interval $[\underline{\delta}_q^*, \overline{\delta}_q^*] \subseteq [\underline{\delta}_q, \overline{\delta}_q]$ in which (6) holds. As summarized in Algorithm 1, we obtain $[\underline{\delta}_q^*, \overline{\delta}_q^*]$ as follows: We propagate $\mathcal{R}_{\xi}(k)$ while checking condition (6) until the transition is not possible anymore or $\overline{\delta}_q$ is reached. The admissible range $[\underline{\delta}_q^*, \overline{\delta}_q^*]$ is provided to the scenario sampling strategy, which selects $\delta_q \in [\underline{\delta}_q^*, \overline{\delta}_q^*]$. This is continued successively for each scene yielding the reachable sets $\mathcal{R}_{\xi,i}$ for the time interval $[0, k_\mathrm{f}]$.

C. Backpropagation

To directly sample trajectories within the reachabe sets (cf. Algorithm 3, l. 12–27), we require consistency with future time steps to avoid sampling states $x_{\xi,i}(k)$ for which no compliant succeeding state exists in the interval $[k{+}1, k_\mathrm{f}]$. However, after the forward propagation, consistency is not ensured due to the bound tightening (4) after each propagation step, which removes inconsistent subsets of $\mathcal{R}_{\xi}(k)$ over-approximatively. We improve the consistency of preceding time steps by removing inconsistent states using backward reachability analysis: By computing backward in time the set of all states from which it is possible to reach a set $\mathcal{R}_{\xi}(k)$ [67], we are able to remove states from which $\mathcal{R}_{\xi}(k)$ is unreachable. A similar principle was previously used to refine the cyclic invariant sets of oscillating systems [68] or the collision-free reachable sets of autonomous vehicles [69]. The backward reachable set

$$\mathcal{R}^*_{\xi,i}(k) = A^{-1}(\mathcal{R}_{\xi,i}(k{+}1) \oplus (-1)b\mathcal{U}_{\xi}) \tag{7}$$

follows from solving (2) for $\mathcal{R}_{\xi,i}(k)$ [70]. Inverting A is always possible because the time discretization of the continuous system dynamics involves a matrix exponential to

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2022.3191179

Algorithm 1 ScenePropagation: propagate reachable sets of a scene and determine admissible scene duration.

Input: Reachable set $\mathcal{R}_{\xi,i}(\tau_q)$ of n_p traffic participants, scene specifications S_q with duration interval $[\underline{\delta}_q, \overline{\delta}_q]$ and S_{q+1}.

Output: Admissible scene duration interval $[\underline{\delta}_q^*, \overline{\delta}_q^*]$, Reachable sets $\mathcal{R}_{\xi,i}(\kappa)$ with $\kappa \in [\tau_q, \overline{\delta}_q^*]$.

$\underline{\delta}_q^* \leftarrow \emptyset$
$\overline{\delta}_q^* \leftarrow \emptyset$
$k \leftarrow \tau_q$
while $\tilde{\mathcal{R}}_\xi(k) \neq \emptyset \wedge k \leq \tau_q + \overline{\delta}_q$ **do**
  for $i = 1$ to n_p **do**
    $\tilde{\mathcal{R}}_{\xi,i}(k{+}1) \leftarrow \overrightarrow{\mathsf{reach}}\big(\mathcal{R}_{\xi,i}(k)\big)$
  end for
  $\mathcal{R}_\xi(k) \leftarrow$ TightenBounds$(\tilde{\mathcal{R}}_\xi(k), S_q)$ ▷ cf. Section V-A
  if $k - \tau_q \geq \underline{\delta}_q$ **then**
    `feasible` $\leftarrow$ CheckTransition$(\mathcal{R}_\xi([\tau_q, k]))$ ▷ (6)
    if `feasible` **then**
      if $\underline{\delta}_q^* = \emptyset$ **then**
        $\underline{\delta}_q^* = k - \tau_q$
      end if
    else if $\underline{\delta}_q^* \neq \emptyset$ **then**
      $\overline{\delta}_q^* = k - \tau_q$
      break
    end if
  end if
  $k \leftarrow k{+}1$
end while
return $[\underline{\delta}_q^*, \overline{\delta}_q^*], \mathcal{R}_\xi([\tau_q, \overline{\delta}_q^*])$

compute A, which is always invertible [71, Ch. 7.2, Thm. 2]. Since the backpropagated sets $\mathcal{R}^*_{\xi,i}(k)$ contain states that are not forward reachable from $\mathcal{R}_{\xi,i}(k{-}1)$, we intersect the backpropagated set with $\mathcal{R}_{\xi,i}(k)$, which yields the updated sets $\hat{\mathcal{R}}_{\xi,i}(k), \forall i \in [1, n_\mathrm{p}]$:

$$\begin{aligned}\hat{\mathcal{R}}_{\xi,i}(k) &= \overleftarrow{\mathsf{reach}}\big(\mathcal{R}_{\xi,i}(k), \hat{\mathcal{R}}_{\xi,i}(k{+}1)\big) := \mathcal{R}_{\xi,i}(k) \cap \mathcal{R}^*_{\xi,i}(k) \\ &= \mathcal{R}_{\xi,i}(k) \cap A^{-1}\big(\hat{\mathcal{R}}_{\xi,i}(k{+}1) \oplus (-1)b\mathcal{U}_\xi\big). \end{aligned} \quad (8)$$

After iterating backward from the final time step k_f, the updated sets are tighter. The forward-backward propagation is illustrated for an example in Fig. 7 including the intermediately computed sets.

D. Obtaining Input Constraints

To sample concrete trajectories for all traffic participants, we iteratively sample states $x_{\xi,i}(k{+}1)$ within the subsequent reachable set $\mathcal{R}_{\xi,i}(k{+}1)$ given the state $x_{\xi,i}(k)$ sampled at the previous time step. Instead of directly sampling from $\mathcal{R}_{\xi,i}(k_0{+}1)$, we translate this state constraint to an interval constraint $[\underline{u}_{\xi,i}(k), \overline{u}_{\xi,i}(k)] \subseteq [u_{\min,\xi}, u_{\max,\xi}]$ for $u_{\xi,i}(k)$, since the interval representation can be processed more naturally by sampling methods.

Due to the piece-wise constant inputs and discretized system dynamics, the set reachable from $x_{\xi,i}(k)$ after one time step is a line segment parametrized by the scalar input $u_{\xi,i}(k)$ as

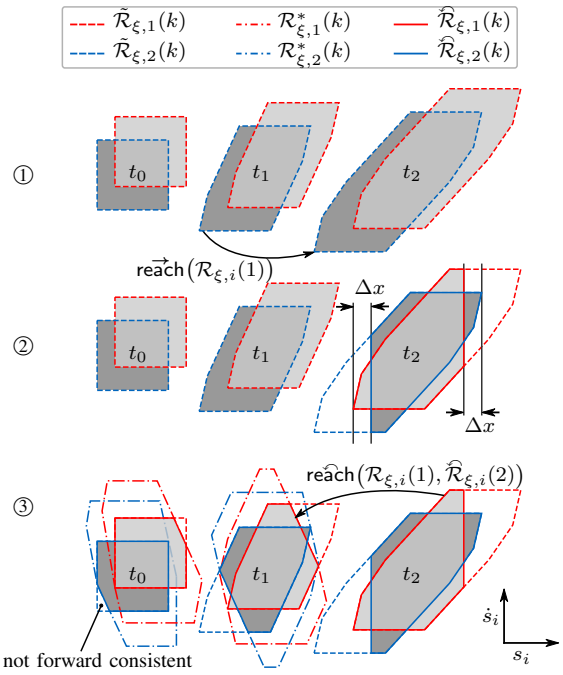


Figure 7. Main steps of the reachability analysis. ①: Forward propagation of reachable sets for two vehicles V_1 and V_2 yields sets $\tilde{\mathcal{R}}_\xi(k)$, which are not compliant. ②: Tightening the bounds at t_2 using the predicate $\texttt{Behind}(V_1, V_2, [\Delta x, \infty])$. ③: Backpropagation until t_0 yielding $\hat{\mathcal{R}}_\xi(k)$, where we remove states that are not forward consistent. For illustrative purposes, we did not tighten bounds at time steps t_0 and t_1.

illustrated in Fig. 8. Using the endpoint $\underline{x}^{\mathcal{R}}_{\xi,i} = [\underline{s}^{\mathcal{R}}, \underline{\dot{s}}^{\mathcal{R}}]^T$ of the line segment and the velocity $\dot{s}_i(k)$, we compute the lower bound $\underline{u}_{\xi,i}(k)$ as

$$\underline{u}_{\xi,i}(k) = \big(\underline{\dot{s}}^{\mathcal{R}} - \dot{s}_i(k)\big)/\Delta t. \quad (9)$$

The upper bound $\overline{u}_{\xi,i}(k)$ is computed correspondingly using $\overline{x}^{\mathcal{R}}_{\xi,i}$.

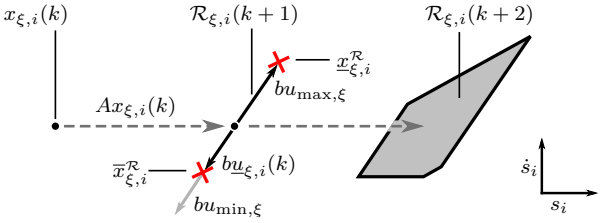


Figure 8. Obtaining input constraints for the time step t_k given the previously sampled state $x_{\xi,i}(k)$ using the vertices of $\mathcal{R}_{\xi,i}(k+1)$.

E. Updating Reachable Sets

After a new state $x_{\xi,j}(k_0) \in \mathcal{R}_{\xi,j}(k_0)$ was sampled starting at the time k of the current sample, we also update future reachable sets $\mathcal{R}_{\xi,j}([k_0, k_\mathrm{f}])$ iteratively using the following repropagation based on the same principle as the backpropagation. Due to the over-approximative bound() operator,

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2022.3191179

the propagation of $\widehat{\mathcal{R}}_{\xi,i}(k)$ is in general not enclosed by $\mathcal{R}_{\xi,i}(k{+}1)$. Thus, we compute the intersection:

$$\begin{aligned}\widehat{\mathcal{R}}_{\xi,i}(k+1) =& \widehat{\mathsf{reach}}\big(\widehat{\mathcal{R}}_{\xi,i}(k), \mathcal{R}_{\xi,i}(k{+}1)\big)\\ :=& \mathcal{R}_{\xi,i}(k{+}1) \cap \big(A\widehat{\mathcal{R}}_{\xi,i}(k) \oplus b\mathcal{U}\big). \end{aligned} \tag{10}$$

The update procedure is summarized in Algorithm 2. In the algorithm, we use the operator $\mathsf{Scene}(k)$ returning the active scene at time k. After performing the forward propagation, we backpropagate the reachable sets to obtain a forward consistent set for the next iteration. By repeating the sampling of inputs iteratively for every traffic participant and time step, we obtain longitudinal trajectories for all traffic participants as we demonstrate in the example in Fig. 5.

Algorithm 2 UpdateReachSets: Adapt reachable sets to a newly sampled state.

Input: Sampled state $x_{\xi,j}(k_0) \in \widehat{\mathcal{R}}_{\xi,j}(k_0)$, reachable sets $\widehat{\mathcal{R}}_{\xi}([k_0, k_\mathrm{f}])$ of n_p traffic participants.
Output: Updated reachable sets $\widehat{\mathcal{R}}_{\xi}(\kappa)$.

1: $\tilde{\mathcal{R}}_{\xi,j}(k_0) \leftarrow \{x_{\xi,j}(k_0)\}$
2: $\mathcal{R}_{\xi}(k_0) \leftarrow$ TIGHTENBOUNDS($\tilde{\mathcal{R}}_{\xi}(k_0), \mathsf{Scene}(k_0)$) ▷ cf. Section V-A
3: $\widehat{\mathcal{R}}_{\xi}([k_0, k_\mathrm{f}]) \leftarrow$REPROPAGATE($\mathcal{R}_{\xi}([k_0, k_\mathrm{f}])$)
4: $\widehat{\mathcal{R}}_{\xi}([k_0, k_\mathrm{f}]) \leftarrow$ BACKPROPAGATE($\widehat{\mathcal{R}}_{\xi}([k_0, k_\mathrm{f}])$)
5: **return** $\widehat{\mathcal{R}}_{\xi}([k_0, k_\mathrm{f}])$

6: **function** REPROPAGATE($\mathcal{R}_{\xi}([k_0, k_\mathrm{f}])$)
7: $\quad\widehat{\mathcal{R}}_{\xi,i}(k_0) \leftarrow \mathcal{R}_{\xi}(k_0)$
8: $\quad$**for** $k = k_0$ to $k_\mathrm{f}{-}1$ **do**
9: $\quad\quad$**for** $i = 1$ to n_p **do**
10: $\quad\quad\quad\widehat{\mathcal{R}}_{\xi,i}(k{+}1) \leftarrow \widehat{\mathsf{reach}}\big(\widehat{\mathcal{R}}_{\xi,i}(k), \mathcal{R}_{\xi,i}(k{+}1)\big)$ ▷ cf. (10)
11: $\quad\quad\quad\widehat{\mathcal{R}}_{\xi}(k+1) \leftarrow$ TIGHTENBOUNDS($\widehat{\mathcal{R}}_{\xi}(k+1), \mathsf{Scene}(k+1)$) ▷ cf. Section V-A
12: $\quad\quad$**end for**
13: $\quad$**end for**
14: $\quad$**return** $\widehat{\mathcal{R}}_{\xi,i}([k_0, k_\mathrm{f}])$
15: **end function**

16: **function** BACKPROPAGATE($\widehat{\mathcal{R}}_{\xi}([k_0, k_\mathrm{f}])$)
17: $\quad\widehat{\mathcal{R}}_{\xi}(k_\mathrm{f}) \leftarrow \widehat{\mathcal{R}}_{\xi}(k_\mathrm{f})$
18: $\quad$**for** $k = k_\mathrm{f}{-}1$ to k_0 **do**
19: $\quad\quad$**for** $i = 1$ to n_p **do**
20: $\quad\quad\quad\widehat{\mathcal{R}}_{\xi,i}(k) \leftarrow \widehat{\mathsf{reach}}\big(\mathcal{R}_{\xi,i}(k), \widehat{\mathcal{R}}_{\xi,i}(k{+}1)\big)$ ▷ cf. (8)
21: $\quad\quad\quad\widehat{\mathcal{R}}_{\xi}(k) \leftarrow$ TIGHTENBOUNDS($\widehat{\mathcal{R}}_{\xi}(k), \mathsf{Scene}(k)$) ▷ cf. Section V-A
22: $\quad\quad$**end for**
23: $\quad$**end for**
24: $\quad$**return** $\widehat{\mathcal{R}}_{\xi,i}([k_0, k_\mathrm{f}])$
25: **end function**

F. Lateral Trajectories

After the longitudinal trajectories, the lateral trajectories can be sampled using the same approach but with constraints for the lateral position bounds at each position $[\underline{d}_i(s_i(k)), \overline{d}_i(s_i(k))]$, e.g., derived from the `InLanelets()` constraint introduced in Definition 7. However, the reachable sets need to be tightened less frequently because the constraints are usually static, i.e., they do not depend on lateral positions of other traffic participants. Therefore, we use optimal control algorithms to plan the lateral trajectory fulfilling the scene constraints $\mathcal{P}_q$ as in our previous work [51].

Table I
OVERVIEW OF NOTATIONS USED FOR REACHABLE SETS.

Set	Description
$\overline{\mathcal{R}}_k$	Consistent reachable set.
$\mathcal{R}_k$	Over-approximative reachable set.
$\tilde{\mathcal{R}}_k$	Propagated reachable set before applying constraints.
$\mathcal{R}^*_k$	Backward reachable set.
$\widehat{\mathcal{R}}_k$	Forward-backward propagated reachable set.
$\widehat{\mathcal{R}}_k$	Repropagated reachable set.

VI. FALSIFICATION USING MONTE-CARLO TREE SEARCH

Algorithm 3 summarizes sampling of a concrete scenario. In l. 4, 9, and 18, the algorithm requires the SCENARIOSAMPLINGSTRATEGY($[\underline{\alpha}, \overline{\alpha}]$) to select concrete parameters within the intervals $[\underline{\alpha}, \overline{\alpha}]$, where the variable α is one of $\delta_q, x_{\xi,i}(0), u_{\xi,i}(k)$, and $u_{\eta,i}(k)$. Subsequently, we use Monte Carlo tree search as a scenario sampling strategy in a closed loop with a SUT to find concrete scenarios complying with ϕ.

A. Scenario Sampling Using Monte-Carlo Tree Search

We denote calls to SCENARIOSAMPLINGSTRATEGY($[\underline{\alpha}, \overline{\alpha}]$) as sampling requests, which we use to build the search tree for the Monte Carlo tree search on the fly while executing Algorithm 3. Hence, each sample request provides the constrained action space as intervals $\mathcal{A}'(\sigma) = [\underline{\alpha}, \overline{\alpha}]$ and the states σ represent the state in the algorithm after applying the action. Due to the order of the requests in the algorithm, the nodes are structured hierarchically with the actions of the falsifier being selected in the order: $\delta_q \xrightarrow{\forall q \in [1,m]} x_{\xi,i}(0) \xrightarrow{\forall i \in [1,n_\mathrm{p}]} u_{\xi,i}(k) \xrightarrow{\forall k \in [0,k_\mathrm{f}],\, i \in [1,n_\mathrm{p}]} u_{\eta,i}(k)$. Since the action bounds are computed individually through reachability analysis for each state, we reduce the branching factor of the search tree and do not rely on a fine discretization to synthesize complex scenarios with possibly tight passages in the feasible action space. To deal with the continuous actions space, we use progressive widening [72], where the number of child nodes added during the expansion step depends on the number of visits of the parent node.

For the Q-function in the tree policy $\pi_\mathrm{T}(\sigma, \mathcal{A}'(\sigma))$, we use continuous rapid action value estimates (Appendix A) [73] that take into account rewards of previously visited subtrees weighted by the similarity of their state-actions pairs. As the playout policy $\pi_\mathrm{P}(\sigma)$, we use a uniform random sampler which is a standard approach [59] and ensures unbiased exploration of the solution space of the abstract scenario.

B. Reward Function

In Algorithm 3, we treat the SUT like any other traffic participant, only the selection of inputs $u_{\xi,\mathrm{ego}}$ is handled by the

Algorithm 3 Iterative sampling scheme for a specification-compliant scenario.

Input: Scenario specification ϕ_s for n_p traffic participants, a function ScenarioSamplingStrategy(), initial reachable sets $\mathcal{R}_\xi(0)$, time horizon k_f, and system under test $M_\mathrm{SUT}(x^\mathrm{S}(k))$.

Output: Compliant time horizon k_con, trajectories $x^\mathrm{S}([0,k_\mathrm{con}])$ for all traffic participants.

▷ Obtain scene durations and initialize reachable sets over the whole time horizon:

1: $\tau_1 \leftarrow 0$
2: **for** $q = 1$ to m **do**
3: $\quad[\underline{\delta}_q^*, \overline{\delta}_q^*], \mathcal{R}_\xi([\tau_q, \overline{\delta}_q^*]) \leftarrow$ ScenePropagation($\mathcal{R}_\xi(\tau_q)$, S_q) ▷ cf. Algorithm 1
4: $\quad\delta_q \leftarrow$ ScenarioSamplingStrategy($[\underline{\delta}_q^*, \overline{\delta}_q^*]$) ▷ cf. Section VI-A
5: $\quad\tau_{q+1} \leftarrow \tau_q + \delta_q$
6: **end for**
7: $\widehat{\mathcal{R}}_\xi([0,k_\mathrm{f}]) \leftarrow$ BackPropagate($\mathcal{R}_\xi([0,k_\mathrm{f}])$) ▷ cf. Algorithm 2, l. 16

▷ Determine initial states:

8: **for** $i = 1$ to n_p **do**
9: $\quad x_{\xi,i}(0) \leftarrow$ ScenarioSamplingStrategy($\widehat{\mathcal{R}}_{\xi,i}(0)$) ▷ cf. Section VI-A
10: $\quad\widehat{\mathcal{R}}_\xi([0,k_\mathrm{f}]) \leftarrow$UpdateReachSets($x_{\xi,i}(0), \widehat{\mathcal{R}}_\xi([0,k_\mathrm{f}])$) ▷ cf. Algorithm 2
11: **end for**

▷ Determine longitudinal trajectories:

12: **for** $k = 0$ to $k_\mathrm{f}-1$ **do**
13: $\quad$**for** $i = 1$ to n_p **do**
14: $\quad\quad$**if** $i = \mathrm{ego}$ **then**
15: $\quad\quad\quad u_{\xi,\mathrm{ego}}(k) \leftarrow M_\mathrm{SUT}(x^\mathrm{S}(k))$
16: $\quad\quad$**else**
17: $\quad\quad\quad[\underline{u}_{\xi,i}(k), \overline{u}_{\xi,i}(k)] \leftarrow$ InputBounds($x_{\xi,i}(k), \widehat{\mathcal{R}}_{\xi,i}(k+1)$) ▷ cf. Section V-D
18: $\quad\quad\quad u_{\xi,i}(k) \leftarrow$ ScenarioSamplingStrategy($[\underline{u}_{\xi,i}(k), \overline{u}_{\xi,i}(k)]$) ▷ cf. Section VI-A
19: $\quad\quad\quad x_{\xi,i}(k+1) \leftarrow$ VehicleDynamics($x_{\xi,i}(k), u_{\xi,i}(k)$) ▷ cf. Eq. (1)
20: $\quad\quad$**end if**
21: $\quad\quad\widehat{\mathcal{R}}_\xi([k,k_\mathrm{f}]) \leftarrow$UpdateReachSets($x_{\xi,i}(k), \widehat{\mathcal{R}}_\xi([k,k_\mathrm{f}])$) ▷ cf. Algorithm 2
22: $\quad\quad$**if** $\widehat{\mathcal{R}}_\xi(k+1) = \emptyset$ **then**
23: $\quad\quad\quad k_\mathrm{con} \leftarrow k$
24: $\quad\quad\quad$**return** k_con, $x^\mathrm{S}([0,k])$ ▷ Return early if incompliant with ϕ_s, cf. Section VI-B
25: $\quad\quad$**end if**
26: $\quad$**end for**
27: **end for**
28: $x_\eta([0,k_\mathrm{f}]) \leftarrow$ComputeLateralTrajectories($x_\xi([0,k_\mathrm{f}])$) ▷ cf. Section V-F
29: **return** k_f, $x^\mathrm{S}([0,k_\mathrm{f}])$

SUT instead of the Monte Carlo tree search. However, when generating scenarios in a closed loop with a black-box SUT, it is not ensured that its behavior always complies with the scenario specification. For instance, when we are interested in scenarios where the SUT overtakes a leading vehicle, the SUT might continue following the leader depending on the behavior of other traffic participants. Thus, one main objective of the search problem is to find scenario concretizations that induce the specified behavior of the SUT. To this end, we check in Algorithm 3, l. 24 for undesired early terminations caused by incompliant inputs of the SUT that result in empty reachable sets $\mathcal{R}_\xi$. Additionally, incompliant trajectories can result from over-approximations of the bound tightening in (4). To guide the search towards compliant concretizations, we reward specification-compliant behavior of the SUT by including the number of compliant time steps $k_\mathrm{con}(\sigma)$ normalized by the scenario duration k_f in the reward $R(\sigma)$ for the Monte Carlo tree search. In addition, a metric $R_\mathrm{c}(\sigma) \in \mathbb{R}^+$ assessing the criticality of the concretized scenario is included to guide the search towards safety-critical scenarios

$$R(\sigma) = \frac{k_\mathrm{con}(\sigma)}{k_\mathrm{f}} + w_\mathrm{c} R_\mathrm{c}(\sigma) \tag{11}$$

using a weight $w_\mathrm{c} \in \mathbb{R}^+$ for balancing the criticality with specification compliance.

VII. Numerical Experiments

We demonstrate the usefulness of our method by applying it to a SUT comprising a motion planning and a prediction module in several traffic scenarios. Parameters for all methods are listed in Table II.

A. System Under Test

We use a motion planner based on Werling et al. [74] that samples trajectory candidates using polynomials for the longitudinal and lateral motion with respect to a curvilinear coordinate system and selects the best collision-free candidate

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2022.3191179

using a cost function. Each sample is associated with a target velocity and a lateral target position within a sampling interval relative to the initial position of the planning cycle. For each scenario, we set a target lane through a desired lateral position $d_{\mathrm{D}} \in \mathbb{R}$. Planning is repeated cyclically after durations of $h_{\mathrm{r}} \in \mathbb{R}^{+}$.

Predictions of other traffic participants required for collision avoidance are provided by a prediction module for a horizon $h_{\mathrm{p}} \geq h_{\mathrm{r}}$ starting at the initial time t_0 of the planning cycle. The predictions are computed assuming a constant velocity model in the lateral direction and for the longitudinal direction, position intervals are used as a prediction based on assumed bounds of the longitudinal acceleration $u_{\xi,i} \in [u^*_{\xi,\min}, u^*_{\xi,\max}]$ and considering $\dot{s}_{\xi,i} \in [0, v_{\max}]$.

B. Reward Function

As the criticality metric in the reward function we use a metric based on time-to-collision t_{ttc}, since we evaluate our approach on highway scenarios: $R_{\mathrm{c}}(\sigma) = \max(1 - \frac{t_{\mathrm{ttc}}(\sigma)}{k_{\mathrm{f}} \Delta t}, 0)$ where 1 denotes the most critical value.

Table II
PARAMETERS USED IN THE EXPERIMENTS.

Motion Planner (SUT)		
planning horizon	h_{p}	3.5 s
replanning cycle	h_{r}	0.175 s
lateral sampling interval		$[-3, 3]$ m
velocity sampling interval		$[0, 25]$ m/s
min. lateral discretization	Δd	0.66 m
Constraint Synthesizer		
step size	Δt	0.35 s
time horizon	t_{f}	10 s
Monte Carlo Tree Search		
criticality weight	w_{c}	1.5
acceleration bounds (prediction)	$[u^*_{\xi,\min}, u^*_{\xi,\max}]$	$[-1.0, 1.0]$

C. Scenario Specifications

We evaluate our approach in total using seven different highway scenarios comprising lane changes, evasive maneuvers, and onramp scenarios. For an overview of all scenarios, see Appendix B. Additionally, we provide the sampled scenarios on commonroad.in.tum.de under the scenario IDs ZAM_Ramp_2-X. Next, we explain two scenarios in more detail.

1) Scenario A - Onramp scenario: In the first scenario, a vehicle merges from the onramp between the ego vehicle and a leading vehicle. We encode in the scenario specification ϕ_{s} that we are interested in failures, in which the SUT crashes into the leading vehicle after it has merged instead of braking or evading to the left lane. Fig. 9 visualizes the four scenes of this specification.

2) Scenario B - SUT evading a merging cut-in vehicle: In this scenario, the ego vehicle is keeping its lane and, when in the second scene, a vehicle is merging in its lane from the left and forces the ego vehicle to swerve to the left lane to avoid a collision. We are interested in failure cases where the SUT crashes into the leading vehicle in the left lane. The four scenes of the specification are visualized in Fig. 10.

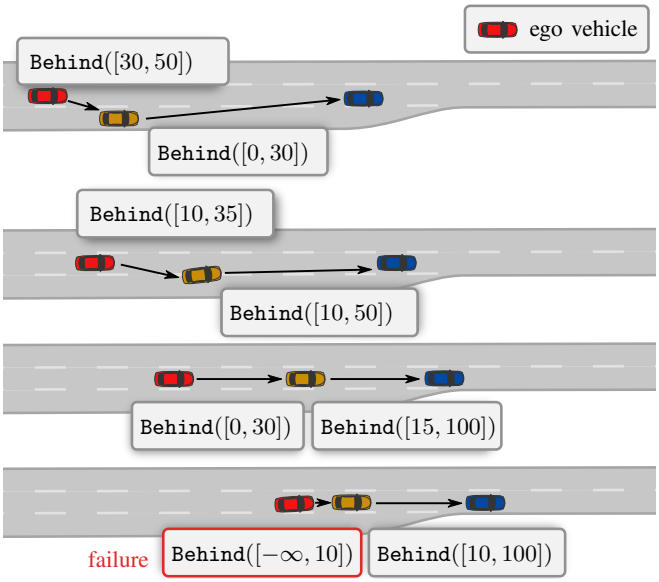


Figure 9. Specified scenes visualized for scenario A. Velocity and lane constraints are not shown explicitly.

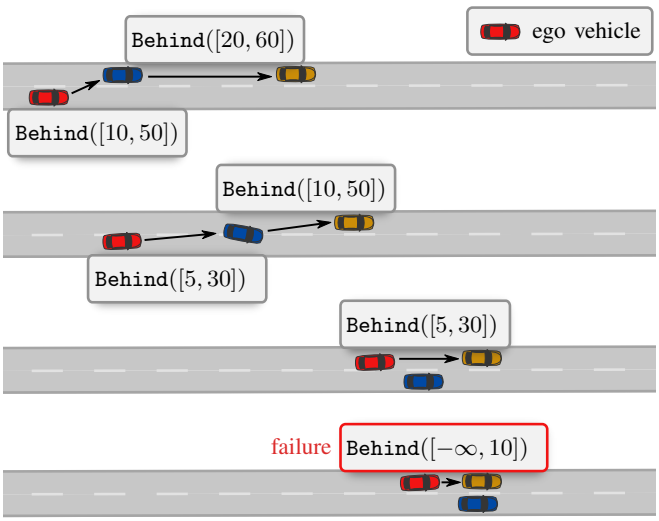


Figure 10. Specified scenes for scenario B. Velocity and lane constraints are not shown explicitly.

D. Evaluation

To account for the random nature of Monte Carlo tree search, we repeat all experiments with 18 different random seeds. The runtime of the experiments is measured on an AMD EPYC 7742 CPU with 2.25 GHz with one core assigned to each experiment. The code is implemented primarily in Python and geometric operations of the reachability analysis are implemented in C++.

We only count experiments as successful falsifications if these are found within 45 minutes of computation time. In Table III, we summarize the number of successful falsifications

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2022.3191179

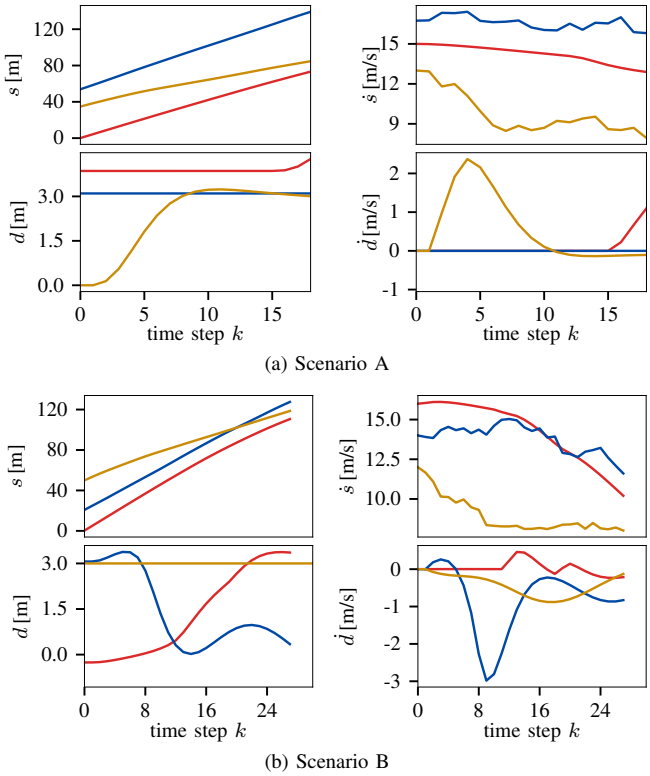
out of the 18 repetitions for different values of the exploration weight c used in (12). Furthermore, the average computation times for finding a failure are listed, and we provide the cumulative number of time steps that were simulated during the playouts. The reachability analysis and the Monte Carlo tree search require on average $47\,\%$ and the SUT $53\,\%$ of the total computation time. Upon manually inspecting the failures of the SUT, we observe that they are primarily caused by incorrect predictions and an insufficient number of trajectory candidates that the motion planner can choose from. To give examples of falsifying scenarios, we show in Fig. 11 the trajectories of the SUT and the other traffic participants sampled during the search of scenario A and scenario B.

E. Ablation Study

We perform an ablation study to evaluate the effect of the reachability-based constraint synthesizer. We compare the previously presented results to a baseline falsifier that does not perform backward reachability analysis in the constraint synthesis and hence, does not exclude actions that will inevitably result in incompliant behavior at succeeding time steps. We show the number of successful falsifications in Table IV of the baseline approach next to the results from Table III for $c = 1.5$. The results show that with the constraint synthesis, the falsifier finds failures more reliably than the baseline approach for all scenarios. In fact, the baseline approach finds failures only in a minority of the experiments. This indicates that the backward reachability analysis significantly affects the sampling efficiency, since otherwise, the sampled scenarios quickly become incompliant due to earlier actions.

Table III
RESULTS OF THE FALSIFICATION EXPERIMENTS: Monte Carlo tree search with different exploration weights c.

Scenario	c	Falsifications (18 trials)	Simulation steps Mean	Simulation steps Std. dev.	Mean time [min]
A	1.2	13	2993	3777	26.0
	1.5	16	1056	804	24.3
	1.8	12	1551	1382	28.3
B	1.2	17	596	872	34.7
	1.5	17	667	722	38.0
	1.8	18	655	734	22.7
C	1.2	18	56	43	4.9
	1.5	18	57	49	4.5
	1.8	18	54	41	4.6
D	1.2	5	1932	1244	32.0
	1.5	7	1278	1240	29.5
	1.8	7	2026	2451	32.7
E	1.2	12	3272	3402	28.9
	1.5	15	5671	3556	30.9
	1.8	17	7568	4068	39.8
F	1.2	9	1238	1110	36.4
	1.5	9	998	1029	36.2
	1.8	12	1379	1664	36.2
G	1.2	18	1056	2044	15.6
	1.5	18	892	732	7.6
	1.8	18	991	692	9.1



(a) Scenario A

(b) Scenario B

Figure 11. Falsifying trajectories in the longitudinal and lateral direction of the SUT (red) and the other traffic participants. Colors match Figs. 9 and 10.

Table IV
COMPARING THE FALSIFICATION RESULTS FROM 18 TRIALS TO A BASLINE WITHOUT BACKWARD REACHABILITY ANALYSIS FOR $c = 1.5$.

Scenario	A	B	C	D	E	F	G
presented approach	16	17	18	7	15	9	18
baseline	2	5	0	0	3	4	1

F. Efficiency Analysis

Next, we analyze how the computation time of the scenario synthesis relates to the difficulty of the scenario specification. To quantify the difficulty of a scenario specification, we use the length $l_u = \overline{u}_{\xi,i}(k) - \underline{u}_{\xi,i}(k)$ of the input interval (9), which is computed by our scenario synthesis. Thus, l_u represents the size of the feasible action space, and we assume that a small action space indicates that it is more challenging to fulfill the scenario specification.

To analyze l_u on many scenario specifications, we randomly change the scenarios from our previous evaluation by scaling the intervals of the predicate `behind` using randomly sampled factors $\gamma \in [0.7, 1.3]$. This way we create 20 random variations of each scenario specification. For each of these specifications, we run our scenario synthesizer for 200 simulation steps and measure l_u and the computation time for each step. The averaged values for each run are plotted in Fig. 12. The linear regression shows that the computation time for our approach decreases for smaller values of l_u, i.e., becomes smaller for challenging scenario specifications. This can be explained by the geometrical operations of the reachability analysis that

become faster when the number of vertices of the polytopes decreases.

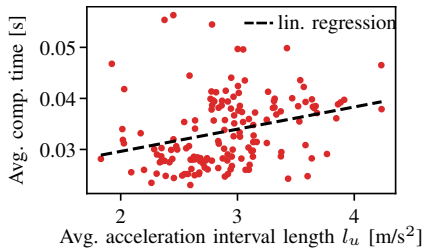


Figure 12. Relation between the computation time per simulation step and the length of l_u: the computation time decreases for challenging scenarios for smaller l_u.

VIII. Conclusions

We have presented an approach to synthesize concrete scenario instances from abstractly specified scenarios for testing autonomous vehicles. By computing reachable sets, which are compliant with the specification, we are subsequently able to efficiently sample a large variety of trajectories for other traffic participants. In contrast to other approaches that might require dedicated simulator implementation or driver models to execute the scenarios, our approach only requires knowledge about the physical bounds of the traffic participants. Therefore, even unusual behavior can be synthesized and we demonstrate how this synthesizer can be used for finding failures using search-based methods. We provide the code of our implementation under https://github.com/mo-kli/FalsificationTrafficScenarios. In the future, our scenario synthesizer may be combined, e.g., with existing adversarial learning-based techniques [30]–[32] to guide their learning process towards specified scenarios.

Appendix

A. Continuous Rapid Action Value Estimates

To estimate the Q-value of a state-action pair, we use continuous rapid action value estimates [73] that improve the confidence in the estimate $Q(\sigma,\alpha)$ for nodes with a low number of visits, which is factored in by the weight $\beta(\sigma,\alpha)\in[0,1]$ and in turn depends on the number of visits:

$$Q(\sigma,\alpha)=\Big(\beta(\sigma,\alpha)Q_{\mathrm{cRAVE}}(\sigma,\alpha) + (1-\beta(\sigma,\alpha))\,\mathrm{Q}_{\mathrm{UCT}}(\sigma,\alpha)\Big).$$

For a state σ and action α the continuous rapid action value estimate is computed as defined in [73, Eq. 7],:

$$Q_{\mathrm{cRAVE}}(\sigma,\alpha)= \frac{1}{\sum_{\sigma_i\in\mathcal{S}}\sum_{\alpha_i(\sigma_i)} \mathrm{e}^{-\log N(\sigma_i,\alpha_i)\left\{\frac{d(\sigma,\sigma_i)^2}{w_\sigma}+\frac{d(\alpha,\alpha_i)^2}{w_\alpha}\right\}}} \cdot \sum_{\sigma_i\in\mathcal{S}}\sum_{\alpha_i(\sigma_i)} R(\sigma_i,\alpha_i)\ \mathrm{e}^{-\log N(\sigma_i,\alpha_i)\left\{\frac{d(\sigma,\sigma_i)^2}{w_\sigma}+\frac{d(\alpha,\alpha_i)^2}{w_\alpha}\right\}}$$

Table V
Descriptions of scenarios.

Scenario	$\mathrm{n_p}$	Description
A	3	A vehicle merges from the onramp between SUT and a leading vehicle and subsequently, the SUT crashes into the merged vehicle from behind.
B	4	Like scenario A but with a third vehicle driving in the lane left to the SUT.
C	4	The SUT merges from the onramp between two vehicles and subsequently the trailing vehicle crashes into the SUT from behind. A third vehicle drives in parallel to the SUT on the leftmost lane.
D	3	The SUT merges from the onramp between two vehicles and subsequently crashes into the leading vehicle.
E	4	The SUT changes from the leftmost lane to the middle lane while another vehicle changes from the rightmost lane to the middle lane. A third vehicle drives in front of the SUT.
F	3	The SUT drives on the middle lane when a vehicle changes from the leftmost lane to the middle lane in front of the SUT. The SUT evades to the leftmost lane where it crashes into a vehicle driving in front.
G	4	The SUT drives in the middle lane and two vehicles change to the middle lane from the left and right lane, respectively.

iterating over all states $\sigma_i\in\mathcal{S}$ using rewards $R(\sigma)$ and weights w_σ for states and w_α for actions. $N(\sigma_i,\alpha_i)$ is the number of times when α_i was selected from node σ_i. The upper confidence bounds applied to trees (UCT) is computed as

$$\mathrm{Q}_{\mathrm{UCT}}(\sigma,\alpha)=\frac{\sum_{j=1}^{N(\sigma,\alpha)}R_j(\sigma,\alpha)}{N(\sigma,\alpha)}+c\sqrt{\frac{\log(N(\sigma))}{N(\sigma,\alpha)}}, \quad (12)$$

where $N(\sigma)$ is the total number of visits of node $\sigma$and $c\in\mathbb{R}^+$ is the exploration weight.

B. Scenario Descriptions

Due to space limitations, the scenario specifications used for the experiments are summarized in Table V. The files with the complete formal specifications are provided in the repository github.com/mo-kli/FalsificationTrafficScenarios.

Acknowledgment

The authors gratefully acknowledge partial financial support from the Central Innovation Programme of the German Federal Government under grant ZF4086013GR9 and from the German Federal Ministry for Digital and Transport (BMDV) within the project Cooperative Autonomous Driving with Safety Guarantees (KoSi).

References

[1] C. Neurohr, L. Westhofen, T. Henning, T. De Graaff, E. Mohlmann, and E. Bode, "Fundamental considerations around scenario-based testing for automated driving," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2020, pp. 121–127.

[2] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer, "A survey of algorithms for black-box safety validation of cyber-physical systems," *Journal of Artificial Intelligence Research*, vol. 72, pp. 377–428, 2021.

[3] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, 2020.

[4] C. Sippl, F. Bock, D. Wittmann, H. Altinger, and R. German, "From simulation data to test cases for fully automated driving and ADAS," in *Lecture Notes in Computer Science*, 2016, pp. 191–206.

[5] C. Wolschke, T. Kuhn, D. Rombach, and P. Liggesmeyer, "Observation based creation of minimal test suites for autonomous vehicles," in *Proc. of the IEEE 28th Int. Symposium on Software Reliability Engineering Workshops*, 2017, pp. 294–301.

[6] J. Bach, J. Langner, S. Otten, E. Sax, and M. Holzapfel, "Test scenario selection for system-level verification and validation of geolocation-dependent automotive control systems," in *Proc. of the Int. Conf. on Engineering, Technology and Innovation*, 2018, pp. 203–210.

[7] F. Kruber, J. Wurst, and M. Botsch, "An unsupervised random forest clustering technique for automatic traffic scenario categorization," in *Proc. of the IEEE Conf. on Intell. Transportation Sys.*, 2018, pp. 2811–2818.

[8] D. Zhao, Y. Guo, and Y. J. Jia, "TrafficNet: An open naturalistic driving scenario library," in *Proc. of the IEEE Int. Conf. on Intell. Transportation Sys.*, 2018, pp. 1–8.

[9] F. Hauer, I. Gerostathopoulos, T. Schmidt, and A. Pretschner, "Clustering traffic scenarios using mental models as little as possible," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2020, pp. 1007–1012.

[10] M. Zipfl, T. Fleck, M. R. Zofka, and J. M. Zöllner, "From traffic sensor data to semantic traffic descriptions: The test area autonomous driving Baden-Württemberg dataset (TAF-BW Dataset)," in *Proc. of the IEEE Int. Conf. on Intell. Transportation Sys.*, 2020, pp. 133–139.

[11] T. A. Wheeler, M. J. Kochenderfer, and P. Robbel, "Initial scene configurations for highway traffic propagation," in *Proc. of the IEEE Conf. on Intell. Transportation Sys.*, 2015, pp. 279–284.

[12] S. Jesenski, J. E. Stellet, F. Schiegg, and J. M. Zollner, "Generation of scenes in intersections for the validation of highly automated driving functions," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2019, pp. 502–509.

[13] E. De Gelder and J. P. Paardekooper, "Assessment of automated driving systems using real-life scenarios," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2017, pp. 589–594.

[14] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE Trans. on Intell. Transportation Sys.*, vol. 18, no. 3, pp. 595–607, 2017.

[15] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," in *Proc. of the 32nd Int. Conf. on Neural Information Processing Systems*, 2018, pp. 9849–9860.

[16] D. Asljung, J. Nilsson, and J. Fredriksson, "Using extreme value theory for vehicle level safety validation and implications for autonomous vehicles," *IEEE Trans. on Intell. Vehicles*, vol. 2, no. 4, pp. 288–297, 2017.

[17] M. R. Zofka, F. Kuhnt, R. Kohlhaas, C. Rist, T. Schamm, and J. M. Zollner, "Data-driven simulation and parametrization of traffic scenarios for the development of advanced driver assistance systems," in *Proc. of the 18th Int. Conf. on Information Fusion*, 2015, pp. 1422–1428.

[18] S. Wagner, K. Groh, T. Kuhbeck, and A. Knoll, "Towards cross-verification and use of simulation in the assessment of automated driving," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2019, pp. 1589–1596.

[19] R. Krajewski, T. Moers, D. Nerger, and L. Eckstein, "Data-driven maneuver modeling using generative adversarial networks and variational autoencoders for safety validation of highly automated vehicles," in *Proc. of the IEEE Int. Conf. on Intell. Transportation Sys.*, 2018, pp. 2383–2390.

[20] S. Shiroshita, S. Maruyama, D. Nishiyama, M. Y. Castro, K. Hamzaoui, G. Rosman, J. DeCastro, K.-H. Lee, and A. Gaidon, "Behaviorally diverse traffic simulation via reinforcement learning," in *IEEE Int. Conf. on Intell. Robots and Systems*, 2020, pp. 2103–2110.

[21] Z. Zhong, G. Kaiser, and B. Ray, "Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles," *arXiv:2109.06126*, 2021.

[22] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivačić, and A. Gupta, "Probabilistic temporal logic falsification of cyber-physical systems," *ACM Trans. on Embedded Computing Systems*, vol. 12, no. 2s, pp. 1–30, 2013.

[23] T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional Falsification of Cyber-Physical Systems with Machine Learning Components," *Journal of Automated Reasoning*, vol. 63, pp. 1031–1053, 2019.

[24] G. Ernst, P. Arcaini, A. Donze, G. Fainekos, L. Mathesen, G. Pedrielli, S. Yaghoubi, Y. Yamagata, and Z. Zhang, "ARCH-COMP 2020 category report: Falsification," *EPiC Series in Computing*, vol. 74, pp. 140–152, 2020.

[25] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2018, pp. 1555–1562.

[26] Y. Yamagata, S. Liu, T. Akazaki, Y. Duan, and J. Hao, "Falsification of Cyber-Physical Systems Using Deep Reinforcement Learning," *IEEE Trans. on Software Engineering*, vol. 47, no. 12, pp. 2823–2840, 2021.

[27] C. Gladisch, T. Heinz, C. Heinzemann, J. Oehlerking, A. von Vietinghoff, and T. Pfitzer, "Experience paper: Search-based testing in automated driving control applications," in *34th IEEE/ACM Int. Conf. on Automated Software Engineering (ASE)*, 2019, pp. 26–37.

[28] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, "Adaptive stress testing for autonomous vehicles," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2018, pp. 1898–1904.

[29] S. Feng, X. Yan, H. Sun, Y. Feng, and H. X. Liu, "Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment," *Nature Communications*, vol. 12, no. 1, pp. 1–14, 2021.

[30] W. Ding, M. Xu, and D. Zhao, "CMTS: A conditional multiple trajectory synthesizer for generating safety-critical driving scenarios," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 4314–4321.

[31] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun, "AdvSim: Generating safety-critical scenarios for self-driving vehicles," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2021, pp. 9904–9913.

[32] B. Chen, X. Chen, Q. Wu, and L. Li, "Adversarial evaluation of autonomous vehicles in lane-change scenarios," *IEEE Trans. on Intell. Transportation Sys., early access, doi: 10.1109/tits.2021.3091477*, 2021.

[33] F. Hauer, A. Pretschner, and B. Holzmüller, "Fitness functions for testing automated and autonomous driving systems," in *SAFECOMP 2019: Computer Safety, Reliability, and Security*, pp. 69–84.

[34] H. Beglerovic, M. Stolz, and M. Horn, "Testing of autonomous vehicles using surrogate models and stochastic optimization," in *Proc. of the IEEE Conf. on Intell. Transportation Sys.*, 2018, pp. 1129–1134.

[35] P. Akella, M. Ahmadi, R. M. Murray, and A. D. Ames, "Formal test synthesis for safety-critical autonomous systems based on control barrier functions," in *Proc. of the IEEE Conf. on Decision and Control*, 2020, pp. 790–795.

[36] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. Iyer, "AV-FUZZER: Finding safety violations in autonomous driving systems," in *Proc. of the Int. Symposium on Software Reliability Engineering*, 2020, pp. 25–36.

[37] C. E. Tuncali and G. Fainekos, "Rapidly-exploring random trees-based test generation for autonomous vehicles," in *Proc. of the IEEE Int. Conf. on Intell. Transportation Sys.*, 2019, pp. 661–666.

[38] M. Koschi, C. Pek, S. Maierhofer, and M. Althoff, "Computationally efficient safety falsification of adaptive cruise control systems," in *Proc. of the IEEE Int. Conf. on Intell. Transportation Sys.*, 2019, pp. 2879–2886.

[39] J. Dahl, G. R. De Campos, C. Olsson, and J. Fredriksson, "Collision avoidance: A literature review on threat-assessment techniques," *IEEE Trans. on Intell. Vehicles*, vol. 4, no. 1, pp. 101–113, 2019.

[40] L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Kramer, C. Gutenkunst, and E. Böde, "Criticality metrics for automated driving: A review and suitability analysis of the state of the art," *arXiv:2108.02403*, 2021.

[41] W. Damm, E. Möhlmann, T. Peikenkamp, and A. Rakow, "A formal semantics for traffic sequence charts," in *Lecture Notes in Computer Science*, 2018, vol. 10760, pp. 182–205.

[42] D. J. Fremont, X. Yue, T. Dreossi, A. L. Sangiovanni-Vincentelli, S. Ghosh, and S. A. Seshia, "SCENIC: A language for scenario specification and scene generation," in *Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation*, 2019, pp. 63–78.

[43] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, and D. Zufferey, "PARAMCOSM: A test framework for autonomous driving simulations," in *Proc. of the Int. Conf. on Fundamental Approaches to Software Engineering*, 2021, pp. 172–195.

[44] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia, "VerifAI: A Toolkit for the Formal Design and Analysis of Artificial Intelligence-Based Systems," in *Proc. of the Int. Conf. on Computer Aided Verification*, vol. 11561, 2019, pp. 432–442.

[45] S. Geyer, M. Baltzer, B. Franz, S. Hakuli, M. Kauer, M. Kienle, S. Meier, T. Weigerber, K. Bengler, R. Bruder, F. Flemisch, and H. Winner, "Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance," *IET Intell. Transport Systems*, vol. 8, no. 3, pp. 183–189, 2014.

[46] E. de Gelder, J. P. Paardekooper, A. K. Saberi, H. Elrofai, O. O. den Camp., J. Ploeg, L. Friedmann, and B. De Schutter, "Ontology for scenarios for the assessment of automated vehicles," *arXiv:2001.11507*, 2020.

[47] N. Kolb, F. Hauer, and A. Pretschner, "Fitness function templates for testing automated and autonomous driving systems in intersection scenarios," in *Proc. of the IEEE Int. Conf. on Intell. Transportation Sys.*, 2021, pp. 217–222.

[48] X. Qin, N. Aréchiga, A. Best, and J. Deshmukh, "Automatic testing with reusable adversarial agents," *arXiv:1910.13645*, 2019.

[49] E. de Gelder, E. Cator, J.-P. Paardekooper, O. O. den Camp, and B. De Schutter, "Constrained sampling from a kernel density estimator to generate scenarios for the assessment of automated vehicles," in *Proc. of the IEEE Intell. Vehicles Symposium, 4th Workshop on "Ensuring and Validating Safety for Automated Vehicles"*, 2021, pp. 203–208.

[50] K. Scheibler, A. Eggers, T. Teige, M. Walz, T. Bienmüller, and U. Brockmeyer, "Solving constraint systems from traffic scenarios for the validation of autonomous driving," in *Proc. of the 4th SC-square Workshop*, 2019, pp. 2–12.

[51] M. Klischat and M. Althoff, "Synthesizing traffic scenarios from formal specifications for testing automated vehicles," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2020, pp. 2065–2072.

[52] I. Majzik, O. Semeráth, C. Hajdu, K. Marussy, Z. Szatmári, Z. Micskei, A. Vörös, A. A. Babikian, and D. Varró, "Towards system-level testing with coverage guarantees for autonomous vehicles," in *Proc. of the 22nd ACM/IEEE Int. Conf. on Model Driven Engineering Languages and Systems*, 2019, pp. 89–94.

[53] J. Zhou and L. del Re, "Reduced complexity safety testing for ADAS & ADF," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5985–5990, 2017.

[54] E. Rocklage, H. Kraft, A. Karatas, and J. Seewig, "Automated scenario generation for regression testing of autonomous vehicles," in *Proc. of the IEEE Int. Conf. on Intell. Transportation Sys.*, 2018, pp. 476–483.

[55] C. Amersbach and H. Winner, "Defining required and feasible test coverage for scenario-based validation of highly automated vehicles," in *Proc. of the IEEE Int. Conf. on Intell. Transportation Sys.*, 2019, pp. 425–430.

[56] J. Duan, F. Gao, and Y. He, "Test scenario generation and optimization technology for intelligent driving systems," *IEEE Intell. Transportation Sys. Magazine*, vol. 14, no. 1, pp. 115–127, 2022.

[57] Y. Li, J. Tao, and F. Wotawa, "Ontology-based test generation for automated and autonomous driving functions," *Information and Software Technology*, vol. 117, Art. no. 106200, 2020.

[58] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2014, pp. 420–425.

[59] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Trans. on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[60] Z. Zhang, G. Ernst, S. Sedwards, P. Arcaini, and I. Hasuo, "Two-layered falsification of hybrid systems guided by Monte Carlo tree search," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2894–2905, 2018.

[61] S. Maierhofer, A. K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2020, pp. 752–759.

[62] E. I. Liu and M. Althoff, "Computing specification-compliant reachable sets for motion planning of automated vehicles," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2021, pp. 1037–1044.

[63] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. on Intell. Vehicles*, vol. 6, no. 2, pp. 232–248, 2021.

[64] V. Kaibel and M. E. Pfetsch, "Some algorithmic problems in polytope theory," in *Algebra, Geometry and Software Systems*, M. Joswig and N. Takayama, Eds. Springer, 2003, pp. 23–47.

[65] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, "Constrained zonotopes: A new tool for set-based estimation and fault detection," *Automatica*, vol. 69, pp. 126–136, 2016.

[66] V. Raghuraman and J. P. Koeln, "Set operations and order reductions for constrained zonotopes," *Automatica*, vol. 139, Art. no. 110204, 2022.

[67] I. Mitchell, "Comparing forward and backward reachability as tools for safety analysis," in *Hybrid Systems: Computation and Control*, 2007, pp. 428–443.

[68] G. Frehse, B. H. Krogh, and R. A. Rutenbar, "Verifying analog oscillator circuits using forward/backward abstraction refinement," in *Proc. of the Int. Conf. on Design, Automation and Test in Europe*, 2006.

[69] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Trans. on Intell. Transportation Sys.*, vol. 19, no. 6, pp. 1855–1866, 2018.

[70] B. Schürmann, M. Klischat, N. Kochdumper, and M. Althoff, "Formal safety net control using backward reachability analysis," *IEEE Trans. on Automatic Control*, vol. 67, no. 11, 2022.

[71] E. B. Saff and A. D. Snider, *Fundamentals of Matrix Analysis with Applications*. John Wiley & Sons, 2015.

[72] A. Couëtoux, J. B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, "Continuous upper confidence trees," in *Proc. of the Int. COnf. on Learning and Intelligent Optimization*, 2011, pp. 433–445.

[73] A. Couëtoux, M. Milone, M. Brendel, H. Doghmen, M. Sebag, and O. Teytaud, "Continuous rapid action value estimates," in *Proc. of the 3rd Asian Conf. on Machine Learning*, 2011, pp. 19–31.

[74] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét frame," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 987–993.

Moritz Klischat Moritz Klischat graduated with an M.Sc. degree in Mechanical Engineering from the Technical University of Munich, Germany, in 2017. Currently, he pursues a Ph.D. in computer science in the group of Prof. Dr.-Ing. Matthias Althoff at the Technical University of Munich. His research interests include testing methods for autonomous vehicles, reachability analysis, and motion planning.

Matthias Althoff is an associate professor in computer science at the Technical University of Munich, Germany. He received his diploma engineering degree in Mechanical Engineering in 2005, and his Ph.D. degree in Electrical Engineering in 2010, both from the Technical University of Munich, Germany. From 2010 to 2012 he was a postdoctoral researcher at Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Ilmenau University of Technology, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.

Chapter 5

Conclusions and Future Work

In this chapter, we conclude the results of the dissertation and relate our work to approaches proposed in the literature that emerged during the writing of this thesis. Finally, we give an outlook on promising directions for future research arising from our work.

5.1 Conclusions

In this thesis, we have proposed new methods for testing and falsifying motion planning algorithms for autonomous vehicles that can be used at different stages during their development phase. We divide the methodology in this thesis into two main chapters. In Ch. 3, we focus on the problem of generating safety-critical traffic scenarios out of non-critical scenarios. To achieve this, we develop an approach to optimize the trajectories of other traffic participants in a provided scenario with respect to a criticality metric. We propose using the drivable area of the SUT as a metric that allows optimizing scenarios independently of a specific system because it abstracts system behavior through computing the set of all feasible trajectories from a given initial state. Based on the drivable area, we further develop methods to prune irrelevant regions from the parameter space, which is applicable to arbitrary road layouts. We show that our method can be applied automatically to a variety of scenario types in highway and urban environments in Sec. 3.2.

Our method is integrated into a framework presented in Sec. 3.2 that automates the generation of scenarios starting with open-source maps from OpenStreetMap that are extracted and enhanced for their use in traffic simulators and as input for motion planning algorithms. Using the traffic simulator SUMO, the maps are populated with traffic participants and used as input for the optimization method that optimizes the motion of the traffic participants to increase a criticality metric of the scenario. Through this completely automated pipeline based on open-source resources and simulation software, we enable the efficient and scalable generation of test scenarios for use in research and industry. The evaluation on more than 1000 optimized scenarios shows that our optimization method is able to reduce the drivable area to 20—30% of its original size, hence resulting in a considerably decreased solution space in which an SUT would be able to plan a collision-free trajectory. Since the optimization of scenarios is independent of a specific SUT, the resulting scenarios can be used to benchmark motion planning algorithms. Hence, the scenarios are published as part of the open-source CommonRoad [35] benchmark suite.

To reduce the time required for generating scenarios, we develop a method for computing the reachable sets more efficiently than previous methods. We exploit invariance properties of the reachable set regarding translations and the initial velocity and pre-compute the reachability between subsets of the reachable set to discretize the problem. This enables the graph-based computation of the reachable set during runtime. Combined with our new methods to reduce the over-approximation of the resulting sets, we find in Sec. 3.3 that our

method reduces the time for computing the reachable set by 80%. In addition, we can consider additional nonlinear constraints, like the friction circle, that were not considered by previous methods. Beyond the generation of scenarios, these advancements are beneficial for applications with real-time requirements such as motion planning.

In the second part of this thesis, we develop in Ch. 4 a method that enables using falsification in a scenario-based approach. To achieve this, we first propose in Sec. 4.1 a constraint-based representation of abstract scenario specifications from which scenarios can be synthesized efficiently using optimization- or sampling-based methods. We then develop an over-approximative algorithm for computing the reachable set of multiple traffic participants that complies with a scenario specification. We propose an approach for sampling trajectories within these sets, which we combine with MCTS for finding specification-compliant scenarios that falsify motion planning algorithms. By constraining the action space within the MCTS using the reachable sets, our methods improve the performance and efficiency when solving this complex search problem. Since the reachable sets at all times only contain states from which failure states can possibly be reached, we further improve the efficiency of the falsification. In Sec. 4.2, we evaluate the method on several abstract scenarios and show improved performance of the falsification compared to a MCTS baseline approach without our reachability-based method. Our method is able to falsify the SUT in 79% of all falsification trials compared to 12% achieved by the baseline approach. Furthermore, we can show that for more complex scenarios with smaller feasible solution spaces for generating trajectories of other traffic participants, the computation time of our method is decreased, meaning that our approach becomes more efficient for complex scenarios.

5.2 Related Work

During the writing of this thesis, other authors proposed related methods for SUT-agnostic optimization of traffic scenarios using metrics using similar concepts like the drivable area. After the publication of our work [1], similar approaches were proposed using metrics that assess and optimize the criticality of traffic scenarios using metrics very similar to the drivable area [101, 102]. Another advantage of our work when using the drivable area is that the potential collisions in the generated scenarios are avoidable as long as the drivable area is not empty. An approach that aims at finding avoidable collisions of motion planners is presented in [103], where different parameterizations of the same planner are used to evaluate whether there is a system realization that can avoid a collision. While being under-approximative compared to our over-approximative drivable area, this metric might not result in false negatives, i.e., scenarios for which a collision is not avoidable for the SUT even though the drivable area is not vanishing. Instead, it can miss many scenarios where a collision could be avoided by using entirely different algorithms instead of just different parameterizations of the same algorithm.

5.3 Future Work

In our work on the optimization-based generation of test scenarios in Ch. 3, traffic rules beyond simple rules such as collision avoidance or off-road driving were not explicitly considered yet. Future research could be carried out from here, adding metrics that consider more complex rules formulated in temporal logic [104]. Temporal-logic rules can be combined with our drivable-area-based criticality metric when used as additional constraints in

the reachability analysis like in [105] to consider the traffic rules in the criticality metric. First steps in this direction were already explored in a student thesis [107]. Similarly, for traffic rules applying to other traffic participants, new research can focus on scenarios where other the traffic participants (almost) violate certain rules.

Further research can be conducted on the drivable-area-based criticality metric to exploit more information from the reachable set that is currently not considered for the criticality. One goal could be to consider not only the size of the drivable area but to additionally analyze the required acceleration to reach each part of the drivable area. The motivation behind this would be to distinguish, e.g., between scenarios in which the ego vehicle is simply blocked by other traffic participants and unable to move and scenarios in which the drivable area is small but large inputs are required to avoid collisions. Information about the acceleration or velocity differences could be extracted from the reachability analysis as well and would provide additional insights. Suitable extensions of the criticality metric could be further backed by experimental evaluations, e.g., through user studies on perceived criticality by vehicle passengers or studies on the correlation of the criticality metrics with the performance of motion planning algorithms. The results of the studies could indicate which metric is best suited for predicting the accident rate in a large scenario dataset.

Instead of using our scenarios for testing, they might be used to train motion planning algorithms that include machine-learning components. Since the datasets used for training contain in general naturalistic traffic data, they are biased toward uncritical and repetitive scenarios. Augmenting those datasets with edge cases generated by our methods could make training more efficient and the resulting models more reliable in safety-critical situations. For this application, additional focus could be placed on realistic behavior when generating scenarios.

Another aspect that could be considered during the scenario generation is the visibility of other road users, which might be hidden by buildings or other obstacles. Undetected obstacles can pose a severe risk that needs to be addressed properly by motion planning algorithms, e.g., in situations with bad oversight. Such critical configurations could be considered when optimizing the motion of other traffic participants and the drivable area of the SUT could be used to ensure that certain traffic participants remain hidden for extended periods of time. Alternatively, the ratio of locations of the drivable area at which other traffic participants are hidden versus locations at which they are visible could be used as an additional term of the criticality metric.

Our falsification method based on reachability analysis holds further potential when integrated with large databases of abstract scenarios and methods to automatically generate abstract scenarios. In that case, the falsification could not only be carried out within single scenarios but the search space could be extended to more a more abstract level to consider, e.g., different maneuver options of other vehicles in the specifications. Similarly, other scenario information such as the road topology or additional properties like weather or road friction could be considered.

Additional improvements can be gained from heuristics based on machine learning models that could be integrated with our MCTS or other approaches similar to those in [70, 87–90]. Currently, we use random sampling in the playouts of the MCTS and the learning-based models can be trained to expose potentially safety-critical behavior which could further increase the efficiency of the falsification. Beyond criticality, probabilistic models of other agents' behavior could be utilized for the sampling. In that case, our falsification methods could be applied to the problem formulation referred to as *stress testing* [106] as well. In turn, a combination with our approach would enable explicitly considering scenario specifications which is not yet addressed in these approaches.

Bibliography

[11] N. Kalra and S. 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*, vol. 94, pp. 182–193, 2016.

[12] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, 2020.

[13] ISO 26262, "Road vehicles – functional safety," 2011.

[14] H. Martin, K. Tschabuschnig, O. Bridal, and D. Watzenig, "Functional safety of automated driving systems: Does iso 26262 meet the challenges?" in *Automated Driving: Safer and More Efficient Future Driving*. Springer, 2016, pp. 387–416.

[15] T. Zhao, E. Yurtsever, J. A. Paulson, and G. Rizzoni, "Formal certification methods for automated vehicle safety assessment," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 232–249, 2023.

[16] C. Pek, S. Manzinger, M. Koschi, and M. Althoff, "Using online verification to prevent autonomous vehicles from causing accidents," *Nature Machine Intelligence*, vol. 2, no. 9, pp. 518–528, 2020.

[17] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv:1708.06374*, 2017.

[18] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *International Journal of Robotics Research*, vol. 39, no. 12, pp. 1419–1469, 2020.

[19] K. Naik and P. Tripathy, *Software Testing and Quality Assurance: Theory and Practice*. Wiley, 2008.

[20] X. Zhang, J. Tao, K. Tan, M. Torngren, J. M. Gaspar Sanchez, M. R. Ramli, X. Tao, M. Gyllenhammar, F. Wotawa, N. Mohan, M. Nica, and H. Felbinger, "Finding critical scenarios for automated driving systems: A systematic mapping study," *IEEE Transactions on Software Engineering (early access)*, 2022.

[21] D. J. Fremont, E. Kim, Y. V. Pant, S. A. Seshia, A. Acharya, X. Bruso, P. Wells, S. Lemke, Q. Lu, and S. Mehta, "Formal scenario-based testing of autonomous vehicles: From simulation to the real world," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, 2020.

[22] Z. Zhong, Y. Tang, Y. Zhou, V. d. O. Neves, Y. Liu, and B. Ray, "A survey on scenario-based testing for automated driving systems in high-fidelity simulation," *arXiv:2112.00964*, 2021.

[23] F. Schuldt, F. Saust, B. Lichte, M. Maurer, and S. Scholz, "Effiziente systematische Testgenerierung für Fahrerassistenzsysteme in virtuellen Umgebungen," in *AAET2013 - Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel*, 2013, pp. 114 – 134.

[24] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 1821–1827, 2018.

[25] H. Weber, J. Bock, J. Klimke, C. Roesener, J. Hiller, R. Krajewski, A. Zlocki, and L. Eckstein, "A framework for definition of logical scenarios for safety assurance of automated driving," *Traffic Injury Prevention*, vol. 20, no. S1, pp. 65–70, 2019.

[26] T. Menzel, G. Bagschik, L. Isensee, A. Schomburg, and M. Maurer, "From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment," *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 2383–2390, 2019.

[27] A. Gambi, T. Huynh, and G. Fraser, "Generating effective test cases for self-driving cars from police reports," in *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 257–267.

[28] C. Sippl, F. Bock, D. Wittmann, H. Altinger, and R. German, "From simulation data to test cases for fully automated driving and ADAS," in *Proceedings of the International Conference on Testing Software and Systems*, 2016, pp. 191–206.

[29] C. Wolschke, T. Kuhn, D. Rombach, and P. Liggesmeyer, "Observation based creation of minimal test suites for autonomous vehicles," in *Proceedings of the IEEE 28th International Symposium on Software Reliability Engineering Workshops*, 2017, pp. 294–301.

[30] C. Roesener, F. Fahrenkrog, A. Uhlig, and L. Eckstein, "A scenario-based assessment approach for automated driving by using time series classification of human-driving behaviour," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, 2016, pp. 1360–1365.

[31] F. Kruber, J. Wurst, and M. Botsch, "An unsupervised random forest clustering technique for automatic traffic scenario categorization," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, 2018, pp. 2811–2818.

[32] R. Krajewski, T. Moers, D. Nerger, and L. Eckstein, "Data-driven maneuver modeling using generative adversarial networks and variational autoencoders for safety validation of highly automated vehicles," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 2383–2390.

[33] F. Hauer, I. Gerostathopoulos, T. Schmidt, and A. Pretschner, "Clustering traffic scenarios using mental models as little as possible," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 1007–1012.

[34] S. Geyer, M. Baltzer, B. Franz, S. Hakuli, M. Kauer, M. Kienle, S. Meier, T. Weigerber, K. Bengler, R. Bruder, F. Flemisch, and H. Winner, "Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance," *IET Intelligent Transport Systems*, vol. 8, no. 3, pp. 183–189, 2014.

[35] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.

[36] W. Damm, E. Möhlmann, T. Peikenkamp, and A. Rakow, "A formal semantics for traffic sequence charts," in *Lecture Notes in Computer Science*, 2018, vol. 10760, pp. 182–205.

[37] R. Queiroz, T. Berger, and K. Czarnecki, "Geoscenario: An open dsl for autonomous driving scenario representation," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 287–294.

[38] C. Sippl, F. Bock, C. Lauer, A. Heinz, T. Neumayer, and R. German, "Scenario-based systems engineering: An approach towards automated driving function development," in *Proceedings of the 13th Annual IEEE International Systems Conference*, 2019, pp. 1–8.

[39] B. Schütt, T. Braun, S. Otten, and E. Sax, "SceML: A graphical modeling framework for scenario-based testing of autonomous vehicles," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 2020, pp. 114–120.

[40] E. de Gelder, J.-P. Paardekooper, A. K. Saberi, H. Elrofai, O. O. den Camp, S. Kraines, J. Ploeg, and B. De Schutter, "Towards an ontology for scenario definition for the assessment of automated vehicles: An object-oriented framework," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 2, pp. 300–314, 2022.

[41] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[42] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, and D. Zufferey, "PARAMCOSM: A test framework for autonomous driving simulations," in *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*, 2021, pp. 172–195.

[43] D. J. Fremont, X. Yue, T. Dreossi, A. L. Sangiovanni-Vincentelli, S. Ghosh, and S. A. Seshia, "Scenic: A language for scenario specification and scene generation," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019, pp. 63–78.

[44] S. Tang, Z. Zhang, Y. Zhang, J. Zhou, Y. Guo, S. Liu, S. Guo, Y.-F. Li, L. Ma, Y. Xue, and Y. Liu, "A survey on automated driving system testing: Landscapes and trends," *ACM Transactions on Software Engineering and Methodology*, vol. 13, pp. 104–116, 2023.

[45] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. P. Flotterod, R. Hilbrich, L. Lucken, J. Rummel, P. Wagner, and E. Wiebner, "Microscopic traffic simulation using sumo," in *Proceedings of IEEE Conference on Intelligent Transportation Systems*, 2018, pp. 2575–2582.

[46] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, 2000.

[47] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1999, no. 1, pp. 86–94, 2007.

[48] S. Wagner, K. Groh, T. Kuhbeck, and A. Knoll, "Towards cross-verification and use of simulation in the assessment of automated driving," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 1589–1596.

[49] L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Neurohr, C. Gutenkunst, and E. Böde, "Criticality metrics for automated driving: A review and suitability analysis of the state of the art," *Archives of Computational Methods in Engineering*, vol. 30, no. 1, pp. 1–35, jan 2023.

[50] S. Söntges, M. Koschi, and M. Althoff, "Worst-case analysis of the time-to-react using reachable sets," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1891–1897.

[51] A. Tamke, T. Dang, and G. Breuel, "A flexible method for criticality assessment in driver assistance systems," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2011, pp. 697–702.

[52] S. Herrmann, W. Utschick, M. Botsch, and F. Keck, "Supervised learning via optimal control labeling for criticality classification in vehicle active safety," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, 2015, pp. 2024–2031.

[53] P. Junietz, F. Bonakdar, B. Klamann, and H. Winner, "Criticality metric for the safety validation of automated driving using model predictive trajectory optimization," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, 2018, pp. 60–65.

[54] J. Dahl, G. R. De Campos, C. Olsson, and J. Fredriksson, "Collision avoidance: A literature review on threat-assessment techniques," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 101–113, 2019.

[55] Y. Li, J. Tao, and F. Wotawa, "Ontology-based test generation for automated and autonomous driving functions," *Information and Software Technology*, vol. 117, Art. no. 106200, 2020.

[56] E. Rocklage, H. Kraft, A. Karatas, and J. Seewig, "Automated scenario generation for regression testing of autonomous vehicles," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 476–483.

[57] J. Duan, F. Gao, and Y. He, "Test scenario generation and optimization technology for intelligent driving systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 1, pp. 115–127, 2022.

[58] C. Amersbach and H. Winner, "Defining required and feasible test coverage for scenario-based validation of highly automated vehicles," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, 2019, pp. 425–430.

[59] I. Majzik, O. Semeráth, C. Hajdu, K. Marussy, Z. Szatmári, Z. Micskei, A. Vörös, A. A. Babikian, and D. Varró, "Towards system-level testing with coverage guarantees for autonomous vehicles," in *Proceedings of the 22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 2019, pp. 89–94.

[60] J. Zhou and L. del Re, "Reduced complexity safety testing for ADAS & ADF," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5985–5990, 2017.

[61] E. De Gelder and J. P. Paardekooper, "Assessment of automated driving systems using real-life scenarios," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 589–594.

[62] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 595–607, 2017.

[63] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 9849–9860.

[64] T. A. Wheeler, M. J. Kochenderfer, and P. Robbel, "Initial scene configurations for highway traffic propagation," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, 2015, pp. 279–284.

[65] S. Jesenski, J. E. Stellet, F. Schiegg, and J. M. Zollner, "Generation of scenes in intersections for the validation of highly automated driving functions," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 502–509.

[66] E. de Gelder, E. Cator, J.-P. Paardekooper, O. O. den Camp, and B. De Schutter, "Constrained sampling from a kernel density estimator to generate scenarios for the assessment of automated vehicles," in *Proceedings of the IEEE Intelligent Vehicles Symposium, 4th Workshop on "Ensuring and Validating Safety for Automated Vehicles"*, 2021.

[67] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 204–211.

[68] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "Trafficsim: Learning to simulate realistic multi-agent behaviors," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 395–10 404.

[69] S. Shiroshita, S. Maruyama, D. Nishiyama, M. Y. Castro, K. Hamzaoui, G. Rosman, J. DeCastro, K.-H. Lee, and A. Gaidon, "Behaviorally diverse traffic simulation via reinforcement learning," in *IEEE International Conference on Intelligent Robots and Systems*, 2020, pp. 2103–2110.

[70] S. Feng, X. Yan, H. Sun, Y. Feng, and H. X. Liu, "Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment," *Nature Communications*, vol. 12, no. 1, pp. 1–14, 2021.

[71] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivačić, and A. Gupta, "Probabilistic temporal logic falsification of cyber-physical systems," *ACM Transactions on Embedded Computing Systems*, vol. 12, no. 2s, pp. 1–30, 2013.

[72] A. Zutshi, S. Sankaranarayanan, J. V. Deshmukh, and J. Kapinski, "Multiple shooting, CEGAR-based falsification for hybrid systems," in *Proceedings of the International Conference on Embedded Software*, 2014.

[73] T. Akazaki, S. Liu, Y. Yamagata, Y. Duan, and J. Hao, "Falsification of cyber-physical systems using deep reinforcement learning," in *Lecture Notes in Computer Science*, 2018, vol. 10951, pp. 456–465.

[74] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer, "A survey of algorithms for black-box safety validation of cyber-physical systems," *Journal of Artificial Intelligence Research*, vol. 72, pp. 377–428, 2021.

[75] O. Buehler and J. Wegener, "Evolutionary functional testing of an automated parking system," in *Proceedings of the International Conference on Computer, Communication and Control Technologies*, 2003, pp. 26–31.

[76] C. Gladisch, T. Heinz, C. Heinzemann, J. Oehlerking, A. von Vietinghoff, and T. Pfitzer, "Experience paper: Search-based testing in automated driving control applications," in *IEEE/ACM International Conference on Automated Software Engineering*, 2019, pp. 26–37.

[77] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. Iyer, "AV-FUZZER: Finding safety violations in autonomous driving systems," in *Proceedings of the International Symposium on Software Reliability Engineering*, 2020, pp. 25–36.

[78] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun, "AdvSim: Generating safety-critical scenarios for self-driving vehicles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9904–9913.

[79] A. Li, S. Chen, L. Sun, N. Zheng, M. Tomizuka, and W. Zhan, "SceGene: Bio-inspired traffic scenario generation for autonomous driving testing," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, 2021.

[80] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing S-TaLiRo as an automatic test generation framework for autonomous vehicles," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, 2016, pp. 1470–1475.

[81] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019, pp. 8271–8277.

[82] C. E. Tuncali and G. Fainekos, "Rapidly-exploring random trees-based test generation for autonomous vehicles," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, 2019, pp. 661–666.

[83] M. Koschi, C. Pek, S. Maierhofer, and M. Althoff, "Computationally efficient safety falsification of adaptive cruise control systems," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, 2019, pp. 2879–2886.

[84] C. E. Tuncali, G. Fainekos, D. Prokhorov, H. Ito, and J. Kapinski, "Requirements-driven test generation for autonomous vehicles with machine learning components," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 265–280, 2020.

[85] N. Arechiga, "Specifying safety of autonomous vehicles in signal temporal logic," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 58–63.

[86] X. Qin, N. Aréchiga, A. Best, and J. Deshmukh, "Automatic testing with reusable adversarial agents," *arXiv:1910.13645*, 2019.

[87] W. Ding, B. Chen, M. Xu, and D. Zhao, "Learning to collide: An adaptive safety-critical scenarios generating method," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2020, pp. 2243–2250.

[88] B. Chen, X. Chen, Q. Wu, and L. Li, "Adversarial evaluation of autonomous vehicles in lane-change scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 10 333–10 342, 2022.

[89] D. Rempe, J. Philion, L. J. Guibas, S. Fidler, and O. Litany, "Generating useful accident-prone driving scenarios via a learned traffic prior," in *Proceeding of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 284–17 294.

[90] W. Ding, M. Xu, and D. Zhao, "CMTS: A conditional multiple trajectory synthesizer for generating safety-critical driving scenarios," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020, pp. 4314–4321.

[91] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, "Adaptive stress testing for autonomous vehicles," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1898–1904.

[92] G. Chou, Y. E. Sahin, L. Yang, K. J. Rutledge, P. Nilsson, and N. Ozay, "Using control synthesis to generate corner cases: A case study on autonomous driving," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2906–2917, 2018.

[93] L. Capito, B. Weng, U. Ozguner, and K. Redmill, "A modeled approach for online adversarial test of operational vehicle safety," in *Proceedings of the American Control Conference*, 2021, pp. 398–404.

[94] G. Mullins, P. Stankiewicz, and S. Gupta, "Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017, pp. 1443–1450.

[95] M. Waga, "Falsification of cyber-physical systems with robustness-guided black-box checking," in *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, vol. 20, 2020, p. 13.

[96] L. Yang and N. Ozay, "Synthesis-guided adversarial scenario generation for gray-box feedback control systems with sensing imperfections," *ACM Transactions on Embedded Computing Systems*, vol. 20, pp. 1–25, 2021.

[97] F. Hauer, A. Pretschner, and B. Holzmüller, "Fitness functions for testing automated and autonomous driving systems," in *SAFECOMP 2019: Computer Safety, Reliability, and Security*, pp. 69–84.

[98] A. Nonnengart, M. Klusch, and M. Christian, "CriSGen: Constraint-based generation of critical scenarios for autonomous vehicles," in *Proceedings of the International Workshop on Formal Methods for Autonomous Systems*, 2019.

[99] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia, "VerifAI: A toolkit for the formal design and analysis of artificial intelligence-based systems," in *Lecture Notes in Computer Science*, vol. 11561, 2019, pp. 432–442.

[100] M. Althoff and S. Lutz, "Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1326–1333.

[101] A. Bussler, L. Hartjen, R. Philipp, and F. Schuldt, "Application of evolutionary algorithms and criticality metrics for the verification and validation of automated driving systems at urban intersections," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 128–135, 2020.

[102] Z. Ghodsi, S. K. S. Hari, I. Frosio, T. Tsai, A. Troccoli, S. W. Keckler, S. Garg, and A. Anandkumar, "Generating and characterizing scenarios for safety testing of autonomous vehicles," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2021, pp. 157–164.

[103] A. Calo, P. Arcaini, S. Ali, F. Hauer, and F. Ishikawa, "Generating avoidable collision scenarios for testing autonomous driving systems," in *Proceedings of the IEEE International Conference on Software Testing, Verification and Validation*, 2020, pp. 375–386.

[104] S. Maierhofer, A. K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 752–759.

[105] E. I. Liu and M. Althoff, "Computing specification-compliant reachable sets for motion planning of automated vehicles," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2021, pp. 1037–1044.

[106] M. Koren and M. J. Kochenderfer, "Efficient autonomy validation in simulation with adaptive stress testing," in *Proceedings of the IEEE Intelligent Transportation Systems Conference*, 2019, pp. 4178–4183.

Supervised Theses of Students

[107] Q. Kögl, "Generation of critical scenarios for automated vehicles with consideration of traffic rules," Bachelor Thesis, Technische Universität München, 2019.

[108] M. Rieger, "Automated conversion of road networks from openstreetmap to lanelets," Bachelor Thesis, Technische Universität München, 2018.

[109] S. Klimaschka, "Generation of critical scenarios for automated vehicles," Bachelor Thesis, Technische Universität München, 2018.

[110] E. Tatlow, "A survey of global solvers for the optimisation of critical traffic scenarios," Bachelor Thesis, Technische Universität München, 2019.

[111] C. Brunnett, "Automatic generation of safety-critical test scenarios for autonomous vehicles using set-based predictions of traffic participants," Master Thesis, Technische Universität München, 2019.

[112] F. Harb, "Graph-based similarity computation of high definition maps for automated vehicles," Bachelor Thesis, Technische Universität München, 2020.

[113] L. Kraußе, "Lane extraction from aerial images using deep learning for generating hd maps," Bachelor Thesis, Technische Universität München, 2020.

[114] M. Frühauf, "Traffic scenario synthesis from temporal logic constraints," Bachelor Thesis, Technische Universität München, 2021.

[115] M. C. Üste, "Assessing the similarity of traffic scenarios using graph neural networks," Guided Research Paper, Technische Universität München, 2022.