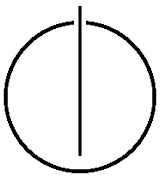
FAKULTÄT FÜR INFORMATIK

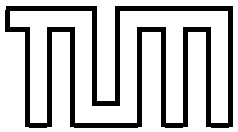
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Dissertation in Informatik

# On Scenario-Based Testing of Automated and Autonomous Driving Systems

Florian Hauer





FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Lehrstuhl IV - Software and Systems Engineering

# On Scenario-Based Testing of Automated and Autonomous Driving Systems

*Florian Hauer*

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

| | |
|---|---|
| Vorsitzender: | Prof. Dr. Matthias Althoff |
| Prüfer der Dissertation: | |
| 1. | Prof. Dr. Alexander Pretschner |
| 2. | Prof. Dr. Markus Lienkamp |

Die Dissertation wurde am 01.03.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 08.04.2021 angenommen.

# Acknowledgments

My utmost gratitude goes to my supervisor Prof. Dr. Alexander Pretschner. He provided me the opportunity as well as the environment to acquire and improve many valuable methodological, technical, and social skills. His research culture of addressing hard unsolved problems from the industry ensured that my research is relevant both scientifically and practically. In our discussions, he challenged my research to the extreme while still being constructive, which made me grow and definitely shaped my way of thinking and my way of structuring argumentations. His guidance and many years of experience taught me the ability to abstract and to take a step back before approaching a problem, which now provides me with the confidence to face any future problem.

Many thanks go to my second supervisor Prof. Dr. Markus Lienkamp for his fast and constructive feedback. The interdisciplinary discussions with him and members of his chair at various occasions broadened my horizon in the domain of automated and autonomous driving systems.

My sincere appreciation goes to Prof. Dr. Shaukat Ali. During the discussions with him and our colleagues at the NII in Japan, I could learn from his extensive publishing experience. His kindness and positivity along with his constructive feedback on my work was most welcome in the process of finishing this thesis.

My sincere gratitude goes to Bernd Holzmüller. With his methodological experience gathered in the automotive industry, he pinpointed interesting problems in practice and challenged my solutions for their practical applicability. He introduced me to numerous experts all over the domain of automated and autonomous driving systems, which allowed me to gather a plethora of different perspectives on the problems discussed in this thesis.

Last but not least I would like to thank my colleagues for the great time I had at the chair. With many of them, I had the chance to work on interesting problems, for many of which we could successfully publish solutions. I genuinely enjoyed the intense scientific discussions as well as the coffee break chatting, which oftentimes morphed into each other.

# Zusammenfassung

Automatisierte und autonome Fahrsysteme werden in virtuellen Testfahrten erprobt, die meist mithilfe szenariobasierter Tests durchgeführt werden. Dabei wird das Fahrsystem Instanzen sowohl von häufig vorkommenden als auch von speziellen Verkehrsszenarien ausgesetzt, um Vertrauen in die Sicherheit des Systems zu gewinnen. Intuitiv ergeben sich daraus zwei Hauptprobleme: Es wird eine vollständige Liste von Szenariotypen benötigt, die alle relevanten Szenariotypen enthält, und das Fahrsystem muss in Testfällen getestet werden, die interessante Instanzen dieser Szenariotypen darstellen.

Das erste Problem kann aus zwei Richtungen angegangen werden, nämlich konstruktiv die Vollständigkeit zu erhöhen und die Vollständigkeit so gut wie möglich zu bewerten. Derzeit leiten Experten Szenariotypen manuell anhand ihres mentalen Modells ab, das auf Erfahrung basiert. Dies erfordert eine Validierung, wofür Redundanz durch automatisierte Ableitung vorgeschlagen wird. Bestehende Arbeiten zur automatisierten Ableitung von Szenariotypen weisen entweder technische Einschränkungen auf oder verlassen sich auf das mentale Modell des Experten, wodurch der Redundanzaspekt verloren geht. In dieser Arbeit wird ein Clustering-Ansatz vorgestellt, der Szenarioinstanzen allein auf Basis syntaktischer Merkmale zu Szenariotypen gruppiert und damit den Einfluss des mentalen Modells des Experten auf die Entscheidung, welche Art von Daten erfasst und für das Clustering verwendet werden, reduziert. Die resultierende Liste der Szenariotypen muss auf Vollständigkeit geprüft werden. Da eine absolute Messung der Vollständigkeit nicht möglich ist, ist das Ziel eine Messung relativ zu realen Verkehrsdaten. Bestehende Arbeiten bieten keinen geeigneten Ansatz. In dieser Arbeit wird ein statistisches Modell beschrieben, das - basierend auf Verkehrsdaten - angibt, ob alle Szenariotypen bekannt sind, die im realen Verkehr bis zu einer bestimmten Wahrscheinlichkeit stattfinden.

Das zweite Problem erfordert zunächst die Generierung von Szenario-Instanzen, die die gewünschte Form haben und interessant sind, also “gute” Testfälle sind, und anschließend die Analyse, ob solche interessanten Testfälle generell “gut” bleiben, wenn sie wiederverwendet werden, z.B. für Regressionstests. Für die Testfallgenerierung wurden in der Vergangenheit suchbasierte Techniken vorgeschlagen. Die vorhandenen Arbeiten liefern jedoch entweder ad-hoc Fitnessfunktionen für ein bestimmtes System oder einen bestimmten Szenariotyp; oder konzentrieren sich auf die technischen Aspekte der suchbasierten Testszenario-Generierung unter der Annahme, dass die Fitnessfunktion gegeben ist. Dabei bleibt der methodische Aspekt unberücksichtigt, wie Fitnessfunktionen generell erstellt werden sollten. In dieser Arbeit werden Templates zur Formulierung von Fitnessfunktionen vorgestellt, sowie methodische Anleitungen zur Kombination und Anwendung, um sicherzustellen, dass die ausgewählten Testfälle die gewünschte Form haben und interessant sind. Einige solcher Testfälle können fehlerhaftes Verhalten aufzeigen. Die Wiederverwendbarkeit solcher Testfälle für verschiedene Versionen und Varianten von Fahrsystemen wird in einer Vielzahl von Arbeiten implizit vorausgesetzt. Nach bestem Wissen des Autors gibt es jedoch keine Arbeit, die tatsächlich einen Beweis für oder gegen diese Annahme liefert. In dieser Arbeit wird ein Gegenbeispiel vorgestellt, das zeigt, dass die Qualität

von Testfällen generell systemabhängig ist. Daher wird die naïve Wiederverwendung von Testfällen nicht empfohlen. Stattdessen wird vorgeschlagen, die Testfallgenerierung für verschiedene Systemversionen und -varianten erneut auszuführen.

# Abstract

Automated and autonomous driving systems are tested in virtual test drives, mostly conducted by scenario-based testing. This involves exposing the driving system to instances of both commonly encountered and special traffic scenario types to gain confidence in the safety of the system. Intuitively, this imposes two main problems: A complete list of scenario types is required containing all such relevant scenario types and the driving system needs to be tested in "good" test cases, which are interesting instances of these scenario types.

The first problem may be tackled from two directions, namely constructively increasing the completeness and assessing the completeness as much as possible. Currently, experts derive scenario types manually using their mental models based on experience. This requires validation, for which redundancy by automated derivation is suggested. Existing works on automated scenario type derivation either come with technical limitations or rely on the experts mental model nullifying the redundancy aspect. This work presents a clustering approach that groups scenario instances to scenario types solely based on syntactic features, reducing the influence of the experts' mental model to the decision of what kind of data is recorded and used for the clustering. The resulting list of scenario types needs to be assessed for completeness. Since an absolute measurement of completeness is infeasible, the goal is a measurement relative to real traffic data. Existing works do not provide a suitable approach. In this work, a statistical model is described that states – based on traffic data – whether all scenario types are known that take place in real traffic up to a certain likelihood.

The second problem requires first the generation of scenario instances that are of the desired form and are interesting, which are "good" test cases, and afterwards the analysis whether such interesting test cases generally stay "good" when re-used, e.g. for regression testing. For test case generation, search-based techniques have been suggested in the past. However, existing works either provide ad-hoc fitness functions for a specific system or scenario type; or focus on the technical aspects of search-based test scenario generation assuming the fitness function to be given. This leaves the methodological aspect unconsidered of how fitness functions generally should be created. This work presents templates to formulate fitness functions as well as the means to combine and apply them to ensure that selected test cases are of the desired form and are interesting. Some of those test cases may reveal faulty behavior. Re-usability of such test cases for different versions and variants of driving systems is implicitly assumed by a variety of works. However, to the best of the author's knowledge there is no work actually providing evidence for or against this assumption. In this work, a counterexample is presented that shows that the quality of test cases generally is system-dependent. Thus, naïvely re-using test cases is not recommended. Instead, it is suggested to re-execute the test case generation for different system versions and variants.

# Contents

# Part I.

# Introduction and Background

# 1. Introduction

*This chapter introduces the topic of scenario-based testing of automated and autonomous driving systems, describes this work's goal of providing a framework for test case generation while striving for completeness. The gaps in the literature and practice as well as the contributions to fill them are described, followed by a description of this work's structure. Parts of this chapter have previously appeared in peer-reviewed publications [54, 56, 57, 59], co-authored by the author of this thesis.*

## 1.1. Testing Automated and Autonomous Driving Systems

The strive for highly automated and autonomous driving systems results in more and more complex and capable systems. Due to the complexity of these systems and the complexity and sheer number of possible driving scenarios, ensuring safety and functional correctness is one of the crucial challenges [62, 63, 82, 85, 86, 87, 126]. Since verification and validation solely by real test drives are practically infeasible due to the high number of required driven kilometer or driven time to reach sufficient confidence [63, 73, 161, 176], the focus shifted to virtual test drives [69].

The de facto standard for virtually testing automated and autonomous driving systems is *scenario-based testing*: testing such driving systems in challenging traffic scenarios. These are (extreme) instances of so-called *scenario types*. Scenario types, also named functional scenarios [106], capture recurring traffic situations, e.g. lane changes or cut-ins. During testing, scenario types are used to generate *scenario instances*, also called concrete test scenarios [106]. Proving that the system works as expected in challenging instances increases the confidence in the system's overall correctness in this scenario type. To extend this confidence to the whole driving task the system has to perform, the list of known scenario types for test case generation is required to be "complete" [40, 59, 130, 129].

### 1.1.1. Scenario Type Derivation

The derivation of a "complete" list of scenario types is challenging. A common approach in industry is to have experts manually create such lists of scenario types. Various means to formalize their mental models are suggested, e.g., the use of skill graphs [16] or ontologies [14, 37]. However, no matter how comprehensive such lists of scenario types may be,

the manual creation process poses multiple risks, most importantly: (i) certain scenario types might be overlooked, (ii) the mental model, according to which the derivation is done, might be inadequate. Using an expert's mental model for scenario type derivation influences the way that scenario types are *structured* and at which level of *granularity* they are located. Literature also calls this the *human test scenario bias* [83]. For instance, an expert could derive scenario types according to the existence of maneuvers like braking or lane changing. Similarly, the derivation could be stopped at the granularity level of *contains a lane change* or *contains a lane change to the right in front of another vehicle on the target lane*. Which of those are appropriate and suitable is generally unclear. Thus, validation is required for the process of manual scenario type derivation.

Completeness of a list of scenario types may generally not be possible. However, one way of striving for completeness and validity is to create a list of scenario types in a methodologically different way. This creates redundancy to potentially identify overlooked scenario types and to confirm or contradict the experts' mental model. Such an alternative approach is to derive scenario types from various kinds of real driving data, e.g. [21, 52, 53, 74, 89, 90, 92, 101, 164, 174]. Recordings of real driving and traffic contain a high number of scenario instances, from which scenario types can be derived in an automated way. Note the assumption that the collected data is sufficiently representative for the system's use case (also called *Operational Design Domain* [157]), as—usually implicitly—assumed by most existing data-driven approaches in this domain: It must cover the multitude of driving tasks and environments of the system's use case, e.g. highway data in different parts of the country for a highway pilot system. The goal is to perform this automated derivation without relying too much on handcrafted features and, thus, reducing the dependence on mental models. This way, it yields scenario types of various levels of granularity, structured mostly independently of a mental model. Only then does this approach create redundancy and can be used to evaluate manually derived scenario types with respect to completeness and adequacy of structure and granularity. In contrast, automated derivation with features based on mental models will yield scenario types related to those mental models, e.g. encoding "braking maneuver" into the features will result in scenario types related to "braking meneuver" [93]. Note that it is inherently impossible to be completely independent of the expert's mental model, since the expert decides what kind of data is collected, e.g. recording the velocity of vehicles over time. Thus, the type of collected data influences the granularity and structure of scenario types. Current approaches either entirely rely on features based on mental models, e.g. [93, 121], or are technically limited, i.e. restrictions in the number of vehicles, the total duration of the scenario, and the type, number, and length of time series; [19, 33, 93, 98, 165].

### 1.1.2. Completeness of a List of Scenario Types

Test case generation is applied for each derived scenario type. This means that the underlying list of scenario types — derived from data as suggested above or by experts, e.g. [36, 77, 158, 166, 178] — has to be *complete*. Only then will testing be complete itself and a test

exit criterion can be formulated. However, one can always come up with another scenario type as well as with instances of those types that are different from the types and instances used before. Simply adding a new or unforeseen object to the scenario might already be sufficient to impose a challenge on the system. For instance, simply having a bouncing kangaroo crossing the road has been reported to confuse emergency braking systems [140]. This means an absolute measurement of completeness is infeasible. Fortunately, not all scenario types are equally likely to occur in reality [26]. This fact can be used to statistically assess that a given list of scenario types at least contains all scenario types that take place in real traffic (data) up to a certain likelihood.

Existing works on this topic focus on different aspects around completeness: 1) on clarifying that testing solely with real test drives is not feasible [63, 73, 161, 176]; 2) on analyzing whether sufficient instances for a specific scenario type are recorded in real traffic [38, 39]; 3) on the comparison of a driving system's performance with human driving for specific scenario types. The completeness analysis of a list of scenario types — even relative to real traffic (data) — is left unaddressed.

### 1.1.3. Test Case Generation Using Search-Based Techniques

For each scenario type in the complete list of scenario types, instances have to be generated, which are the final test cases. A common approach is to create parameterized scenarios, where some characteristics of a scenario type are described by parameters to vary different aspects of a scenario like the starting positions of traffic participants. The domains of these parameters span a multi-dimensional space of possible test scenarios, meaning that a multitude of different test cases can be described with one parameterized scenario. Each combination of parameter values defines one such test case. During test case generation, the "good" test cases have to be selected.

The definition of a "good" test case that is used in this work is in the spirit of limit testing and in line with defect-based testing [128]: *A "good" test case can reveal potentially faulty system behavior. That means in a "good" test scenario, a correct system approaches the limits of the safe operating envelope (see [85]), and a faulty system violates them [57].* Inside the envelope, the system is allowed to freely optimize its performance [85], and as long as it does not leave the envelope, it is considered safe. Note that cost-effectiveness is another aspect of "good" test cases [128], which, however, is not in the focus of this work.

Not every test case that is contained in the space of the parameterized scenario is of the desired form, meaning that it contains a desired maneuver like a lane change, or is interesting in that it causes a correct driving system to approach the safety boundaries or causes a faulty driving system to violate them. Identifying the "good" test cases that are of the correct form and challenge the driving system is difficult. Existing works suggest the use of search-based techniques, e.g. [3, 9, 23, 30, 155]. These techniques explore a so-called search space with the goal to find the individual that yields the best fitness value given by a so-called fitness function. Here, the search space is the space spanned by the parameters of the parameterized scenario and the individuals are the test cases. The fitness function

assigns a test case a fitness value, which is a quality measurement of how "good" a test case is: the better the fitness value, the better the test case. Such a fitness function needs to be created with care such that it is able to guide the search technique to the test cases that have the correct form and that challenge the system. A test case of the desired form should be assigned a better fitness value than one without the correct form; a challenging test case of the correct form should yield a better fitness value than a less challenging test case of the correct form. Formulating fitness functions correctly is difficult, time-consuming, and requires experience. Wrongly derived fitness functions leave "good" test cases unidentified, which might even lead to wrong conclusions about the test results. Thus, creating fitness functions ad-hoc, as done in the past, is not sufficient. For the derivation of fitness functions at a large scale, methodological guidance for test engineers is needed.

However, existing work on the search-based testing of driving functions focuses on *technical aspects* or assumes the search space and the fitness function for the test cases to be given, or creates them ad-hoc. Many such presented fitness functions do not yield "good" test cases, because they do not ensure the correct form of the test case or do not yield test cases that challenge the system. This leaves the methodological aspect unconsidered of how the fitness function should be created. Note that there is an exception: In [80], which has been published after this thesis' publication on fitness functions [57], language snippets like "on lane" or "is behind" are provided to create specifications, which are translated to fitness functions ensuring correct scenario form.

### 1.1.4. Re-Use of Scenario Instances

Once "good" test cases have been created, one might have the intention to save time and, thus, might want to re-use them for other versions and variants of the driving system. However, this requires that the quality of the test cases does not drop when re-used, meaning that "good" test cases stay "good". This is in general only possible if the quality of the test case does not depend on the system's behavior. Since different versions and variants of a driving system may perform a maneuver like a lane change at different points in time during a scenario and in different ways, it seems that a test case that is "good" for one system may turn "bad" for another system. This is one perspective. However, many approaches in literature and industry are based on the assumption that scenario instances can generally be re-used [11, 95, 107, 145]. For instance, recording scenario instances in traffic and replaying them as test cases in simulation is an instance of re-using test cases. These perspectives are conflicting and, thus, analysis is required.

There is one work [22] that discusses the general re-use of test cases, which has been published after the publications of this thesis. In [22], it is stated that test cases may yield different results when used in different simulation environments, since simulation environments may use different simulation models.

However, to the best of the author's knowledge there are no existing works that investigate the re-usability of test cases among different versions and variants of automated and autonomous driving systems. This might be because — intuitively — it is difficult to

show that test cases are generally re-usable among different driving systems, meaning that "good" test cases always stay "good" when re-used. However, if there is a "good" test case that turns "bad" when re-used, the general re-usability is disproved by counter-example. An investigation is needed.

## 1.2. Problem Statements and Research Gaps

This work tries to solve two main problems, for which several gaps need to be addressed:

- **Problem 1** is to ensure that the driving system is tested in scenario instances of "all" scenario types. Only then does testing provide the basis for a safety argument that the system can cope with "all" traffic situations. However, identifying a "complete" list of scenario types is infeasible, since one can always find another scenario type. Still, one can strive for completeness by deriving scenario types in methodologically different ways to achieve redundancy, e.g. by complementing the manual scenario type derivation with a data-driven one. This increases the confidence in the list of identified scenario types. Additionally, the resulting list of scenario types needs to be assessed for completeness. Again, an absolute measurement of completeness cannot be given. However, all available information, e.g. collected traffic data, can be used to assess whether the list of identified scenarios is complete relative to the available information. Thus, a data-driven scenario type derivation process is required as well as an approach to assess relative completeness.
  - **Gap 1**: Existing works on clustering recorded traffic scenarios either have technological shortcomings, directly encode the experts mental model into the clustering process, or both. There is a need to reduce the influence of mental models during clustering to yield methodological redundancy with manually derived scenario types, especially with respect to the scenario types' level of granularity and structure.
  - **Gap 2**: Existing works on test exit criteria for testing automated and autonomous driving systems mostly focus on showing that gathering miles is not a feasible test exit criterion. Other works try to use completeness as text exit criterion, but focus on the completeness of identified instances for a single scenario type relative to data. Complementing those works, there is a need to provide the named relative completeness assessment of a list of scenario types.
- **Problem 2** is to ensure that the driving system is tested with "good" test cases, which are instances of scenario types. This requires a definition of what it means that a test case is "good". Such "good" test cases need to be generated, which are of the correct form and challenging. It also requires to understand whether the "good" test cases generally stay "good" when re-used, e.g. for regression testing, since new ones might have to be generated to ensure that the system is still tested in "good" test cases.

- **Gap 3**: Existing works on search-based test scenario generation for automated and autonomous driving systems focus on the technical aspects of the search or assume the fitness function to be given or create it ad-hoc. Especially, ensuring a desired scenario form with the help of fitness functions is left unconsidered. There is a need for methodological guidance for creating fitness functions such that test cases are identified in the search space, which are of the correct form and challenging.
- **Gap 4**: Many existing works implicitly assume that test cases for automated and autonomous driving systems may be created once and re-used for other versions and variants, and in this sense "good" by definition. However, whether this holds is not shown. There is need for analysis whether this is generally possible.

## 1.3. Solution

This work presents a framework for scenario-based testing of automated and autonomous driving systems while considering completeness. Methodological and technical solutions are presented where literature leaves gaps (filled elements in Fig. 1.1): The idea of scenario-based testing is to automatically and manually identify a reasonably small set of relevant dynamic traffic situations, or scenario types; check if the set of scenario types is complete; and then derive "good" system-specific test cases for each scenario type. Numbers in Fig. 1.1 will be referred to in the text.

Initially, experts manually derive (1) a list of scenario types (2). Several sources of information, e.g. specification of and requirements for the system (3) as well as their mental model (4) serve as input for this manual derivation. In parallel, automated clustering (5) of scenario instances that are contained in real driving data (6) is performed. As it is typical for a data-driven approach, note that this clustering requires representative data of real traffic. Also note that the automated clustering is not fully independent of the expert's mental model, since the expert decided what kind of data is collected. The resulting list of scenario types complements and validates the manually derived scenario types with respect to the level of granularity and the structural adequacy. Using real driving data and a catalog of scenario types as input, a statistical completeness check is performed (7). This completeness assessment is relative to the available real driving data. The more representative the data is for the actual operational design-domain, the more meaningful is this relative assessment. If the catalog is found to be incomplete, further manual and data-driven scenario type derivation may be necessary. Otherwise, the scenario types are used as a basis for the derivation of system-specific test cases. For each scenario type, a parameterized scenario is derived (8) based on scenario type description as well as scenario parameter types (9) that are used to vary characteristics of the scenario, e.g. starting positions of traffic participants. Those might be provided by experts or careful system analysis. Similarly, based on the scenario type description, a fitness function is created (10) that ensures that challenging test cases with correct form are identified. Both the parameterized scenario (11) and the fitness

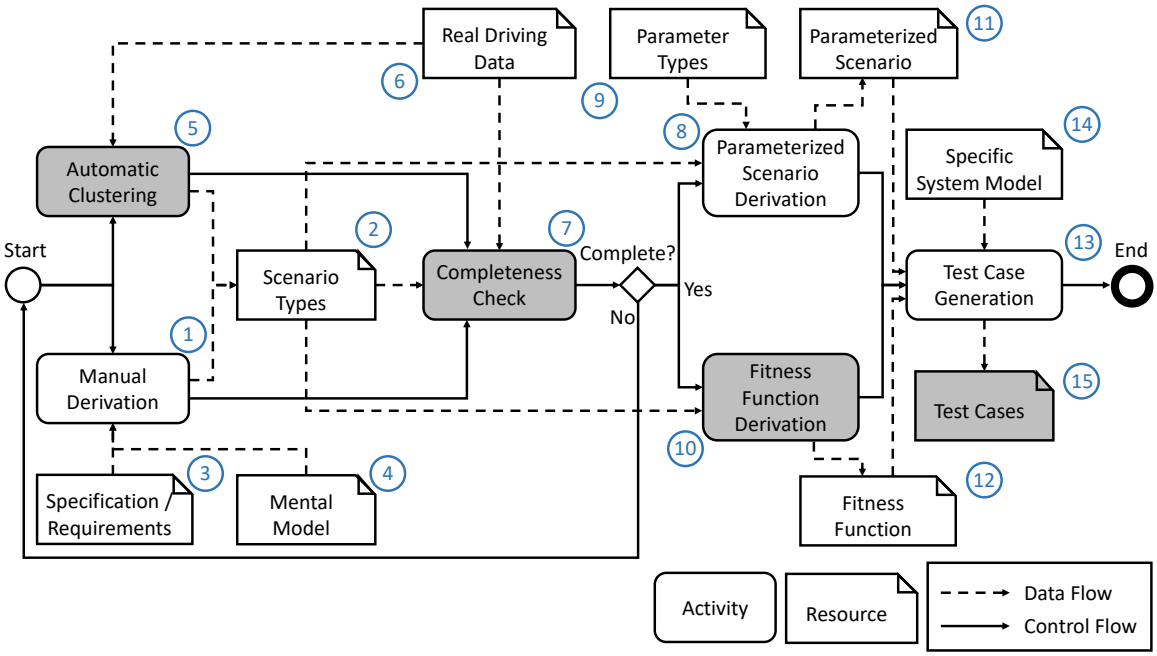



Figure 1.1.: Big Picture (previous versions appeared in [55, 59])

function (12) are the input to the test case generation (13). Using a system specific model (14) for simulation, the test case generation yields "good" test cases (15).

This framework is intended to yield the *basis for safety argumentations*. This work provides the means to derive for scenario types such a fitness function that measures the quality of a test case, i.e. measuring whether the test case has the correct form and how close a correct system approached the safe operating boundaries and by how far a faulty system surpassed those. A search-based technique will try to optimize this quality, meaning it searches for the test case (of the correct form) in which a correct system gets closest to the safety boundaries or in which a faulty system violates those boundaries the most. If the driving system under test is still safe in these system-specific extreme situations, meaning it stays within the boundaries of the safe operating envelope, one can assume that it is also safe in all other test cases of this scenario type's parameterized scenario. Repeating this argument for all scenario types yields the basis for an overall safety argument. To improve the completeness of the known list of scenario types, automated derivation of those is suggested. To assess the completeness of the known list of scenario types, a statistical model is presented.

## 1.4. Contributions

This work makes the following contributions (filled activity boxes in Fig. 1.1):

- To fill **Gap 1** and provide a solution for (5), a clustering method is presented that tries

to keep the influence of expert mental models as low as possible. Recorded traffic data in the form of time series is clustered based on purely syntactical features to derive clusters that represent scenario types. This provides methodological redundancy to the scenario types manually derived by experts, thus aiming at complementing and validating expert efforts with respect to granularity and structural adequacy of the scenario types.

- To fill **Gap 2** and provide a solution for (7), a statistical model is presented that helps assessing the completeness of a list of scenario types relative to a given data set. The model is based on the so called Coupon Collector's Problem. From real data, a histogram of occurrence probabilities of scenario types is derived, which serves as input to the Coupon Collector's Problem. The model yields a statistical statement whether a sufficient amount of data is collected and, therefore, all scenario types are known for this data set.

- To fill **Gap 3** and provide a solution for (10), a set of fitness function templates for search-based test scenario generation are presented. The templates can be used for both (i) to ensure that the generated scenario instances contain the desired maneuvers and (ii) to specifically test against the boundaries of a safe operating envelope. The templates enable measuring the quality of a test case and the quality of the system's behavior in a test case. Thus, the templates provide methodological guidance to experts for the creation of suitable fitness functions for generation of "good" test cases.

- To fill **Gap 4** and potentially re-use test cases (15), an experiment is presented that shows that "good" test case are generally not re-usable. "Good" test cases for different system variants are generated and "re-used" for the other variants. By measuring the test case quality and analyzing the ability to reveal faulty behavior, it is analyzed whether test-cases can generally be re-used.

## 1.5. Publications

As part of this publication-based doctoral thesis, these contributions have previously appeared in the following peer-reviewed publications published in proceedings of international conferences:

- Addressing **Gap 1**: **Florian Hauer**, Ilias Gerostathopoulos, Tabea Schmidt, Alexander Pretschner: Clustering Traffic Scenarios Using Mental Models as Little as Possible, IEEE Intelligent Vehicle Symposium (IV), 2020 ([54])

- Addressing **Gap 2**: **Florian Hauer**, Tabea Schmidt, Bernd Holzmüller, Alexander Pretschner: Did We Test All Scenarios for Automated and Autonomous Driving Systems?, IEEE Intelligent Transportation Systems Conference (ITSC), 2019 ([59])

- Addressing **Gap 3**: **Florian Hauer**, Alexander Pretschner, Bernd Holzmüller: Fitness Functions for Testing Automated and Autonomous Driving Systems, International Conference on Computer Safety, Reliability and Security (SafeComp), 2019 [57]
- Addressing **Gap 4**: **Florian Hauer**, Alexander Pretschner, Bernd Holzmüller: Re-Using Concrete Test Scenarios Generally Is a Bad Idea, IEEE Intelligent Vehicle Symposium (IV), 2020 [56]

## 1.6. Structure

The remainder of this work is structured as follows: Chapter 2 provides an overview of the background of this work. Afterwards, the solutions to the presented gaps are described in Chapters 3-7 in the order of the respective gaps (automated clustering of scenario types, statistical assessment of completeness, derivation of fitness functions, analysis of general reusability of test cases). A discussion of existing works is conducted in Chapter 8, including both a description of related work covering the non-filled steps in Fig. 1.1 as well as a description of the state of the art to which the approaches of this work (filled steps in Fig. 1.1) are compared to. This work is concluded in Chapter 9, containing a discussion of this work's results and an outlook of future work.

# 2. Background and Preliminaries

*This chapter provides a general overview over the background of this thesis. It describes the fundamentals of automated driving systems and their use case. Further, test goals are described, e.g., safety as defined in the literature. Parts of this chapter have previously appeared in peer-reviewed publications [54, 56, 57, 59], co-authored by the author of this thesis.*

## 2.1. Driving Systems

Driver assistance systems, automated, and autonomous driving systems support the driver and partially or completely take over the driving task. This means the driving systems take over longitudinal and / or lateral steering, maneuvering the vehicle from the origin to the target location while respecting the traffic rules and ensuring safety. The semantic of *safety* in scenario-based testing is discussed below in Sec. 2.2.3.

### 2.1.1. Levels of Automation

The degree to which driving systems take over the driving task is divided into so called levels of automation. The German BASt (Bundesanstalt für Straßenwesen, en. federal highway research institute) names four levels of automation [45]. However, the generally accepted standard are the five levels of automation published later on by SAE [139], which we refer to in this work; those are:

- **No Automation (Level 0)**: "the full-time performance by the human driver of all aspects of the dynamic driving task, even when enhanced by warning or intervention systems"
- **Driver Assistance (Level 1)**: "the driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the human driver performs all remaining aspects of the dynamic driving task"
- **Partial Automation (Level 2)**: "the driving mode-specific execution by one or more driver assistance system of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the human driver performs all remaining aspects of the dynamic driving task"

- **Conditional Automation (Level 3)** (Note that from here onward, the system monitors the driving environment):"the driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task with the expectation that the human driver will respond appropriately to a request to intervene";
- **High Automation (Level 4)** (Note that from here onward, the system itself is responsible for the "fallback performance of [the] dynamic driving task"): "the driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task, even if a human driver does not respond appropriately to a request to intervene"
- **Full Automation (Level 5)**: "the full-time performance by an automated driving system of all aspects of the dynamic driving task under all roadway and environmental conditions that can be managed by a human driver"

Generally speaking, the approaches presented in this work may be applied to systems from all levels. However, the focus of this work lies especially on systems of the level 4 and 5, where the system has no human driver as fallback. Most prominently, the system class *highway pilot* of automation level 4 is used to demonstrate the clustering approach in Chap. 3 and the fitness function templates in Chap. 5.

### 2.1.2. System-Environment Interaction

On an abstract level, driving systems generally follow the MAPE-K structure of autonomic computing [75], slightly adapted by the driving systems literature [125, 151]. A simplified and abstracted overview of the components of the driving system and the environment is presented in Fig. 2.1. The environment, containing among others the vehicle that contains the driving system, driver, road, traffic, and weather, is monitored by sensors, e.g. lidars, radars, cameras, slip sensors, engine speed sensor, and others. The current traffic scene is analyzed, e.g., that a collision is imminent, before a suitable behavior is planned, e.g. an evasion maneuver. The actuators, e.g. (lateral) steering controllers, execute the planned behavior. All those steps are supported by environment and system models, e.g., to estimate the error margin of the sensors, to assign semantics to the sensed traffic scene or to select the correct behavior and control actions based on a physical vehicle model.

Such driving systems are generally tested in closed-loop simulations, meaning that the actions of the system influences the environment, which in turn influences the system's behavior. A commonly used term for such simulation setups is X-in-the-loop (XiL), where X depends on the realism of the setup. In early stages of the development, only a model of the system exist (MiL), while later the complete software (SiL), hardware (HiL) or even the whole vehicle including the driving system (VehiL) [48, 49] is available for testing. Intuitively, in MiL and SiL setups, the hardware is virtual, while in HiL some parts of the system, e.g. the camera [114, 113], or the whole vehicle (VehiL) including the system may be real. The scenario type derivation and completeness methods that are presented

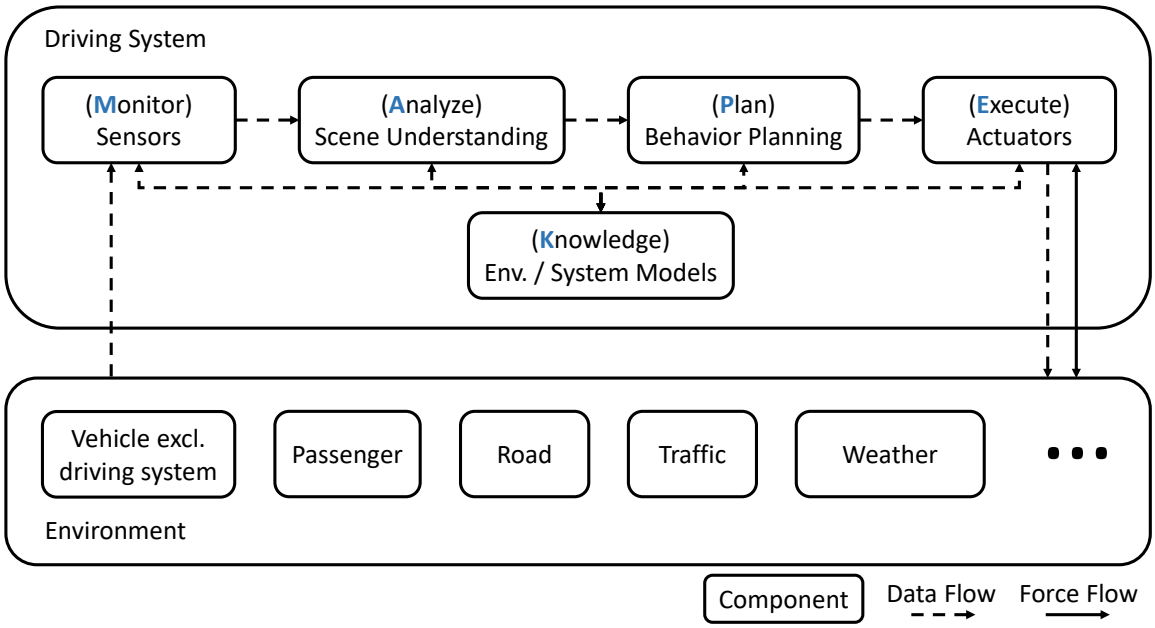



Figure 2.1.: System-Environment Interaction (c.f. [75, 151] about system architectures)

here are fundamental for all setups. However, the search-based test case generation is best suited for completely virtual setups, i.e. MiL and SiL, since search-based techniques require the execution of a high number of test cases. For a limited number of scenario types, an application of search-based test generation may also be feasible in HiL setups.

## 2.2. Terminology Concerning Scenario-Based Testing

A driving system has to provide its functionality in its whole operational design domain (ODD), which is defined as the "operating conditions under which a given driving automation system or feature thereof is specifically designed to function, including, but not limited to, environmental, geographical, and timeof-day restrictions, and/or the requisite presence or absence of certain traffic or roadway characteristics" [139]. For development and testing of automated and autonomous driving systems, the ODD is usually split into micro-ODDs [84] or scenarios types, also called functional scenarios [15, 106, 148] or scenario categories [37].

### 2.2.1. Scenarios for Testing

When using such scenario types for testing, the goal is to mimic real traffic scenarios in simulation to test the safety of the driving behavior of automated and autonomous driving systems. A variety of different scenario types, or functional scenarios [15, 106, 148, 156],

(lane change, emergency brake, etc.) take place in real traffic. Lists of such scenario types are derived from experience [14, 37, 99, 158] and real data [54], and the completeness of such lists is determined with statistical models as presented in this work. For each scenario type, a parameterized scenario is created, also called *logical scenario* [15, 106, 148], for the description of which the literature suggests various formats [8, 51, 68, 131]. The intention is to capture the variability of the real world with $n$ parameters $P$ and their domains $D_j \in D, j = 1..n$, e.g. the initial velocity of other traffic participants in a scenario type is not set to a specific value of $100km/h$, but is assigned a parameter $v_{other}$ with domain $[80, 130]$. The domains span a space $A = D_1 \times D_2 \times ... \times D_n \subset \mathbb{R}^n$ of test cases. Assigning to each parameter a value from its domain yields a single, executable test case, also called *concrete test scenario* [15, 106, 148]. Not every test case in such a space $A$ is of the correct form (e.g. the ego vehicle should perform a lane change, but does not) and among those that have the correct form not all are interesting (e.g. all other vehicles are several hundred meters away from the ego vehicle) [57]. In other words, many test scenarios in such a search space are not the "good" test scenarios we are searching for.

### 2.2.2. "Good" Test Cases

A test case describes the input and environment conditions for simulation. The expected behavior of driving systems is described with the help a safe operating envelope (cf. Fig. 2.2).

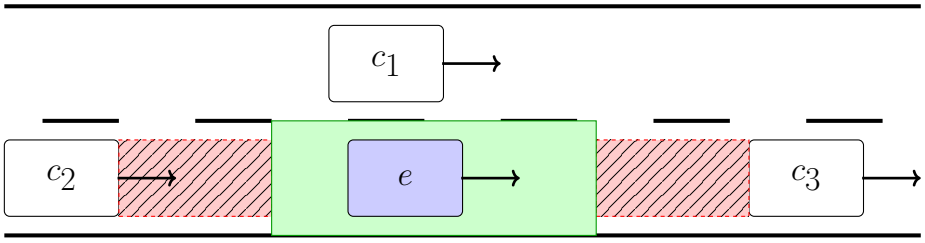



Figure 2.2.: Example of a safe operating envelope (green plain rectangle) bounded by the necessary safety distances (red shaded rectangle) and lane markings

Inside the envelope, the system (inside the ego vehicle $e$) is allowed to freely optimize its performance [85], and as long as the vehicle does not leave the envelope it is considered safe. In the spirit of limit testing, we define **"good" test cases** to test vehicle safety as follows [57, 128]:

> *A "good" test case can reveal* potential *faulty system behavior. In a "good" test scenario, a correct system approaches the limits of the safe operating envelope, and a faulty system violates them.*

### 2.2.3. Safety as a Test Goal

Test goals are the purpose for generating and executing test cases [148]. Ideally, the quality of test cases is measurable according to the test goal's purpose; the system quality becomes assessable, and it allows for a conclusion about the fulfillment of the test goal.

Functional correctness, safety, and compliance with traffic rules are closely intertwined and ensuring them is one of the fundamental requirements for the release of automated and autonomous driving systems [1]. Ideally, the requirement documents, foremost the functional specification, provide information about the functionality of the system. The respective quality criteria are mostly formulated from a safety perspective, considering different subgoals, most importantly safety thresholds for technical and physical properties as well as for safety distances in time (e.g. [47, 72, 160, 172]) and space (see below). Their combination forms a safe operating envelope as described above.

For the experiments presented in the publications of this work, a safety distance in space is used. Those usually imply a threshold, which should not be violated to avoid danger. For the determination of such a minimal safety distance, (inter)national laws may serve as a baseline, e.g. this excerpt from the Vienna Convention on Road Traffic [41], which is also part of the German traffic regulations [42]:

> *"The driver of a vehicle moving behind another vehicle shall keep at a sufficient distance from that other vehicle to avoid collision if the vehicle in front should suddenly slow down or stop."*

In a series of works, an interpretation of this safety distance notion has been derived [124, 133, 134, 135]. Because of space limitation, only a short description is given (cf. Figure 2.3):

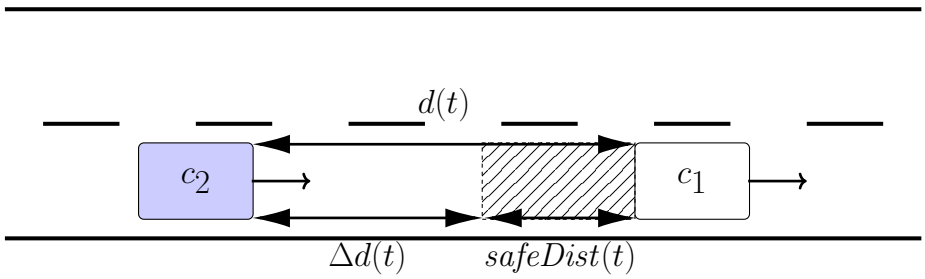



Figure 2.3.: Minimum safety distance at a specific moment in time $t$

For two vehicles $c_i, i \in \{1, 2\}$ with velocities $v_i > 0$ and minimum accelerations $a_i < 0$ and the assumption that $c_1$ drives in front of $c_2$ with distance $d(t)$, this series of works provides case distinctions [134] for checking whether $c_2$ keeps sufficient distance, such that no collision takes place in the case where $c_1$ applies $a_1$ and $c_2$ applies $a_2$ after reaction time $\delta$. According to this notion, we can compute both the minimum safety distance $safeDist(t)$ which $c_2$ has to respect, and the remaining distance until violation $\Delta d(t)$, both for every time step $t$ of a test case.

However, any other appropriate safety distance function may be used as well. Also commonly used are the *Responsibility-Sensitive Safety* (RSS) model [144] or the *Safety Force Field* (SFF) model [120]. Similar to the series of works above, both aim at formalizing traffic rules. This includes not only safety distances for various kinds of scenario types, but also suggestions on how the driving system should behave to restore safety distances if they are violated.

# Part II.

# Methodological And Technical Solutions

# 3. Clustering Traffic Scenarios Using Mental Models as Little as Possible

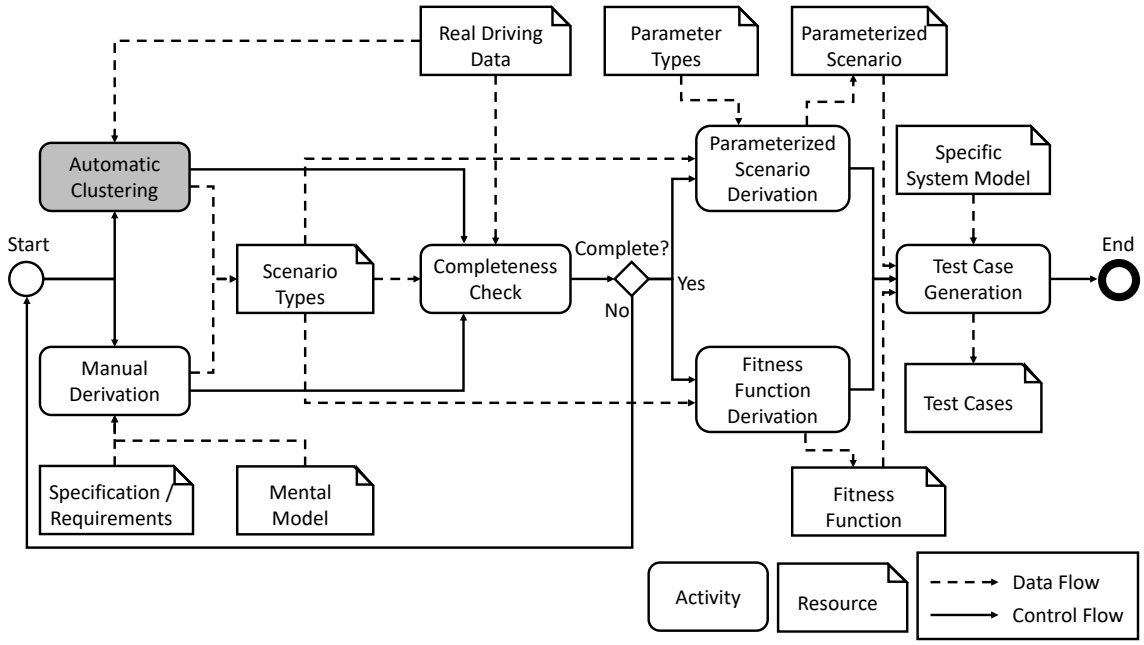



Figure 3.1.: Big Picture (previous versions appeared in [55, 59])

**Summary:** Currently, scenario types are manually derived by experts. To complement and validate their efforts, automated clustering may be performed on scenario instances that have been collected in real traffic. Since the automated clustering should serve as a methodological redundancy to the manual scenario type derivation, it has to be done by relying on the expert's mental model as little as possible.

- Problem: How can recorded scenario instances be clustered to scenario types to complement and validate the manually derived scenario types?
- Gap / Contribution: Existing works are either technically limited, i.e. restrictions in the number of vehicles, the total duration of the scenario, and the type, number, and

length of time series; or heavily depend on the expert, i.e. using handcrafted features or relying on the expert to compute shape and number of clusters.

- Solution: Scenario instances are represented as time series on which the clustering is performed solely based on syntactic features. The number of clusters is determined based on a syntactic metric. This reduces the influence of the expert's mental model to the choice of what kind of data is recorded.
- Evaluation: The presented approach is applied to the highD data set [89] with manual labeling of the resulting clusters.
- Results: For a two-lane highway data set containing 346 scenario instances, 57 clusters representing 38 manually labeled scenario types could be identified. For a three-lane data set with 414 scenario instances, 78 clusters representing 67 scenario types could be identified. The yielded scenario types are found to be more granular than [67, 158, 166] and similar granular as [36, 178].
- Limitations: The approach is not fully independent of the expert, since an expert has to choose what kind of data is recorded in real traffic. The resulting clusters require manual inspection and labeling, which limits the scalability of the approach.

**Author Contribution:** F. Hauer and his supervisor A. Pretschner conceived and discussed the problem statement. The theoretical solution was derived by F. Hauer, who discussed the transfer to a technical solution with I. Gerostathopoulos. During the implementation, F. Hauer was supported by T. Schmidt. The experiments and result analysis were conducted by F. Hauer. The manuscript creation was mainly done by F. Hauer in close discussion with I. Gerostathopoulos and A. Pretschner.

# Clustering Traffic Scenarios Using Mental Models as Little as Possible

Accepted manuscript for the proceedings of the IEEE Intelligent Vehicles Symposium 2020
Published version: https://doi.org/10.1109/IV47402.2020.9304636

Florian Hauer, Ilias Gerostathopoulos, Tabea Schmidt, Alexander Pretschner

***Abstract*— Test scenario generation for testing automated and autonomous driving systems requires knowledge about the recurring traffic cases, known as scenario types. The most common approach in industry is to have experts create lists of scenario types. This poses the risk both that certain types are overlooked; and that the mental model that underlies the manual process is inadequate. We propose to extract scenario types from real driving data by clustering recorded scenario instances, which are composed of timeseries. Existing works in the domain of traffic data either cannot cope with multivariate timeseries; are limited to one or two vehicles per scenario instance; or they use handcrafted features that are based on the mental model of the data scientist. The latter suffers from similar shortcomings as manual scenario type derivation. Our approach clusters scenario instances relying as little as possible on a mental model. As such, we consider the approach an important complement to manual scenario type derivation. It may yield scenario types overlooked by the experts, and it may provide a different segmentation of a whole set of scenarios instances into scenario types, thus overall increasing confidence in the handcrafted scenario types. We present the application of the approach to a real driving dataset.**

## I. Introduction

Automated and autonomous driving systems (ADAS) are commonly tested in simulation using *scenario-based testing*: testing such driving systems in challenging traffic scenarios. These are (extreme) instances of so-called scenario types. Scenario types, or functional scenarios [17], capture recurring traffic situations. One example is a vehicle following another on the right lane of a two-lane highway when both vehicles are overtaken by a third vehicle. During testing, scenario types are used to generate *scenario instances* which vary in different aspects [3], [4], [9]. In the example, different instances may consider different driving speeds or distances between the cars. The goal of scenario-based testing is to identify instances that stress the autonomous driving behavior (e.g., near crashes, abrupt acceleration or deceleration). Proving that the system works as expected in the challenging instances increases confidence in the system. This requires that the list of known scenario types for test case generation is "complete," e.g. as discussed in [10].

The derivation of a "complete" list of scenario types is challenging. A common approach in industry is to have experts manually create such lists of scenario types. However, no matter how comprehensive such lists may be, the manual creation process poses multiple risks: (i) certain scenario types are overlooked, (ii) the mental model, according to which the derivation is done, is inadequate. Using an expert's mental model for scenario type derivation influences the way that scenario types are structured and at which level of granularity they are located. For instance, an expert could derive scenario types according to the existence of maneuvers like braking or lane changing. Similarly, the derivation could be stopped at the granularity level of *contains a lane change* or *contains a lane change to the right in front of another vehicle on the target lane*. Because this manual derivation process necessarily introduces bias, there is an obvious need to validate the results.

An alternative approach is to derive scenario types from real driving data such as [11], [12], [16]. Such recordings of real driving contain a high number of scenario instances, from which scenario types can be derived in an automated way (but which risk missing relevant scenario types [10]). In this work, we present an approach that derives scenario types in an automated way without relying on handcrafted features and that is nearly independent of the mental model of an expert. Note that because a human has to select features and distance measures, it is impossible to *completely* remove any kind of mental model—but we aim at minimizing the introduced bias. Our approach yields scenario types of various levels of granularity, structured independently of an elaborate mental model. Thus, it can be used to evaluate manually derived scenario types w.r.t. completeness and adequacy. Note that a hand-written set of rules, e.g. to detect a lane change in the data, depends on the mental model of the person(s) providing the rules, similar to manual derivation. We hence believe that both manual and automated derivation of scenario types should be executed redundantly.

Existing works have suggested clustering techniques to group recorded driving data according to specific features extracted from each recorded drive. The technical solutions presented in these existing works come with at least one of the following technical limitations (see §V): (i) they are restricted to scenario instances with only one or two vehicles; (ii) they are not capable of handling multivariate timeseries of variable length; or (iii) are restricted to scenario instances of two seconds duration. Thus, such approaches cannot be applied in general to arbitrary scenario instances. Moreover, some approaches make use of handcrafted features that are based on a mental model. For instance, [13] uses one feature which explicitly encodes whether or not a scenario instance contains a braking maneuver. We propose a solution that

F. Hauer, T. Schmidt, and A. Pretschner are with the Department of Informatics at the Technical University of Munich, Germany. (e-mail: {florian.hauer, tabea.schmidt, alexander.pretschner}@tum.de).
I. Gerostathopoulos is with the Faculty of Science at the Vrije University in Amsterdam, Netherlands. (e-mail: i.g.gerostathopoulos@vu.nl).

overcomes such technical limitations of existing works.

Our *contributions* are two-fold. From a *technical* perspective, the presented approach generalizes existing clustering approaches to scenario instances that are composed of any number and any kind of timeseries, containing any number of vehicles, and are of any duration. Thus, technical limitations of existing approaches are overcome. From a *methodological* perspective, the presented approach reduces the dependency on mental models for automated scenario type derivation. This way, it can potentially identify scenario types missed during manual derivation improving completeness, and can increase the confidence in the manual derived scenario types, thus validating the mental model of the experts.

In §II, we introduce scenario-based testing. §III explains the technical details of automated scenario clustering. Experiments and insights are discussed in §IV, followed by a presentation of related work in §V. We conclude in §VI.

## II. Overview of Scenario-Based Testing

The goal of scenario-based testing of ADAS is to subject the driving system to a variety of traffic scenario types. For each type, "good" test cases are generated, which are test cases that can reveal *potential* faulty behavior [9], [21]. Intuitively, the more complete a set of scenario types is, the more convincingly testing can ensure correct system behavior. Currently, experts derive scenario types manually according to their experience in form of a mental model. This process comes with the described shortcomings. Our work aims at the automated derivation of scenario types as depicted in Fig. 1. The goal is to complement manual derivation by potentially identifying scenario types that were overlooked and by increasing the confidence that the manually derived list of scenario types is complete.

An increasing amount of (publicly available) real driving data (1) serves as foundation. The data sets were designed for different purposes and collected in various ways and locations (see [26] for a survey). While some data sets, e.g. [16], are recorded from a single ego-vehicle's perspective, others are created from bird's eye perspective, e.g. [11].

We focus on the automated clustering approach (2), which aggregates real driving data into scenario types. The desired result (3) is a set of clusters, each representing a scenario type. Scenario instances that have the same structure are hence grouped into the same cluster, e.g. several instances where the ego-vehicle performs a lance change to the left behind a decelerating other vehicle. Ideally, there is not

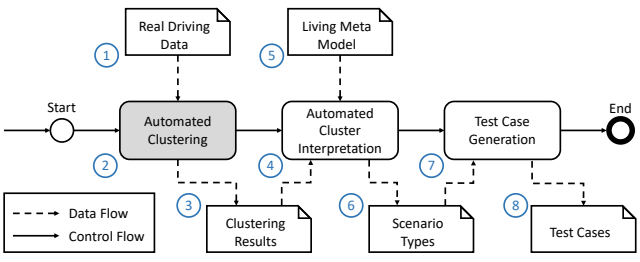



Fig. 1. Automated scenario type derivation for scenario-based testing.

more than one cluster representing the same scenario type. Conversely, clusters that contain scenario instances of two distinct scenario types are not desired either. Determining purity and minimality obviously requires careful inspection. Moreover, automated clustering approaches cluster solely based on *syntactic* features and do not interpret the scenario instances in a *semantic* way, as a human expert would do. The resulting clusters may be perfect in terms of the clustering quality on a *syntactic* level, e.g. measured by silhouette scores; and yet they may not represent the scenario types that a human expert would expect.

As a second step, automated cluster interpretation (4) is applied. It uses a living meta-model (5) for scenarios to interpret clustering results and yield the desired scenario types (6). A starting point for such a living meta model are existing scenario meta models [1], [7], [23]. The idea of a living meta-model for scenarios is to create a meta-model and improve it over time, such that it increasingly resembles the scenario types of real traffic. The meta-model assigns *semantic* meaning to the cluster contents that are merely more than timeseries. For instance, a "lane change" is detected whenever the lateral position of a vehicle exceeds a certain threshold. The cluster may be interpreted as the scenario type shared by most of the scenario instances in the cluster. This process yields a list of scenario types. Ideally, such a list is as complete as possible, in that it contains "all" relevant possible scenario types of real traffic [10].

Finally, for each scenario type in the list, a variety of different approaches for test case generation (7) can be used to generate "good" test cases (8) [8].

## III. Automated Clustering Approach

An overview of the approach is provided in Fig. 2. The first step is data preparation (1). Real driving datasets usually consist of $n$ data segments or scenario instances $d_i, i \in [1, n]$. We assume this segments to be given; a real drive recording of many kilometer length has to be segmented first. Two such scenario instances are shown in Fig. 2, "car following" and "cut-in." A scenario instance $d_i$ consists of a list of $m$ timeseries $ts_{j,d_i}, j \in [1, m]$ that describe the evolution of $m$ object attributes related to an ego-vehicle over time. For instance, a timeseries can be the evolution of the distance $s_{c_1} - s_e$, where $s_e$ is the position of the ego-vehicle and $s_{c_1}$ the position of a preceding vehicle (Fig. 2, top). Since our technique can cope with any number of timeseries $m$, it overcomes the technical limitations of existing works that only allow small or specific numbers. We will assume that each scenario instance consists of the same list of timeseries. The time interval of the $m$ timeseries of a single scenario instance is equally long, but usually differs among scenario instances. This way, we overcome the technical limitations of very short or fixed lengths scenario instances. Depending on the dataset, the number of timeseries (attributes) $m$ may range from a dozen to even a hundred, capturing e.g. the difference in longitudinal and lateral positions, velocities, or accelerations between an ego-vehicle and vehicles on its left lane, its right lane, its own lane, etc.

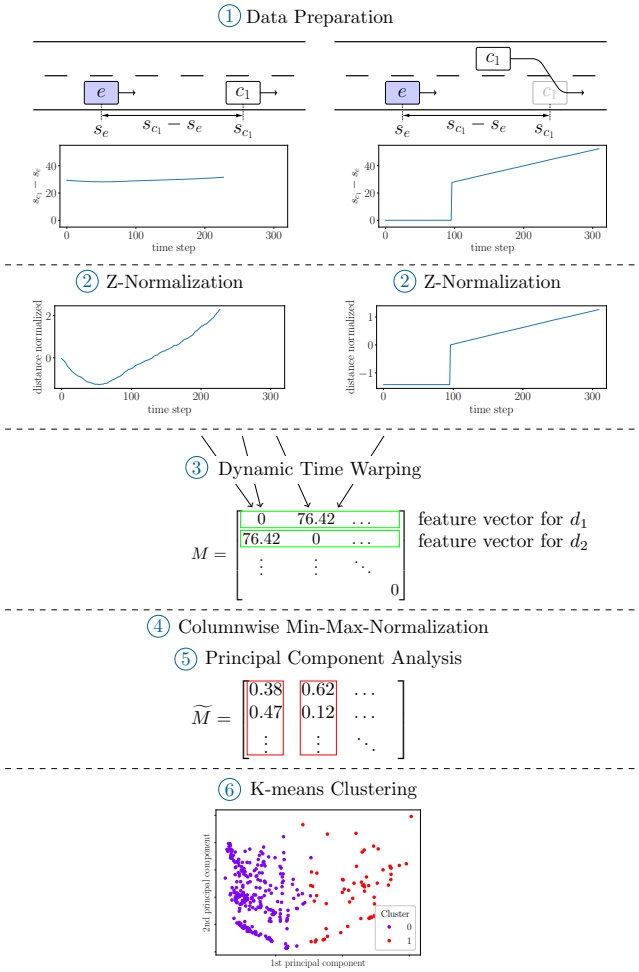



Fig. 2. Overview of the proposed approach.

Once the data is prepared, normalization is applied. We chose z-normalization (2) as suggested in [6], since it emphasizes the structure of the timeseries and neglects the absolute values, desired as described above. For each timeseries $ts_{j,d_i}$, z-normalization is applied individually.

For the computation of the feature vectors used for clustering, we use (3) Dynamic Time Warping (DTW) [19]. DTW is one of several methods to measure the difference between two timeseries. Those timeseries may be of different lengths, and the key structural characteristics may be shifted and stretched over time without affecting the final score of DTW (contrary to e.g. Euclidean distance). We use DTW based on the L1-norm; see e.g. [2]. In our example of Fig. 2, the DTW distance between the timeseries of the two scenario instances is 76.42, while the DTW distance between two identical timeseries is 0. Based on this, a distance vector $q$ between two scenario instances $d_i$ and $d_j$ can be defined as $q_k = DTW(ts_{k,d_i}, ts_{k,d_j})$ with $k \in [1, m]$. This means $q$ contains the pairwise DTW distances of the timeseries of the two scenario instances. In our example, since we used $m = 1$ timeseries of each scenario instance, $q$ has length 1, in particular $q = [76.42]$. The final feature vector of $d_i$ is created by concatenating all distance vectors between $d_j, j \in [1, n]$ and $d_i$. Such a feature vector of a scenario instance can be understood as the difference to all other scenario instances based on the individual timeseries. In total, each feature vector has a length of $n * m$. This results in a feature matrix $M$ of dimension $n \times n * m$.

Using such a similarity measure instead of handcrafting features, our approach intuitively generalizes well. This concept of feature computation—and with that the clustering—does not depend on the mental model of an expert. Moreover, by comparing timeseries in a direct way using DTW, instead of comparing their summary statistics such as moving averages or min-max values, we preserve important patterns that may be lost in aggregation.

The next step, columnwise min-max-normalization (4), ensures that all features are scaled to $[0..1]$. Remember that the feature vectors' length is linear in the number of scenario instances to be clustered. To reduce the dimensionality of the feature vectors and facilitate clustering, we use Principal Component Analysis (PCA) (5), a well-known statistical procedure for dimensionality reduction. For our experiments, the PCA was parameterized to keep 95% of the variance in the features, which resulted in 15 dimensions (§IV).

For the clustering itself, we chose classic k-means (6), since it allows for an easier interpretation of the clustering results compared to other techniques. We also experimented with hierarchical clustering, which yielded very similar results; and density-based clustering, which resulted in clusters of undesired structure and quality and were difficult to interpret. When using k-means, we do not prescribe the number $k$ of clusters. Instead, we run k-means for every $k$ from 2 to the number of scenario instances $n$ and let a state of the art knee/elbow detector [22] choose the best $k$ based on the inertia of the clustering results.

## IV. Experiments

### A. The highD Dataset

We applied our approach to the highD dataset [11], which is just one dataset containing well-structured highway traffic data. The data was recorded from a bird's-eye perspective with the help of a drone-mounted camera. Fig. 3 shows an exemplary picture. Each vehicle's trip from end to end of the field of view of the recording camera corresponds to one scenario instance with this vehicle as ego-vehicle.

*1) Data Preparation:* Surrounding vehicles may be very far away from the ego-vehicle due to the recording in bird's-eye perspective. Therefore, we apply pre-processing to the data in form of a *range of interest*. Other vehicles outside of this range are not considered to be neighbors of the ego-vehicle. This region of interest should be the maximum range at which other cars still influence the scenario type.

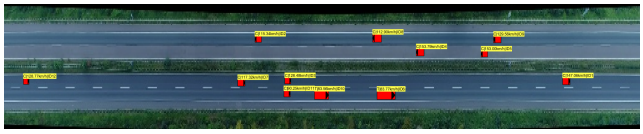


Fig. 3. Exemplary image of the highD dataset

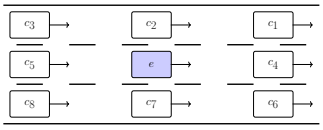


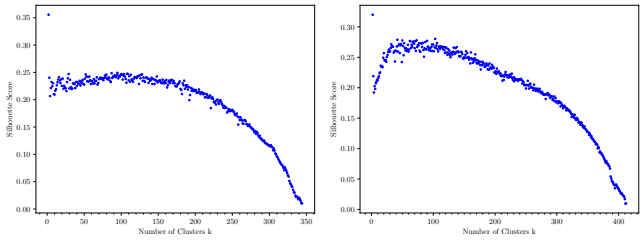

Fig. 4. Eight car model for environment modelling of the ego-vehicle

We chose 60m with the intuition that the scenario type from the perspective of the ego-vehicle mainly depends on the next and not so much on the second next car ahead or behind. Another pre-processing step is to filter the scenario instance for those where the ego-vehicle is of type "car," since we are interested in scenario types to test automated and autonomous driving systems for passenger cars. Trucks and other vehicle types may still be part of the environment.

*2) Choosing Relevant Timeseries:* There are several kinds of timeseries in the data, including longitudinal and lateral positions as well as velocities, and meta data, e.g. vehicle type. Following the intuition of the *range of interest*, we consider the eight cars around the ego vehicle at every time step (Fig. 4) similar to [11]. Based on the available data, we computed the longitudinal and lateral distances from the ego-vehicle to those eight other vehicles. As long as there is no such other vehicle at one of those eight positions, the respective timeseries is set to 0. This results in $m = 2 * 8$ timeseries per scenario instance. Note that the eight car model and the choice of time series can be understood as a mental model. We argue, however, that this choice constitutes a minimalistic model. In principle, the presented technique can be applied to more cars and more timeseries to further reduce the influence of the mental model. Scenario instances are of different lengths, since vehicles pass the field of view with different velocities. The number of time steps varies between 200 for fast vehicles and 300 for slow ones.

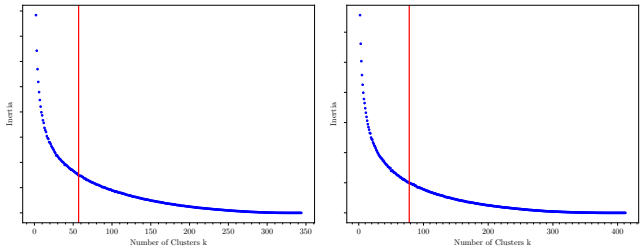
### B. Experiment Results

We cluster the data for a two-lane and a three-lane recording, monitored over stretches of 420m, containing 346 and 414 scenario instances respectively. Clustering is done for every number of clusters $k$ from two to the number of scenario instances $n$. A common way to identify the best $k$ is the one with the highest so-called silhouette score [24]. Fig. 5 shows the silhouette scores for both datasets and all $k$. The maximum silhouette scores for both datasets is at $k = 2$, which is not surprising: In the two-lane dataset, all scenario instances have either other vehicles on the left or on the right. Clearly, the best clustering is to divide the scenario instances into two clusters, one for driving on each lane. Potential lane changes are assigned to the cluster of the two lanes on which the ego-vehicle drives for a longer duration. For most of the scenario instances of the three-lane data, this explanation still holds. Even though the best $k$ equals 2, this is not a helpful clustering result; we therefore seek the next best $k$. However, for both datasets, a wide range of $k$ provide similar silhouette scores, making a clear decision impossible. Therefore, we rely on the elbow (or *knee*) method [18] to identify a good $k$. We apply this method to the inertias of the




(a) Two lane data (b) Three lane data

Fig. 5. Silhouette score for each number of clusters in $[2..n]$




(a) Two lane data (b) Three lane data

Fig. 6. Inertias for each number of clusters in $[2..n]$; red line is the chosen number of clusters by the kneedle algorithm [22]

clustering results (Fig. 6) and let the state-of-the-art elbow detector Kneedle [22] identify the elbow, providing us with $k = 57$ for two-lane data and $k = 78$ for three-lane data.

To understand the experiment results, we manually inspected the 57 and 78 clusters and interpreted the clusters as the scenario types that are shared by most of the instances contained. Intuitively, one would watch the video recordings to manually interpret the cluster representatives or run an automated interpretation. We cannot present videos in this paper nor can we show all the scenario type descriptions of all clusters. Instead, we present the 16 timeseries of one exemplary cluster of the two-lane data in Fig. 7. Additionally, we provide the clustering results for the presented experiments online.[1] For the two-lane case, the 57 clusters represent, upon manual inspection, 38 different scenario types. $38 < n = 57$ is a result of the nature of traffic data: Some scenario types are very rare compared to others, which leads to more than one cluster for a single common scenario type. For the three-lane data, we manually identified 67 distinct scenario types among the 78 clusters.

### C. Threats to Validity

All experiments face threats to validity. We applied the presented approach to the highD dataset, which is limited in terms of data diversity, since the data is recorded on a straight highway section of 420m length without ramps. To transform the data from bird's-eye perspective to ego-vehicle perspective, we applied the discussed *range of interest*. By inspection of the data, we chose a suitable value, which might not be perfect to identify scenario types. For the clustering, longitudinal and lateral distances are experimentally selected

[1] https://drive.google.com/open?id=1JApX49mbT-zULq3uFmiRRm4ja5SWFG23

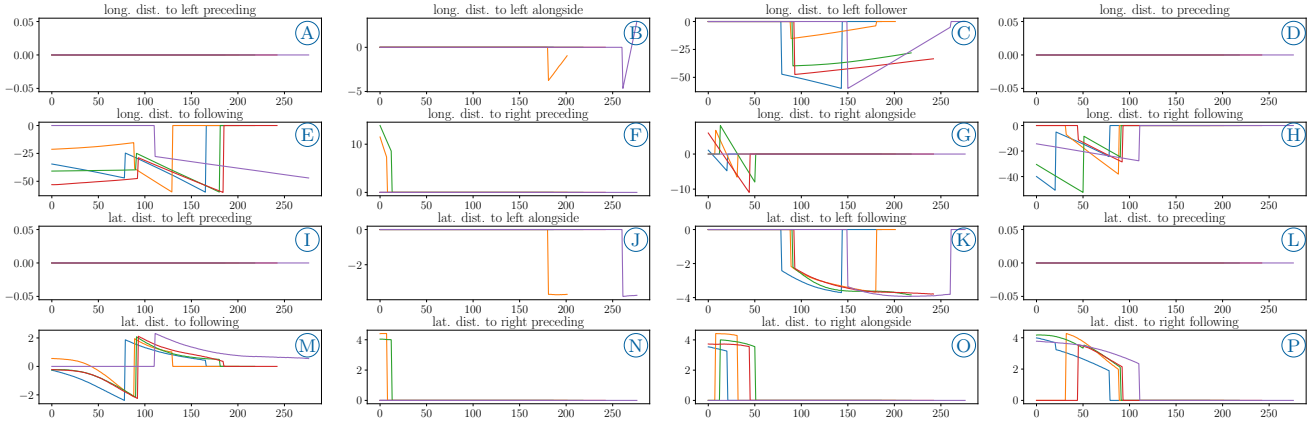



Fig. 7. Shown are the 16 timeseries of five scenario instances that are contained in a cluster of the two-lane dataset. The five scenario instances are plotted in blue, orange, red, green, and purple. The scenario instance plotted in blue is a good representative for this cluster. In this scenario instance, the ego-vehicle drives on the left lane with a vehicle behind it, as seen in plot (E). Then, the ego-vehicle drives by another vehicle on the right lane, which can be seen in the plots (G) and (H). Finally, the ego-vehicle performs a lane change to the right in front of the overtaken vehicle, which is indicated by the jump from -2 to 2 in plot (M). This behavior defines the scenario type for this cluster.

as information source. It might be that there exists a better set of timeseries. Similarly, there might be more suitable clustering techniques than k-means, even though we experimentally found k-means to perform better than others. The quality of clustering results strongly depends on the correct number of clusters. We used the elbow method, which leaves room for interpretation. We tried to mitigate eye-balling by using the Kneedle [22] algorithm. However, there might still be better numbers of clusters. The final experiment results have been analyzed and interpreted manually.

### D. Discussion

Our feature vectors (§III) are defined solely by the distance between one scenario instance and all other scenario instances, based on the individual timeseries. These features hence do not encode any further "semantics." The clustering depends on the timeseries data only. Information that is not contained in the timeseries data cannot impact clustering: scenario instances cannot be grouped according to this missing information, and missing scenario types cannot be found. For instance, if weather conditions should intuitively impact the resulting clusters but weather information is not input to the clustering algorithm, then the resulting scenario types cannot distinguish different weather conditions, unless of course this weather information correlates with other information provided as input (in this case, however, "weather" cannot be identified as a relevant feature). The choice of data in time series hence constitutes a mental model. Similarly, we are aware that the eight-car-model constitutes a mental model, but argue that it encodes a minimum amount of information needed for clustering.

Selecting the number of clusters $k$ can be understood as the selection of a distance threshold up to which scenario instances are put into the same cluster. Choosing more or fewer clusters allows the adjustment of the granularity of the resulting scenario types. We let the Kneedle algorithm [22] automatically perform this choice to avoid further bias.

The resulting scenario types together with the provided level of granularity are meant to provide redundancy w.r.t. the scenario types (& granularity) of the experts' manual derivation. This raises the question of what the "correct" level of granularity is for testing automated and autonomous driving systems, since it is crucial for the safety argumentation. Manual derivation of scenario types relies on the correctness of the mental model of the expert performing this derivation, which motivates the need for redundancy. Our work provides a perspective on scenario types that is barely influenced by mental models and can be used to identify further scenario types and to validate the scenario types yielded by manual derivation, both in terms of correct granularity and completeness.

## V. Related Work

A multitude of existing works are concerned with clustering timeseries; see [15] for an overview.

In the domain of traffic engineering, the goal is to understand the usage and demand of the road network [2] or of single road sections [5]. Both works cluster two-dimensional GPS position timeseries of individual vehicle trips from start to destination. Since the position timeseries of a single vehicle does not contain information about surrounding vehicles and since such a trip is composed of a multitude of scenario instances, their approach is not suitable to cluster single scenario instances with traffic interactions to scenario types. From a technical perspective, their approaches are limited to the two-dimensional GPS position timeseries.

In [14] and [25], "driving encounters" between two vehicles are clustered based on vehicle position trajectories. The approach yields four [14] and ten [25] clusters where one cluster contains driving encounters at crossings, another clusters contains driving encounters where vehicles approach each other on opposing lanes, and so on. Arguably, this level of granularity does not provide a sufficient level of detail to yield fine-grained scenario types. For instance, it ignores the various different ways how two or more vehicles may interact at a crossing. From a technical perspective, the approach is

limited to two-vehicle interactions while in reality scenario instances take place with more than two vehicles.

In [20], an approach is presented that clusters collision data to identify different types of collisions. Abstract, categorical information is used as input for the clustering, e.g. the gender of the driver or three categories of injuries. Intuitively, this approach is limited to the specific use case of collision data composed of categorical data. It is not applicable to (non-collision) driving data presented as timeseries.

The goal of [13] is to extract traffic scenario types from simulated driving data. The approach is limited to scenario instances of two seconds' length and interactions between two vehicles. The clustering is based on handcrafted features, such as aggregations and characteristic points within the timeseries, e.g. whether or not a braking maneuver took place as well as the velocity of both vehicles at the start and at the end of the two second time span. In reality there are traffic scenarios with (i) more than two interacting vehicles and (ii) usually such scenarios are longer than two seconds. Further, handcrafted features come with the discussed shortcomings.

In sum, this paper closes the following gap: It overcomes technological limitations, i.e. restrictions in the number of vehicles, the total duration of the scenario, and the type, number, and length of timeseries. Moreover, our approach clusters without handcrafted features and, thus, arguably relies on a minimum mental model of an expert.

## VI. Conclusions

We motivated the need for scenario types by test scenario generation, highlighting that the current manual derivation by industrial experts poses a risk of incompleteness and inadequacy. We proposed an automated clustering approach to extract scenario types from real driving data. It solely relies on the difference between the timeseries of recorded scenario instances. We applied the presented approach to the highD dataset [11]. The presented experiment results show the application of the clustering to both a two-lane and a three-lane highway recording. We discussed how the presented feature creation allows the clustering to yield different levels of granularity, and how the clusters are influenced by the choice of data, and distance measures according to the eight-car-model, used for clustering. We have argued that the partitioning of the scenario instances into types is barely influenced by a mental model and the level of granularity is not pre-set by an expert. As the clustering results can therefore be used to evaluate handcrafted scenario types, this makes the presented approach valuable from a methodological perspective, in addition to overcoming the technical shortcomings of existing works. However, further research is necessary to understand what an adequate level of granularity is for scenario types. It heavily influences the test case generation and, therefore, also the overall assessment of the driving system. We believe that the presented approach is a first step in this direction.

## VII. Acknowledgements

This work was supported by the Intel Collaborative Research Institute "Safe Automated Vehicles."

## References

[1] J. Bach, S. Otten, and E. Sax. Model based scenario specification for development and test of automated driving functions. In *IEEE Intelligent Vehicles Symposium*, pages 1149–1155, 2016.

[2] P. Besse, B. Guillouet, J.-M. Loubes, and F. Royer. Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Trans. on Intelligent Transportation Systems*, 17(11):3306–3317, 2016.

[3] A. Calo, P. Arcaini, S. Ali, F. Hauer, and I. Fuyuki. Generating avoidable collision scenarios for testing autonomous driving systems. In *IEEE Intl. Conf. on SW Testing, Verification and Validation*, 2020.

[4] A. Calo, P. Arcaini, S. Ali, F. Hauer, and I. Fuyuki. Simultaneously searching and solving multiple avoidable collisions for testing autonomous driving systems. In *Genetic and Evolutionary Computation Conference*, 2020. to appear.

[5] M. Y. Choong et al. Modeling of vehicle trajectory clustering based on lcss for traffic pattern extraction. In *IEEE International Conference on Automatic Control and Intelligent Systems*, pages 74–79, 2017.

[6] D. Goldin and P. Kanellakis. On similarity queries for time-series data: constraint specification and implementation. In *Intl. Conf. on Principles and Practice of Constraint Prog.*, pages 137–153, 1995.

[7] L. Hartjen, F. Schuldt, and B. Friedrich. Semantic classification of pedestrian traffic scenarios for the validation of automated driving. In *IEEE Intelligent Transp. Systems Conf.*, pages 3696–3701, 2019.

[8] F. Hauer, B. Holzmüller, and A. Pretschner. Re-using concrete test scenarios generally is a bad idea. In *IEEE Intelligent Vehicles Symposium (IV)*, page to appear, 2020.

[9] F. Hauer, A. Pretschner, and B. Holzmüller. Fitness functions for testing automated and autonomous driving systems. In *Intl. Conf. on Computer Safety, Reliability, and Security*, pages 69–84, 2019.

[10] F. Hauer, T. Schmidt, B. Holzmüller, and A. Pretschner. Did we test all scenarios for automated and autonomous driving systems? In *IEEE Intelligent Transportation Systems Conf.*, pages 2950–2955, 2019.

[11] 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 *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2118–2125, 2018.

[12] A. Krämmer, C. Schöller, D. Gulati, and A. Knoll. Providentia - a large scale sensing system for the assistance of autonomous vehicles. *Robotics Science and Systems Workshops (RSS Workshops)*, 2019.

[13] F. Kruber, J. Wurst, and M. Botsch. An unsupervised random forest clustering technique for automatic traffic scenario categorization. In *IEEE Intelligent Transp. Systems Conf.*, pages 2811–2818, 2018.

[14] S. Li, W. Wang, Z. Mo, and D. Zhao. Cluster naturalistic driving encounters using deep unsupervised learning. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1354–1359, 2018.

[15] T. W. Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[16] W. LLC. Waymo open dataset. online at `https://waymo.com/open/`, retrieved 6th December 2019, 2019.

[17] T. Menzel, G. Bagschik, and M. Maurer. Scenarios for development, test and validation of automated vehicles. *arXiv:1801.08598*, 2018.

[18] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, June 1985.

[19] M. Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[20] P. Nitsche, P. Thomas, R. Stuetz, and R. Welsh. Pre-crash scenarios at road junctions: A clustering method for car crash data. *Accident Analysis & Prevention*, 107:137–151, 2017.

[21] A. Pretschner. Defect-based testing. In: Dependable Software Systems Engineering, 2015.

[22] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan. Finding a kneedle in a haystack: Detecting knee points in system behavior. In *IEEE Intl. Conf. on Distr. Comp. Systems Workshops*, pages 166–171, 2011.

[23] J. J. So, I. Park, J. Wee, S. Park, and I. Yun. Generating traffic safety test scenarios for automated vehicles using a big data technique. *KSCE Journal of Civil Engineering*, 23(6):2702–2712, 2019.

[24] A. Starczewski and A. Krzyżak. Performance Evaluation of the Silhouette Index. In *Artificial Intelligence and Soft Computing*, pages 49–58. Springer International Publishing, 2015.

[25] W. Wang, A. Ramesh, and D. Zhao. Clustering of driving scenarios using connected vehicle datasets. *arXiv:1807.08415*, 2018.
[26] H. Yin and C. Berger. When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets. In *IEEE Intelligent Transportation Systems Conf.*, pages 1–8, 2017.

# 4. Did We Test All Scenarios for Automated and Autonomous Driving Systems?

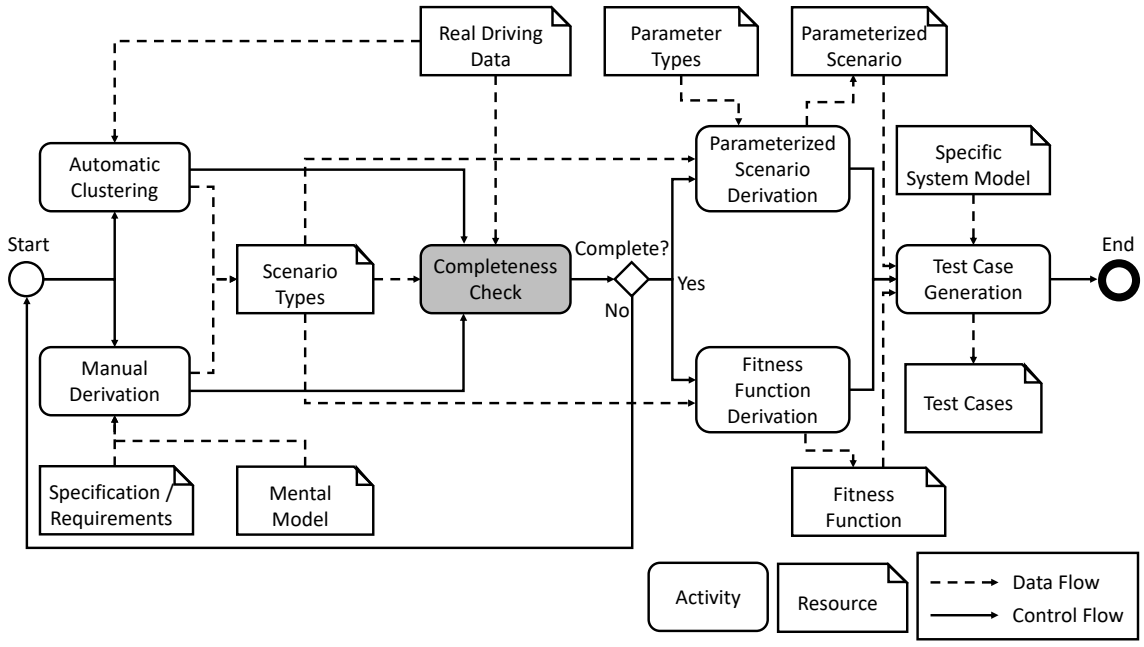



Figure 4.1.: Big Picture (previous versions appeared in [55, 59])

**Summary:** Test case generation is applied for each relevant scenario type. This means that the underlying list of scenario types — derived from data (see Sec. 1.1.2) or by experts — has to be *complete*. Only then will testing be complete as well and a test exit criterion can be formulated. A completeness assessment of the list of scenario types is required. An absolute completeness measurement is infeasible, but an assessment relative to a given real traffic data set is possible.

- Problem: How to assess a list of scenario types for completeness relative to a given data set?
- Gap / Contribution: Existing works either focus on the fact that testing solely with real test drives is not feasible or lack an argument that all scenario types are known.

- Solution: A statistical model based on the Coupon Collector Problem is presented that takes a histogram of occurrence probabilities (derived from traffic data) as well as an occurrence probability up to which all scenario types should be known and states whether sufficient amounts of data has been collected in traffic. If so, it follows that all scenario types up to the provided occurrence probability are known for a given data set.
- Evaluation: The model is applied to one histogram of occurrence probabilities from literature [175] and three artificial distributions.
- Results: The experiment results show the applicability of the model to measure the completeness of a list of scenario types. The experiment results indicate that the amount of necessary collected scenario instances is practically feasible. For instance if one would like to ensure that all scenario types with an occurrence probability of at least 0.00001 are known, the model yields numbers of required collected scenario instances roughly between 450,000 and 500,000 for the given histograms of occurrence probabilities.
- Limitations: The solution idea heavily relies on a representative histogram of occurrence probabilities, the derivation of which requires representative data of real traffic. The model is influenced by the level of granularity of the scenario types.

**Author Contribution:** F. Hauer, his supervisor A. Pretschner, and the industrial partner B. Holzmüller conceived the problem statement. F. Hauer and T. Schmidt — in close discussion with A. Pretschner — derived the final model. Implementation and experiment execution was shared between F. Hauer and T. Schmidt. The manuscript creation was done by F. Hauer and T. Schmidt, supported by A. Pretschner.

# Did We Test All Scenarios for Automated and Autonomous Driving Systems?

Accepted manuscript for the proceedings of the IEEE Intelligent Vehicles Symposium 2020
Published version: https://doi.org/10.1109/ITSC.2019.8917326

Florian Hauer*, Tabea Schmidt*, Bernd Holzmüller and Alexander Pretschner

***Abstract*— To ensure safety and functional correctness of automated and autonomous driving systems, virtual scenario-based testing is used. Experts derive traffic scenario types and generate instances of these types with the support of test generation tools. Since driving systems operate in a real-world environment, it is always possible to find a new scenario type as well as new instances of scenario types that are different from all other scenario types and instances. Thus, the testing process to find faulty behavior may continue forever. There is a practical need for test ending criteria for both of the following problems: Did we test *all scenario types*? Did we sufficiently test *each type with specific instances*? We address the first question and present a suitable test ending criterion and methodology. Whether the system is tested in each scenario type is reduced to the question whether *all* test scenarios are known. We analyze driving data to provide a statistical guarantee that *all* scenario types are covered. We model this as a Coupon Collector's Problem. We present experimental results for the application of this model to different driving tasks of automated and autonomous driving systems.**

## I. Introduction

Striving for highly automated and autonomous driving systems results in ever more complex and capable systems. Due to the complexity of these systems and the complexity and sheer number of possible traffic scenarios, ensuring safety and functional correctness is a crucial challenge [1]. Since verification and validation by real test drives alone become practically infeasible [2], [3], [4], the focus is currently shifting to virtual test drives. For virtual testing, scenario-based closed-loop testing in the form of X-in-the-Loop settings is used [5]. This means, that the automated or autonomous driving system, e.g. in form of a model, software, or hardware is tested in a simulated traffic scenario, where the system behavior affects the behavior of the simulated environment.

Automated and autonomous driving systems must provide their functionality in every possible scenario of the potentially infinite number of scenarios. These scenarios can be clustered into scenario types, e.g. "free drive", "lane change", "cut-in", or "emergency braking"; [6], [7] present related catalogs of scenario types. One can always come up with another scenario type as well as with instances of those types that are different from the types and instances used before. Simply adding a new object to the scenario might already be sufficient to impose a challenge on the system as the example of a kangaroo shows [8]. Intuitively, there is the strong need for a test ending criterion, which can be defined by addressing the two following issues. Firstly, we need to judge whether *all* scenario types that occur in real traffic are known to us. Secondly, we have to decide whether we tested each of those types sufficiently. Approaches that answer these questions have to provide statistical guarantees for safety argumentation and certification while still being applicable in practice. Such a test ending criterion would greatly benefit the release process of automated and autonomous vehicles, since resources can be used more effectively to fulfill the test ending criteria.

The **contribution of this paper** is the following: We provide an approach to the first part of the described test ending criterion for automated and autonomous driving systems. The question whether *all* scenario types that occur in real traffic are tested is modeled as an instance of the Coupon Collector's Problem. The resulting model receives statistical data from real driving data as input and yields an answer to the question as output.

§II introduces scenario-based testing. We recap the Coupon Collector's Problem in §III in order to model the test ending criterion accordingly in §IV. §V provides insights from experiments. We discuss related work in §VI and conclude in §VII.

## II. Scenario-Based Testing

Testing aims both to (1) gain confidence that functionality was implemented correctly (requirements-based testing) and (2) provoke failures (defect identification). In software testing, test case selection is usually done by partitioning the input domain, and then picking or generating a few inputs (i.e., test cases) for each block of the partition. If the purpose is requirements-based testing, blocks are chosen w.r.t. common functionality, which are driving tasks, or scenario types, in our case. If the purpose is defect identification, then blocks are chosen w.r.t. some defect hypothesis. For instance, such a hypothesis may state that failures are more likely to occur at the boundaries of relevant intervals; or that specific weather conditions negatively impact the accuracy of sensors. Either way, the intuition is that the blocks should exhibit "similar" behavior in terms of (1) requirements and/or (2) the same class of potential failures they provoke. Partition-based testing, understood as requirements-based testing, is

*: Alphabetically ordered

F. Hauer, T. Schmidt, and A. Pretschner are with the Department of Informatics at the Technical University of Munich, Germany. (email: {florian.hauer,tabea.schmidt,alexander.pretschner}@tum.de).

B. Holzmüller is with ITK Engineering, Germany, e-mail: bernd.holzmueller@itk-engineering.de

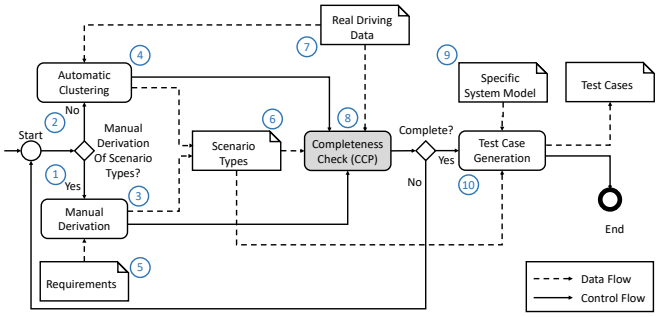



Fig. 1. Big Picture

precisely what we advocate here: the scenario types define the blocks of the partition of the input domain. These scenarios types are then used to generate scenario instances, possibly in a defect-based way.

### A. Big Picture

The idea of scenario-based testing is to automatically or manually identify a reasonably small set of relevant dynamic traffic situations, or scenario types; check if the set of scenario types is complete; and then derive system-specific tests for each scenario type. While this paper is concerned with checking completeness of scenario types only, we need to explain the big picture in Fig. 1.

Initially, we choose between manual (1) and automatic (2) identification of scenario types. For manual identification (3), several sources of information (5), e.g. requirements, safety analysis, functional specifications, and traffic rules are used for the identification of scenario types (6). For automatic identification of scenario types (4), automated clustering techniques are applied to a subset of pre-recorded real driving data (7), e.g. suggested in [9]. Note that depending on the distance measure used by the clustering algorithm, the automatically identified clusters need not necessarily correspond to scenario types that a human would identify with typical recurring traffic situations. Also note our assumption that the collected data is sufficiently diverse as—usually implicitly—assumed by most existing data-driven approaches in this domain: It must cover the multitude of driving tasks of the automated or autonomous system under test, e.g. data in different parts of the country for which the system is designed. The data must not be biased towards a small part of the driving task, e.g. only collecting highway drive data on a single straight 10km long segment.

We now want to assess if the identified scenario types can be expected to cover all real driving situations, which constitutes the technical contribution of this paper. Using the real driving data and the catalog of scenario types as input, we model this problem as a Coupon Collector's problem (8), as explained in §III. If the catalog is incomplete, we iterate the process (and possibly need to record more driving data, which is not shown in the Figure). Otherwise, we use the scenario types as a basis for the derivation of system-specific test cases. We motivate that we cannot simply re-use

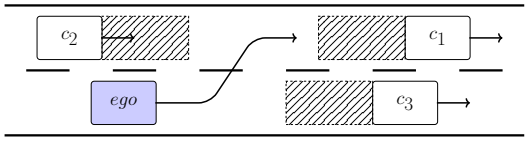



Fig. 2. Example test case for testing lane change functionality

recorded drives as tests (9) in §II-B and sketch how to derive system-specific tests (10) in §II-C.

### B. The Need for System-Specific Tests

The quality of pre-recorded real drives as test cases is system-specific as the following example demonstrates: The ego vehicle *ego* is driving on a two-lane highway behind the car $c_3$ and performs a lane change into the gap between the cars $c_1$ and $c_2$. Suppose our goal is to test whether the system keeps sufficient safety distance (shaded areas in Fig. 2) to the surrounding cars during the lane change.

Assume that in a specific test case for this scenario type, one system version (or configuration) does not keep sufficient safety distance to $c_1$. This means that this test case revealed faulty system behavior. Now assume that a second system version is implemented such that it keeps a larger safety distance to surrounding cars and, because of this, does not even perform a lane change in this test case! The same test case that is able to reveal a faulty behavior for one system is not even a very interesting test case of the correct form (meaning it contains a lane change of the ego vehicle into the gap) for another system. This means that the quality of a test case depends on a specific system's behavior: Recorded tests may be good or bad at revealing failures, thus fundamentally questioning the predictability of the testing procedure.

### C. Test Case Derivation

Therefore, system-specific test cases for each scenario type need to be *generated* for each version of the system. At first sight, we could choose random scenario instances from each scenario type. Analytical as well as empirical considerations [10] show that if the goal of testing is to reveal failures, then test cases need to be chosen on the grounds of defect hypotheses. Otherwise, fully randomly picking tests from the entire input domain cannot be shown to be inferior to partition-based testing in general, which in turn questions the very effort of defining the partition. Empirically, one generally applicable defect hypothesis states that failures are more likely to occur at the boundaries of suitably chosen input blocks, i.e., in "extreme" scenarios for each scenario type. Therefore, we will first use scenario types to partition the input domain (and test requirements), and second "extreme" scenarios in each of the blocks to specifically target failure-provoking behaviors. In this vein, existing works present a multitude of test generation techniques. Very popular are search-based techniques that try to identify the extreme test cases, e.g. [11], [12]. The topic of test case generation, however, relates to the second part of a test ending criterion (Did we test a scenario type sufficiently?), which we do not further discuss in this work. Our focus instead is on the

number of relevant scenario types that we compute on the grounds of real test drive data.

## III. The Coupon Collector's Problem

The Coupon Collector's Problem (CCP) is an instance of the Urn Problem as described in [13]. A famous example of the CCP are the collectable pictures of soccer players during a world championship. There exist $N$ different types of coupons in the urn. Each type $j$ is drawn with a constant probability of $p_j > 0$. It holds that $1 \leq j \leq N$ and $\sum_{j=1}^{N} p_j = 1$. One is interested in the number of samples that have to be drawn independently (with replacement) from the urn such that each type is at least drawn once. We will later turn our attention to the problem of unknown numbers of coupons. Additionally, one would like to know how large the probability is to have a complete collection of all types of coupons when we have drawn $S$ coupons [14], [15]. This means that a solution to the CCP takes the probabilities of all types of coupons $p_j$ as input and yields a number of necessary samples as output. Existing works distinguish two cases for this problem: In the first case, $p_j$ is equal for all types $j$, while in the second one, the probabilities $p_j$ may differ for the distinct types.

In this work, we encounter the case of unequal probabilities of scenario types. Let $X$ be the random variable describing the number of samples that need to be drawn until all types are seen at least once. For the CCP, the samples are assumed to be independent of one another. The following formula can be used for the computation of the estimated value $E(X)$ of expected necessary samples for a complete set [15]:

$$E(X) = \int_0^{+\infty} (1 - \prod_{i=1}^{N} (1 - e^{-p_i x})) \, dx$$

Unfortunately, there is no analytical solution available to compute the probability of having seen all types after drawing $S$ samples. Inspired by [16], we use Monte Carlo simulations to calculate this probability.

The input for the simulations is a set of types $j$ (those will be the scenario types later on) and their probabilities $p_j$. A single simulation of the CCP is achieved by randomly drawing samples from an urn until all types are seen at least once. The idea of the Monte Carlo simulation is to repeat this single simulation many times to yield good estimations for the mean, variance and standard deviation of the random variable $X$. We start with a fixed number of $1000$ simulations to obtain a good estimation of the variance $\sigma$ and mean $\overline{X}$ of $X$. This allows us to compute the number of necessary simulations $sim$ as suggested in [17]: When calculating this number, we use one percent for the standard error $e$ of the sample mean $\overline{X}$. This means that $\frac{\sigma}{\sqrt{sim}}$ needs to be smaller than one percent of the mean value $\overline{X}$ as estimated by the first $1000$ simulations. We use a confidence level $conf$ of $0.95$ (z-score $z_{1-0.05/2} = 1.96$) while computing the number of simulations $sim$ as follows:

$$sim \geq \frac{z_{1-\alpha/2}^2 * \sigma^2}{e^2}$$

We can derive the probability $P(X \leq S)$ to discover all distinct types at least once when drawing at most $S$ samples after executing an additional $sim - 1000$ simulations. For the computation of $P(X \leq S)$ we regard how often it occurs that $X$ is equal to $i$ during the simulations with $1 \leq i \leq S$. By adding up these occurrences $occ(i)$ we can calculate the probability in a similar way as in [18]:

$$P(X \leq S) = \frac{1}{sim} \sum_{i=1}^{S} occ(i)$$

In the same way we can add up the occurrences $occ(i)$ until we achieve a given threshold $\tau$. Therefore, we search for the smallest number of samples $S$, for which the added up occurrences are equal to or greater than $\tau$.

$$S = \text{ minimize } Y \text{ subject to } \frac{1}{sim} \sum_{i=1}^{Y} occ(i) \geq \tau$$

is the number of samples that need to be drawn until all types are seen at least once with probability $\tau$.

## IV. The Test Ending Criterion as CCP

We have seen that one test ending criterion can be reduced to the question whether the list of scenario types is complete. If all scenario types that happen in real traffic are contained in the collected data, the list is complete. This means that we need to see an instance of each scenario type at least once in the data. We are hence interested in the question of whether a scenario type exists in real traffic that is not reflected in the collected data. This can be modeled as a CCP.

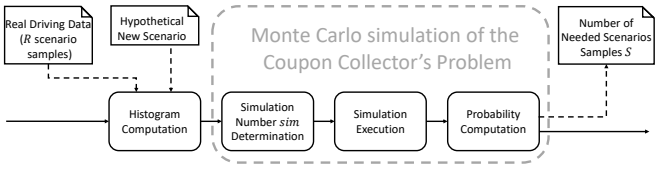



Fig. 3. Process of computing the needed number of samples

We start with a given set of real drive data *that may or may not cover all scenario types*. From this data we derive the number of necessary scenario samples to find all scenario types at least once as shown in Fig. 3. A scenario sample is a single instance of an arbitrary scenario type. Note that scenarios vary in time and in driven distance. The scenarios in the collected driving data are assigned to one of the $N$ distinct scenario types, which is usually easy to do in an ad-hoc way and can also be automated using machine learning [19]. For each type, we count the number of occurrences in the collected data. With these numbers we build a histogram of occurrence probabilities, which serves as an input for the CCP. However, applying the CCP directly to these probabilities would compute the necessary number of scenario samples until all the already known scenario types have been seen at least once when randomly sampling—and this of course is of limited interest if we already know which scenario samples belong to which scenario types. Instead,

we are now interested in deciding if there may be scenario types in the real world that are not covered by the collected data. To this end, we assume that there exists a hypothetical undiscovered scenario with occurrence probability $p_{new}$. This probability is not known a priori, but can iteratively be estimated. The probabilities of the other scenarios $p_j$ are scaled linearly to $p'_j$ such that $p_{new} + \sum_{j=1}^{N} p'_j = 1$ holds. The updated probabilities $p'_j$ together with $p_{new}$ serve as input for the CCP, which then computes the number of necessary scenario samples to see all scenario types at least once, including the newly added hypothetical type. For computation purposes, we use a Monte Carlo simulation as described above. The number of simulations $sim$ within a single Monte Carlo simulation is determined and executed to achieve a result with a given confidence level and error rate as mentioned in §III. Afterwards, we calculate the necessary number of scenario samples $S$ to see all scenario types at least once with a certain probability $\tau$.

$S$ can be used to answer the question whether we did collect all scenarios: Assume that at some point the collected data contains $R$ scenario samples and that $R > S$, meaning that more scenario samples have been collected than the computed number of samples to see a new hypothetical scenario. Further assume that no such new scenario type has been seen during the collection of the $R$ scenario samples. However, with probability $\tau$, we should have seen a new hypothetical scenario type that has an occurrence probability of at least $p_{new}$. Therefore, we conclude that our list of scenarios is complete with regard to the provided confidence values. Data collection can hence be stopped.

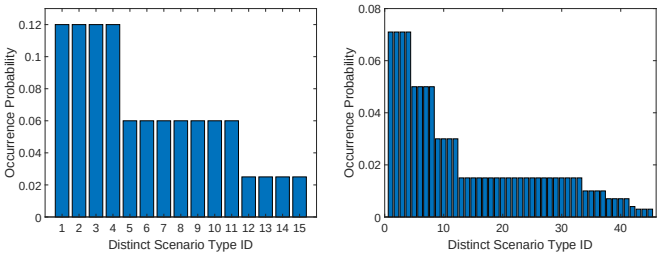
## V. Experiment

Automated and autonomous driving systems have to handle a variety of driving tasks, most prominently piloting through highway or city traffic. Depending on the location, these driving tasks differ a lot. Therefore, certain scenario types may or may not be encountered in some places. For instance, on a German highway, the drivers are obligated to drive on the furthest right lane, when possible. In contrast, this is not the case in the USA. Analogously, speed limits vary from country to country with Germany as the extreme case, where there are no speed limits at all on some parts of the highway. Similarly, distinct scenarios can be found for the city driving task. In many European city centers, cars and bicycles are separated on different lanes, whereas in India, there is mainly mixed traffic. Therefore, the number of and kinds of scenarios are highly dependent on the circumstances under which the data was collected.

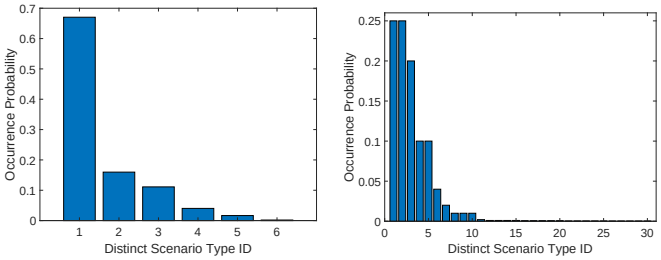
Therefore, in our experiments we are interested in finding out how differently sized and shaped distributions of scenario types affect the computed number of necessary scenario samples $S$. Additionally, we try to gain insight into the influence of $p_{new}$ and $\tau$ on the number of samples $S$, since both variables are required as an input for our approach.

We use four different distributions of scenarios in the experiments and vary the parameters $\tau$ and $p_{new}$. The confidence level $conf$ for calculating the number of simulations $sim$ is set to $0.95$ in all experiments. Also, we fix the standard error $e$ to $1\%$ in the computations of the necessary number of simulations. Both of these values are set according to the opinions of experts in the field.

Car manufacturers and suppliers that are developing automated and autonomous driving systems usually have databases with real drive data as well as histograms of occurrence probabilities for a variety of different locations and driving tasks. They can directly apply the presented approach to their data. The histograms of the four distributions used in the experiments can be found in Fig. 4. The distributions 1, 2, and 4 are fictional. The third one is based on [20], where a distribution is presented that is derived from real drive data.




(1) Highway: $R = 1,000$ (2) Highway: $R = 50,000$




(3) City: $R = 1,000$ (4) City: $R = 50,000$

Fig. 4. Histograms of the four distributions used in the experiments

The first two distributions display the task of driving on a highway. Both are shaped in a way such that the differences in probabilities are relatively moderate. The first distribution is a snapshot after collecting $R = 1,000$ scenario samples of fifteen scenario types with a smallest probability of $0.025$. To show the process of data collection and the discovery of new scenario types, we extended the first distribution and assume additional scenario types at $R = 50,000$ collected scenario samples. Forty-five different scenario types are contained with the smallest probability being $0.003$. We regard the city driving task in the third and the fourth distribution. Analogously to the other two distributions, the third one displays the state after collecting $R = 1,000$ scenario samples, whereas the last distribution is the result of gathering $R = 50,000$ scenario samples. We based the third distribution on the work in [20], in which the authors derive a distribution of six different scenario types from data collected by the U.S. Department of Transportation. There are only six scenario types, since the authors manually create these types and map collected data to them, which prevents the discovery of new scenarios. However, the shape of the distribution with one very common scenario type and a lot of

rare types with a lowest probability of $0.0019$ is in line with the intuition of traffic in cities with a multitude of uncommon situations. We extended this distribution to the fourth one with a few rare and a large amount of very rare scenario types with lowest probabilities of $0.0001$.

Within the experiments, the necessary number of scenario samples $S$ is computed for different combinations of distribution, $p_{new}$, and $\tau$. The results of the experiments can be found in Table I. We executed each experiment 30 times and provide $S$ as well as the standard deviation $\sigma$ of $S$.

TABLE I

CALCULATED NO. OF NEEDED SAMPLES $S$ AND THEIR STANDARD DEVIATION FOR THE FOUR DISTRIBUTIONS; 30 EXPERIMENT RUNS

| distribution | | $p_{new}$ | $\tau = 0.95$ | | $\tau = 0.99$ | |
|---|---|---|---|---|---|---|
| number | $N$ | | $S$ | $\sigma$ | $S$ | $\sigma$ |
| 1 | 15 | 0.001 | 2,991 | 18.72 | 4,608 | 59.39 |
| 2 | 45 | 0.001 | 3,001 | 21.60 | 4,594 | 57.45 |
| 3 | 6 | 0.001 | 3,063 | 34.49 | 4,634 | 67.07 |
| 4 | 30 | 0.001 | 46,007 | 444.68 | 62,250 | 893.71 |
| 1 | 15 | 0.0001 | 29,966 | 165.81 | 45,930 | 451.78 |
| 2 | 45 | 0.0001 | 30,312 | 226.41 | 46,561 | 507.33 |
| 3 | 6 | 0.0001 | 29,988 | 167.46 | 45,881 | 333.53 |
| 4 | 30 | 0.0001 | 47,876 | 485.93 | 63,862 | 1058.07 |
| 1 | 15 | 0.00001 | 332,544 | 2111.74 | 510,755 | 5002.41 |
| 2 | 45 | 0.00001 | 333,595 | 2436.66 | 512,982 | 4761.08 |
| 3 | 6 | 0.00001 | 299,330 | 2462.43 | 460,993 | 4742.39 |
| 4 | 30 | 0.00001 | 299,600 | 2907.31 | 458,658 | 5097.27 |

Our results provide evidence for the following two conclusions. **Firstly**, and probably not too surprisingly in hindsight, if the probability of the new scenario $p_{new}$ is much smaller than the probabilities of the other scenarios $p'_j$, the probability of the new scenario becomes the dominant factor in the calculations of the number of needed samples. The smaller the probability of a scenario, the longer we have to wait to discover this scenario. Therefore, we need more samples to see all scenarios at least once. If there is one scenario with a much smaller probability than the other ones, we need a large amount of samples to see this scenario for the first time and often have seen the others on the way. Thus, the calculated number of needed samples for a complete collection is largely dependent on this small probability. We therefore call it "dominant" in the computations. This dominance can be seen in the results. For the distributions $1$, $2$, and $3$ even the smallest probabilities are higher than $p_{new}$. All of the distributions lead to a similar amount of samples for all parameter settings. On the other hand, the distribution $4$ contains fifteen scenarios that have a low probability between $0.0001$ and $0.0005$. Therefore, the results for this distribution are not dominated by probabilities of $0.001$ and $0.0001$ for $p_{new}$ and thus differ from the results of the other distributions. However, for $p_{new} = 0.00001$ the new scenario becomes dominant in distribution $4$ and the resulting $S$ is similar to the one of the other distributions. **Secondly**, associated with the first findings, we discover that the number $N$ of scenario types that were discovered before the experiments does not seem to impact the results. This can be seen especially for the distributions representing the task of driving on highways. The calculated numbers of necessary samples for the distributions $1$ and $2$ are close to each other with $2,991$ and $3,001$ samples for $p_{new} = 0.001$. But, the first distribution contains only $N = 15$ distinct scenarios, whereas the second one consists of forty-five scenario types. The same can be seen for a lower probability of $p_{new}$. In both distributions the probability of the new scenario is smaller than the probabilities of the already seen scenarios. In these cases, the probability of the new scenario dominates the calculations of the number of needed samples as mentioned earlier. Therefore, the other scenarios and their quantity become less relevant. Since we intuitively would search for a new scenario that has a smaller probability than the others (as otherwise we should have seen it before), the insights from these experiments may generalize to real world data.

These concrete examples visualize how the question can be answered whether we did collect all scenarios: For the distribution 2, we calculate that for $p_{new} = 0.0001$ and $\tau = 0.99$ a number of $S = 46,561$ scenario samples is needed to see each scenario type at least once. Since the distribution $2$ contains already $R = 50,000$ samples, we can state that with a probability of $99\%$ there exists no new scenario that has a probability of $0.0001$ or higher. Otherwise, we would already have seen it. Thus, we know that our list of scenarios is complete for the case that we are not interested in finding a scenario that has a lower probability than $0.0001$. However, if we are interested in a hypothetical undiscovered scenario with $p_{new} = 0.00001$ a lot more scenario samples would be needed ($S = 512,982 > R = 50,000$) and the data collection has to continue.

## VI. RELATED WORK

In [2], a statistical calculation is presented to show that verification and validation solely by real test drives is infeasible. The average driven kilometers between two fatal accidents on German highways are used to derive that $6.61$ billion kilometers need to be driven to encounter at least one of these scenarios. Similarly, also in [3] fatal accidents are used. It is stated that tens of billions of kilometers are needed for direct measurement of sufficient events for statistical analysis. The authors suggest to use virtual scenarios instead. Another work [4] states that millions or even billions of miles have to be driven to arrive at an acceptable level of certainty. Instead, it is suggested to accelerate testing by using mainly critical scenarios. All these works did the important task of raising the awareness that verification and validation by real drive testing alone is not feasible. However, they do not provide a practically applicable test ending criterion.

Other works [19], [21] suggest that the driving system needs to be at least as good as the human driver. The driving behavior of a human driver is analyzed in all scenario types of a specific list of scenario types. An expected system behavior for those scenario types is derived. Then, the system performance is measured with respect to this expected behavior. This might be used as a test ending criterion for those specific scenario types: Stop testing once it can be shown that the driving system is better than a human driver. However, it cannot be used as a general test ending criterion,

since the list of scenario types might not be complete. In this case, it cannot be argued that a system is safe, because it only performed better than a human driver in some scenarios.

There exist works [22], [23] that analyze real drive data and generate for each scenario type a histogram of occurring instances. For example, a cut-in scenario is happening with different relative positions of the vehicles. Those distributions are then used for test case generation by using parameterized scenarios and selecting concrete values for parameters according to the distributions. However, they do not present a test ending criterion.

## VII. Conclusion

We started by describing virtual testing of automated and autonomous driving systems with simulated scenarios. Because of the potentially infinite number of different scenarios the system has to cope with, one can always come up with a new scenario type or a new instance of a type that is different from all others. This immediately raises the need for a test ending criterion, consisting of two parts: Did we test *all scenario types*? Did we sufficiently test *each type with specific instances*? We presented a criterion for the first question as well as a methodology to apply it in practice together with other established scenario-based testing approaches. The test ending criterion is formulated as the question if sufficient real drive data is collected such that *all* scenario types of the real traffic are contained in the data. For the computation if they are indeed *all* scenario types, we model this as a CCP. We show how it can be used as a test ending criterion for testing automated and autonomous driving systems, providing the basis for a safety argumentation for the release of said systems.

It is important not to draw the wrong conclusions from our results. Our approach *effectively indicates* that additional data may be needed if *an unspecified scenario type with a certain occurrence probability* is expected not to have been covered yet. However, it *cannot guarantee* that this additional data will eventually contain a *specific missing scenario type*: Continuing to collect data in the flatlands obviously is unlikely to reveal scenario types prevalent in the mountains.

Further research has to be conducted regarding the probability of the undiscovered hypothetical scenario $p_{new}$ as well as the threshold $\tau$. We assume them to be given and show experiment results for a variety of different values, but it is difficult to choose suitable values a priori. Another assumption in our work is that the samples from the real drive data are independent from one another. This is needed to apply the CCP. For the experiments, the manually created distributions could be far from reality, which is a threat to the validity of the experimental insights. Methodologically problematic is the use of an unsupervised clustering technique, e.g. [9], for drive data clustering. Depending on the applied distance metric between clusters, the clustering technique might provide a different number of clusters, which means a different number of scenario types. We also have mentioned above that depending on the distance measure, the automatically derived clusters need not necessarily correspond to real driving situations that a human would come up with. From a testing perspective, these synthetic clusters are not necessarily less adequate than real driving situations but are likely less easy to interpret by certification authorities.

## References


[1] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 1, pp. 15–24, 2016.

[2] W. Wachenfeld and H. Winner, "The release of autonomous vehicles," in *Autonomous Driving*. Springer, 2016, pp. 425–449.

[3] T. Helmer, L. Wang, K. Kompass, and R. Kates, "Safety performance assessment of assisted and automated driving by virtual experiments: Stochastic microscopic traffic simulation as knowledge synthesis," in *IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, 2015, pp. 2019–2023.

[4] D. Zhao and H. Peng, "From the lab to the street: Solving the challenge of accelerating automated vehicle testing," 2018, online at www.mcity.umich.edu, retrieved 11th March 2019.

[5] S. Ulbrich, F. Schuldt, K. Homeier, M. Steinhoff, T. Menzel, J. Krause, and M. Maurer, "Testing and validating tactical lane change behavior planning for automated driving," in *Automated Driving*. Springer, 2017, pp. 451–471.

[6] J. Zhou and L. del Re, "Reduced complexity safety testing for adas & adf," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5985–5990, 2017.

[7] H. Hungar, F. Köster, and J. Mazzega, "Test specifications for highly automated driving functions: Highway pilot," 2017.

[8] K. Saleh, M. Hossny, and S. Nahavandi, "Kangaroo vehicle collision detection using deep semantic segmentation convolutional neural network," in *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2016, pp. 1–7.

[9] F. Kruber, J. Wurst, and M. Botsch, "An unsupervised random forest clustering technique for automatic traffic scenario categorization," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2811–2818.

[10] A. Pretschner, "Defect-based testing." In: Dependable Software Systems Engineering, 2015.

[11] R. B. Abdessalem, S. Nejati, L. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *Proc. of the 40th Int. Conf. on Software Engineering (ICSE)*, 2018.

[12] J. Deshmukh, M. Horvat, X. Jin, R. Majumdar, and V. S. Prabhu, "Testing cyber-physical systems through bayesian optimization," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, p. 170, 2017.

[13] J. Kobza, S. Jacobson, and D. Vaughan, "A survey of the coupon collector's problem with random sample sizes," *Methodology and Computing in Applied Probability*, vol. 9, no. 4, pp. 573–584, 2007.

[14] A. V. Doumas, "How many trials does it take to collect all different types of a population with probability p?" *Journal of Applied Mathematics and Bioinformatics*, vol. 5, no. 3, p. 1, 2015.

[15] M. Ferrante and M. Saltalamacchia, "The coupon collector's problem," *Materials matemàtics*, pp. 0001–35, 2014.

[16] W. Kurt, "Count bayesie: The toy collector's puzzle," online at https://www.countbayesie.com/blog/2015/10/13/the-toy-collectors-puzzle, retrieved 26th February 2019, 2015.

[17] G. D. Israel, "Determining sample size," 1992.

[18] S. N. Luko, "The "coupon collector's problem" and quality control," *Quality Engineering*, vol. 21, no. 2, pp. 168–181, 2009.

[19] 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 *IEEE 19th Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1360–1365.

[20] D. Zhao, Y. Guo, and Y. J. Jia, "Trafficnet: An open naturalistic driving scenario library," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–8.

[21] C. Roesener, J. Sauerbier, A. Zlocki, F. Fahrenkrog, L. Wang *et al.*, "A comprehensive evaluation approach for highly automated driving," in *25th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration*, 2017.

[22] 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.

[23] E. de Gelder and J.-P. Paardekooper, "Assessment of automated driving systems using real-life scenarios," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 589–594.

# 5. Fitness Functions for Testing Automated and Autonomous Driving Systems

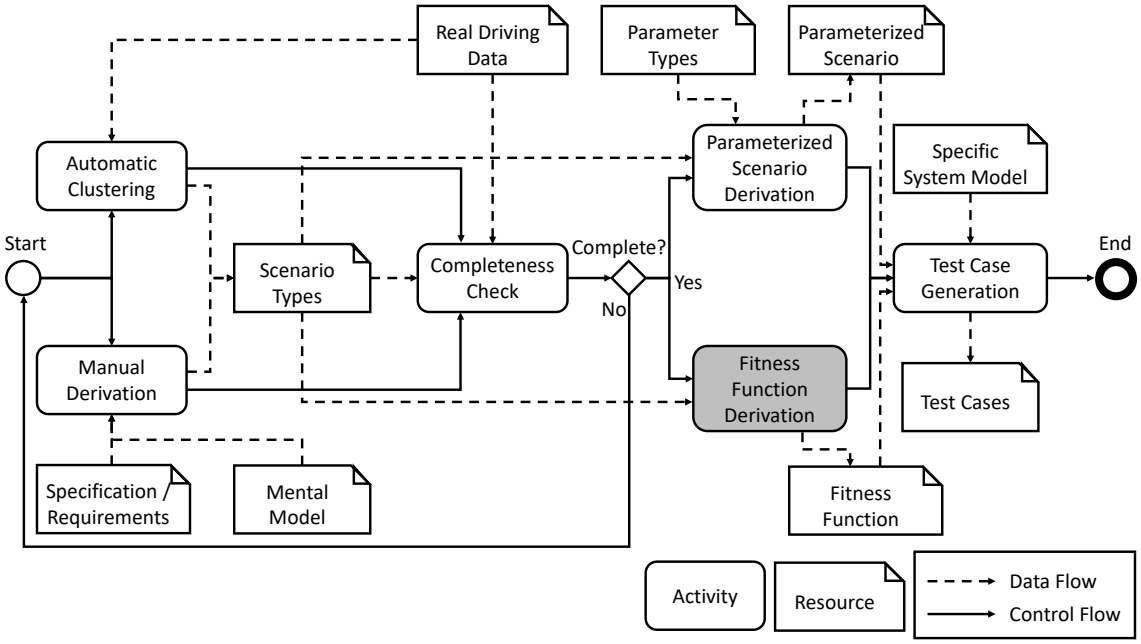



Figure 5.1.: Big Picture (previous versions appeared in [55, 59])

**Summary:** For each in the complete list of scenario types, a parameterized scenarios is created. The parameters span a multi-dimensional space of possible test scenarios. Not every test case that is contained in the space of the parameterized scenario is of the desired form, meaning that it contains a desired maneuver like a lane change, or is interesting in that they cause a correct driving system to approach the safety boundaries or cause a faulty driving system to violate them. For test case generation with search-based techniques, a suitable fitness function is needed that ensures the desired form of the test case and that the test case is interesting. Formulating fitness functions correctly is difficult, time-consuming, and requires experience. For the derivation of fitness functions at large scale, methodological guidance for test engineers is needed.

- Problem: How can practitioners be supported with methodological guidance to derive suitable fitness functions for test case generation with search-based techniques?
- Gap / Contribution: Existing works either provide ad-hoc fitness functions for a specific system or scenario type; or focus on the technical aspects of search-based test scenario generation assuming the fitness function to be given. This leaves the methodological aspect unconsidered of how fitness functions generally should be created.
- Solution: Fitness function templates as well as the means to combine and apply them are presented to ensure that selected test cases are of the desired form and are interesting.
- Evaluation: The templates are used to create fitness functions for a list of 24 relevant scenario types provided by [178]. The searches are executed for each scenario type using the high-fidelity physical simulation CarMaker [31] and a system under test that was built according to a series of papers [117, 118, 119].
- Results: For the 24 scenario types, fitness functions could be formed by applying the proposed methodology and their functionality could be demonstrated. (For those of the 24 for which no clear safety criterion is provided by literature or authority standards, assumptions were made about the meaning of *safety* to instantiate the templates. Any other safety measurement for instantiation may be used as well.)
- Limitations: The templates are evaluated for highway scenarios. Additional or adapted templates might be necessary for non-highway scenario types.

**Author Contribution:** F. Hauer, his supervisor A. Pretschner, and the industrial partner B. Holzmüller conceived the problem statement. F. Hauer derived and implemented the templates as well as the means to combine them for application with search-based techniques. The experiments (including the implementation of an exemplary driving system) and result analysis was conducted by F. Hauer. The manuscript creation was done by F. Hauer in discussion with A. Pretschner.

# Fitness Functions for Testing Automated and Autonomous Driving Systems

**Accepted manuscript for the proceedings of the 38th International Conference on Computer Safety, Reliability and Security 2019**
**The final authenticated publication is available online at**
**https://doi.org/10.1007/978-3-030-26601-1_5**

Florian Hauer[1], Alexander Pretschner[1], and Bernd Holzmüller[2]

[1] Technical University of Munich, Arcisstraße 21, 80333 Munich, Germany
{florian.hauer,alexander.pretschner}@tum.de
[2] ITK Engineering GmbH, Im Speyerer Tal 6, 76761 Ruelzheim, Germany
bernd.holzmueller@itk-engineering.de

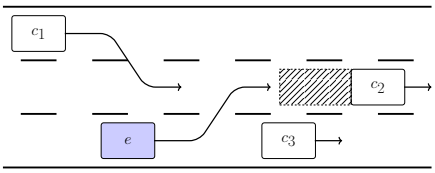
**Abstract.** Functional specifications and real drive data are typically used to derive parameterized scenarios for scenario-based testing of driving systems. The domains of the parameters span a huge space of possible test cases, from which "good" ones have to be selected. Heuristic search, guided by fitness functions, has been proposed as a suitable technique in the past. However, the *methodological challenge of creating suitable fitness functions* has not been addressed yet. We provide templates to formulate fitness functions for testing automated and autonomous driving systems. Those templates ensure correct positioning of scenario objects in space, yield a suitable ordering of maneuvers in time, and enable the search for scenarios in which the system leaves its safe operating envelope. We show how to compose them into fitness functions for heuristic search. Collision and close-to-collision scenarios from real drive data serve as a use case to show the applicability of the presented templates.

**Keywords:** System Verification · Automated & Autonomous Driving · Scenario-Based Testing · Search-Based Techniques

## 1 Introduction

Striving for highly automated and autonomous driving systems results in evermore complex and capable systems. Due to the complexity of these systems and the complexity and sheer number of possible scenarios, ensuring safety and functional correctness is a crucial challenge [9]. Since verification and validation by real test drives alone are practically infeasible [17], the focus shifts to virtual test drives. For virtual testing, scenario-based closed-loop testing in the form of X-in-the-Loop settings is used [16]. Such scenarios describe dynamic traffic situations to test the behavior of the automated or autonomous driving system. A whole set of such scenarios is encoded by a *parameterized scenario*. We show such a parameterized scenario for a highway pilot in Fig. 1. The ego vehicle $e$

accelerates from standstill and approaches car $c_3$, which is driving at lower velocity than $e$. $e$ then changes to the middle lane, while simultaneously $c_1$ changes also to the middle lane behind $e$. During this scenario, $e$ must not violate the safety distances, e.g. the one to $c_2$ (shaded area in Fig. 1). Each other car $c_i$, $i \in \{1, 2, 3\}$ has a parameter for its longitudinal starting position $s_{0,c_i}$, a starting time $t_{start,c_i}$ for accelerating from standstill, and a desired velocity $v_i$ it tries to reach and hold throughout the scenarios. In addition, the lane change of $c_1$ is triggered at a specific time, described by parameter $t_{lc,c_1}$. The domains of these ten parameters span a ten-dimensional space of possible test scenarios.




| Parameter | Lower Bound | Upper Bound |
|---|---|---|
| Starting positions $s_{0,c_i}$ [m] | 0 | 300 |
| Starting times $t_{start,c_i}$ [s] | 0 | 3 |
| Target velocities $v_i$ [km/h] | 80 | 140 |
| Lane change time $t_{lc,c_1}$ [s] | 6 | 13 |

**Fig. 1.** Parameterized highway scenario with ten parameters and their domains

Most scenarios in this space are not useful test cases, however. In some scenarios, $e$ will not even perform a lane change; will perform it in front of $c_2$ instead of behind it; or $c_1$ performs its lane change several seconds later than $e$. Instead, "good" test cases need to be identified within the parameter space. In one interpretation of "good" test scenarios, a correct system *approaches* safe operating limits, and a faulty system violates them. Existing works suggest the use of search-based techniques. These were successfully applied for testing classic advanced driver assistance systems (SAE levels 1&2 [14]), e.g. a parking assistant, an adaptive cruise control, an emergency braking system, and their combination.

Those works focus on technical aspects, e.g. on how to improve the search algorithm, and assume the fitness functions to be given or created ad-hoc. This was an important, and successful, first step. Because these search-based techniques are so promising, we want to apply them to testing automated and autonomous driving systems of SAE levels 4&5 [14]. Such systems are fundamentally different, as they take over the complete driving task including decision making and executing active maneuvers in dynamic traffic scenarios. Thus, the variety of different possible parameterized scenarios is huge, which requires the definition of many different fitness functions. However, formulating fitness functions correctly is difficult, time-consuming, and requires experience. Wrongly derived fitness functions leave "good" test cases unidentified, which might even lead to wrong conclusions about the test results. It seems clear that creating fitness functions ad-hoc, as done in the past, is not sufficient. For the derivation of fitness functions at large scale, methodological guidance for test engineers is needed.

The **contribution of this paper** is the following: We provide such guidance in the form of a set of fitness function *templates* for testing automated and autonomous driving systems in dynamic traffic scenarios with heuristic search. It is further explained how those templates can be easily combined and applied to identify "good" test cases for complex scenario types.

§2 explains scenario-based testing and the application of search-based techniques in this domain. The templates are described in §3, before §4 presents ways to combine them. An application is provided in §5. We discuss related work in §6 and conclude in §7.

## 2 Scenario-Based Testing with Search-Based Techniques

In scenario-based testing of automated and autonomous driving systems, the goal is to test the behavior of such systems in dynamic traffic situations. A multitude of different scenario types exist. Several sources of information are used for the identification of those types, e.g. requirements, safety analysis, functional specifications, traffic rules, and real (test) drives. For each scenario type, one or more parameterized scenarios are derived, each describing a set of test cases. Generalizing and adapting the formalism of [2] and [10], we define a parameterized scenario as $(X, V, D)$, where $X$ is the data set that describes the scenario type (e.g. lane change) and context (e.g. two-lane highway). It can be described using the OpenScenario [1] or CommonRoad [5] formats. The variables $v_i \in V$ $(i \leq n)$ are parameters (e.g. velocities of traffic participants) with their domains $D_i \in D$. Assigning a value to each $v_i$ yields a single test case. The domains in $D$ span an infinite search space $A = D_1 \times D_2 \times \ldots \times D_n \subset \mathbb{R}^n$ of test cases.

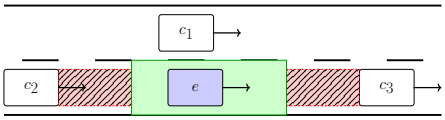



**Fig. 2.** Example of a simple safe operating envelope (green plain rectangle) bounded by the necessary safety distances (red shaded rectangles) and lane markings

The simulated scenario describes input and environment conditions of a **test case**. The expected behavior of continuous systems is described with the help of domains and thresholds. In this context, a safe operating envelope is used (Fig. 2). Inside the envelope, the system is allowed to freely optimize its performance [9], and as long as it does not leave the envelope, it is considered safe. By that the safe operating envelope provides a description of safe system behavior. It depends on the scenario and changes over time during the scenario. Recent works, e.g. the responsibility-sensitive safety (RSS) model [15], the safety force field model [11] as well as other formal models [13] presented such envelopes. These works provide a model of safe system behavior even for scenarios, in which the system alone cannot guarantee complete safety, as other traffic participants may still cause accidents. In the spirit of limit testing, we define a **"good" test case** as follows (see [12]):

> *A "good" test case can reveal* potentially *faulty system behavior. That means in a "good" test scenario, a correct system approaches the limits of the safe operating envelope, and a faulty system violates them.*

A **fitness function** $f : A \to W$ assigns every test case a **quality** value $w \in W$, which depends on the observed behavior of the system under test in the respective test case. It is important that a total order on the fitness values is preserved, such that a scenario gets a better quality value than another if it is a better test case. If the search space $A$ and the quality function $f$ are created accordingly, then search-based techniques may be used to find the "good" test cases in the following way (see Fig. 3):

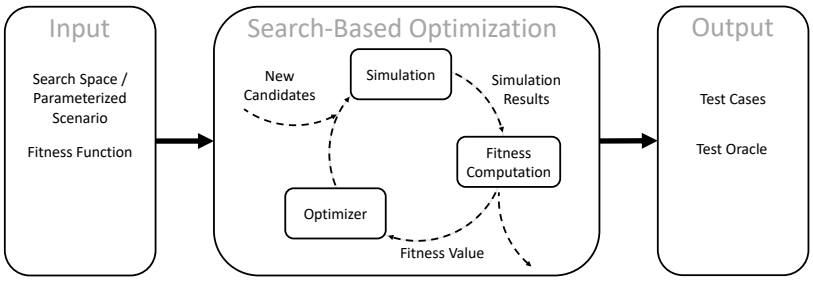



**Fig. 3.** Search-based techniques for scenario optimization

An initial set of scenario candidates is created either by reusing existing scenarios, by using manually created ones by experts, or by generating them randomly. These candidates are then executed in a simulation and the simulation results are evaluated by the fitness function, which returns a quantitative quality measure for the respective scenario. According to these fitness values, the optimization algorithm tries to adapt the parameter values in order to obtain scenarios of better quality. This iteration may be continued until a maximum number of iterations is reached, the assigned computation time is spent or the optimizer fails to find a better solution. This means that during the optimization process, the system is usually tested in one test case per fitness function evaluation, depending on the applied optimization technique. In the ideal case, search-based techniques would find the global optimum, which is the best scenario. This scenario is called worst-case. In the case that the system does not leave the safe operating envelope in the worst-case, it is considered to be safe.

In the following, templates are presented, which may be combined to fitness functions. For this work, the search space is assumed to be given, e.g. we use a parameterized scenario created by a domain expert.

## 3 Fitness Function Templates

In order to capture all potential scenarios, we present templates to aim for qualitative and quantitative test goals. Our goal is to find test cases, in which the system violates the safe operating envelope, e.g. by coming below a distance threshold. We call this a quantitative test goal, since a quantitative value (e.g. a distance between cars) is used to assign a fitness value. We present a suitable template to search for a violation of the safe operating envelope.

However, search spaces usually contain many scenarios in which a desired system behavior, e.g. a lane change, does not take place because the necessary context to provoke it does not occur. For instance, there is no lane change if there is no car to be overtaken. In theory it might be possible to only use a quantitative test goal and search for the violation of a safe operating envelope in a search space covering all possible scenarios. However, in practice this is undesired for several reasons. Scenario types (e.g. lane change, cut-in) are human-interpretable; testing every type on its own provides information about the quality of the system behavior in those specific scenarios. Further, testing these interpretable scenario types will be required by certification authorities. Lastly, such a theoretical search space that contains all possible scenarios is high-dimensional and complex. The search for a safety violation would be difficult - or even practically infeasible - for current search-based techniques. Thus, we need to ensure that the scenario description encodes the relevant parts of the context. Those are called qualitative test goals, since the mere existence of the relevant circumstances is used to assign a fitness value.

For dynamic scenarios, two aspects are of fundamental importance: space and time. Scenario objects need to be at the correct location at a specific moment, e.g. one car should be ahead of another. Furthermore, scenario events need to take place at the right moment in time, e.g. two cars should change the lane simultaneously. Since the (dynamic) behavior of the ego vehicle is unknown a-priori, the correct timing of maneuvers and positioning of scenario objects cannot be established statically and a-priori, e.g. by setting suitable parameter domain boundaries. However, incorporating such desired qualitative test goals into fitness functions is possible. We hence present specific templates for timing and positioning to ensure that such qualitative goals are fulfilled. During optimization those templates identify the scenarios that fulfill the qualitative test goals. Among them the best scenario is searched with the template that aims at the quantitative test goal. Note that in this work, **minimization** is used for optimization purposes. In the following, we will explain the generic idea first, before transferring it to templates for automated and autonomous driving systems.

### 3.1 Template for Testing Against Safe Operating Envelopes

We start with a very basic, simple, and intuitive template. Even though most of the existing works in this domain do not state it explicitly, their idea is to measure a certain system behavior and identify a test case in which this system behavior exceeds a threshold. For the case of a constant threshold, a qualitative generic example is provided in Fig. 4.

The blue time series describes a system behavior and the red line a threshold that must not be violated. This means the maximum value of the blue curve must not be greater than the threshold. During an optimization process, it is desired that better and better scenarios are found, which means that the maximum of the blue curve gets closer and closer to the red line or even surpasses it. The following fitness function idea may be used to achieve the described search behavior (assuming minimization): $f_{\text{idea},1} = -\max(\textit{blue curve})$

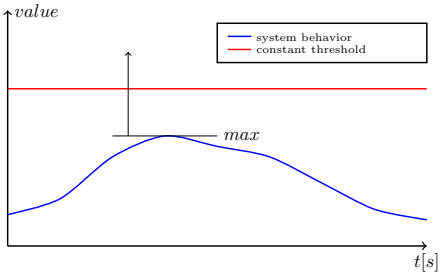



**Fig. 4.** Generic case of testing system behavior against a constant threshold

Now, this idea is transferred for testing automated and autonomous driving systems in dynamic traffic scenarios. As described in Section 2, instead of a constant threshold, a safe operating envelope is used, e.g. as presented in recent works [15,11,13]. Those works express safety often as a safety distance in time or space, which is usually depending on velocities of and relative positions among cars and, thus, is changing over time. We use *safeDist* as a placeholder for the computation of a safety distance according to such a safe operating envelope. We stick to the example of Fig. 1, but for the sake of simplicity, only $c_2$ and the safety distance to it are considered on a single lane for now (see Fig. 5):

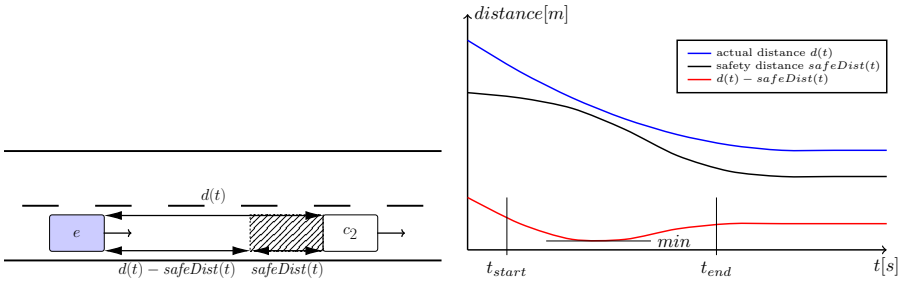



**Fig. 5.** Schematic depiction of a safety distance that should not be violated

The ego vehicle $e$ is approaching another vehicle $c_2$, which is driving at lower velocity. Once $e$ gets closer, it will reduce its velocity until it reaches the velocity of $c_2$. During this period, $e$ must not violate the safety distance. Applying the classic idea $f_{\text{idea},1}$ as fitness function would mean that the scenario is searched, in which the distance $d$ between $e$ and $c_2$ gets smallest. However, a small $d$ does not necessarily mean that the *safeDist* threshold is violated, since safety distances might be even smaller (relatively speaking) in scenarios with low velocities. One cannot conclude by the achieved fitness value whether the safe operating envelope has been violated or not. The dynamically changing safety distance has to be included into the template:

**Template 1:** $$f_{\text{template},1} = \min(d(t) - \mathit{safeDist}(t)) \tag{1}$$

The difference of $d(t)$ and $\mathit{safeDist}(t)$ is denoted as the remaining buffer until violation of the threshold (see image 5). Within the scenario, the minimum

remaining buffer is used as characteristic value, since it is the most dangerous moment. By applying this template, search techniques will identify the scenario in which the minimum of the remaining buffer is smaller than the minimum remaining buffer in all other scenarios. This has the side effect that the following test oracle can be applied for this template: If the remaining buffer is greater or equal than 0 even in the worst-case scenario, the system did never enter and, thus, never violate the safety distance. It even kept an additional distance equal to the remaining buffer in the worst-case. It never left the safe operating envelope and it is considered safe. If the remaining buffer is negative, a faulty system behavior is revealed. In this case, the absolute value is the amount by which the system violated the safety distance. Using this template, an argumentation basis for the release process is provided by making the system behavior measurable. With the help of this measurement, systems can be compared with respect to their performance in the system-specific worst-case.

### 3.2 Templates for Ensuring Qualitative Test Goals

**General Idea to Ensure Qualitative Test Goals.** A specific scenario does or does not satisfy the desired qualitative test goals. The following templates can be used to ensure such goals. By combination of multiple templates (§4), multiple qualitative test goals can be fulfilled. In the case of non-fulfillment of a goal, we assign the value $m$ as fitness value, which has to be greater than any value that corresponds to a qualitative test goal being fulfilled. If $m$ is constant, the optimizer will perform like random selection. However, we want to apply search-based techniques to identify scenarios that satisfy the desired qualitative test goals. Thus, this $m$ should be a gradual measurement to provide a ranking among the scenarios that do not fulfill this qualitative test goal. In order to gradually reach a "fulfilling" scenario, a measurement is used for how far a scenario is away from fulfilling the goal. Since the mere fulfillment of the qualitative test goal is sufficient, every scenario that does so is equally good and, thus, receives the same constant fitness value, e.g. 0:

$$f_{\text{idea,2}} = \begin{cases} m, & \text{qualitative test goal not fulfilled} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Assume a time series (blue curve in Fig. 6), which serves as input for the computation of $m$. At a specific time $t_1$, a qualitative test goal should be fulfilled. Fulfillment means that the value of the time series $value(t_1)$ at $t_1$ is in between the red thresholds $z_{min}$ and $z_{max}$. Note that in general those thresholds do not need to be constant.

If $value(t_1)$ is outside the area described by the thresholds, $m$ is the distance of $value(t_1)$ to the closer threshold to reach the area in between. During the optimization, $value(t_1)$ would approach the area. To avoid having one fitness

function per threshold, the mean of the thresholds is chosen:

$$f_{idea,3} = \begin{cases} \left| \frac{z_{min}+z_{max}}{2} - value(t_1) \right|, & value(t_1) \text{ outside} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

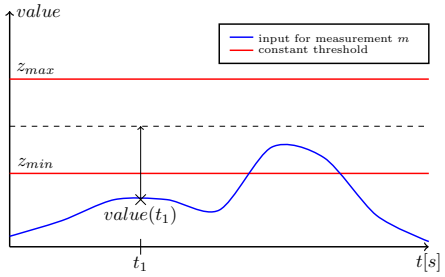



**Fig. 6.** Depiction of the general idea: The value of a curve at a specific moment has to be within a specific domain.

**Template for Correct Positioning of Scenario Objects.** This general idea is now transferred to a template. It ensures that scenario objects, e.g. cars, are correctly located relative to each other at a specific moment in time during the scenario. In Fig. 7, a scenario is depicted in which the ego vehicle $e$ and the other cars $c_1$, $c_2$ are driving on two lanes next to each other. Assume that the qualitative test goal is that $e$ is located in between $c_1$ and $c_2$ at a specific moment $t_{event}$. This might be desired in the case that $e$ should perform a lane change into the gap bounded by $c_1$ and $c_2$.

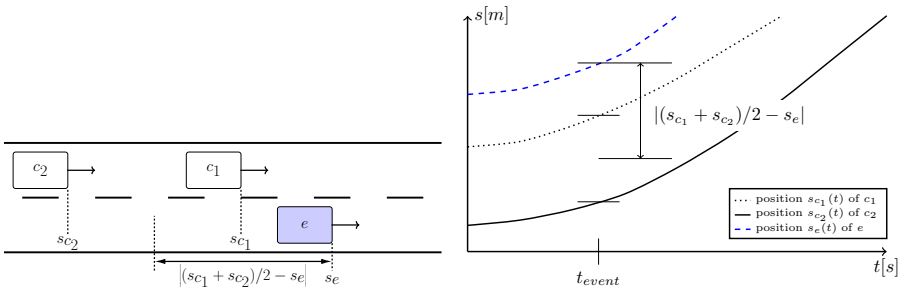



**Fig. 7.** Qualitative test goal: $e$ should be located in the gap at $t_{event}$

The position of the ego vehicle $s_e$ is used to compute the measurement $m$. The positions of the other cars $s_{c_1}$, $s_{c_2}$ serve as thresholds. Note that in contrast to above, here the thresholds are not constant. The transferred template looks

as follows:

**Template 2a:** (4)

$$f_{template,2a} = \begin{cases} \left| \frac{s_{c_1}(t_{event}) + s_{c_2}(t_{event})}{2} - s_e(t_{event}) \right|, & e \text{ not in between } c_1 \text{ and } c_2 \\ 0, & \text{otherwise} \end{cases}$$

During the optimization, the structure of the template will bring $e$ closer and closer to the gap until it is in the gap. However, as it is the case for the introductory example in Fig. 1, there might not be a gap. Only a single other car is of interest for relative positioning. The ego vehicle should be located behind $c_2$ for its lane change. This is reduced to the situation of Fig. 8.

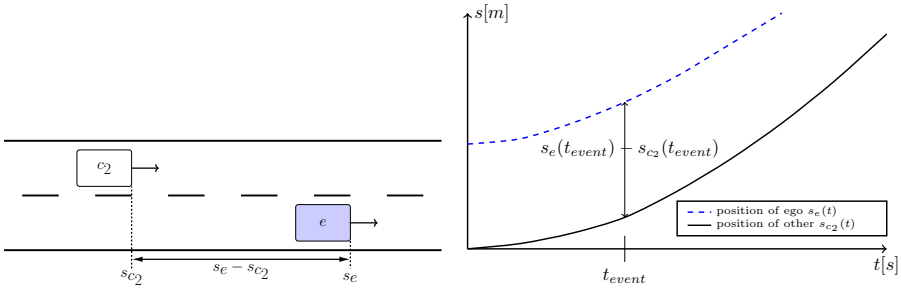



**Fig. 8.** Qualitative test goal: $e$ should be located behind $c_1$ at $t_{event}$

In the case that $e$ is ahead of $c_2$, the distance between them is used as measurement $m$. Since there is only one threshold ("behind of"), there is a slight difference to template 2a. This simplifies the template to the following, where only the distance to the one threshold is used:

**Template 2b:** (5)

$$f_{template,2b} = \begin{cases} s_{c_2}(t_{event}) - s_e(t_{event}), & s_e(t_{event}) < s_{c_2}(t_{event}) \\ 0, & \text{otherwise} \end{cases}$$

**Template for Correct Timing of Scenario Events.** So far, a template for the search of safe operating violations as well as templates for correct positioning of scenario elements were discussed. In the following, a template for timing is presented. It can be used to ensure that events, e.g. the start of a maneuver, are happening at the right moments in time relatively to each other. In the example of Fig. 1, the ego vehicle and the $c_1$ are supposed to perform their lane changes onto the middle lane simultaneously. This means that $c_1$ starts its lane change during the lane change of $e$. This is resembled in Fig. 9.

To allow $c_1$ to start its lane change even a bit before $e$, an offset $\Delta t_1$ can be used. In general, also an offset $\Delta t_2$ is possible, even though here it is set to 0. A $\Delta t_2 > 0$ would mean that $c_1$ starts lane changing after $e$ already completed its lane change. The general idea of $f_{idea,3}$ is adjusted to yield a template for

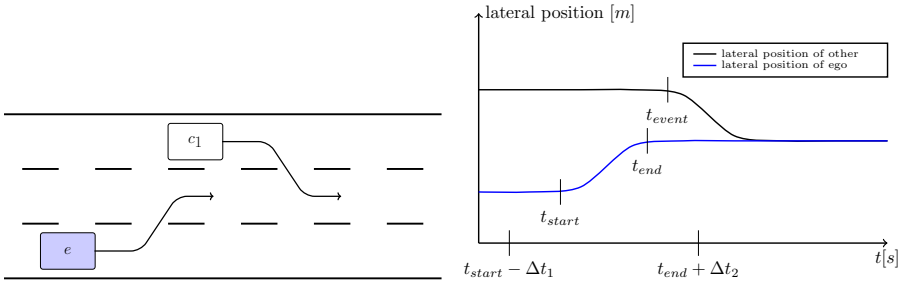



**Fig. 9.** Qualitative test goal: Lane changes should happen simultaneously

timings. However, this time the thresholds are not on the vertical axis as it is the case for the location templates, but on the horizontal one. The thresholds are the start $t_{start}$ and the end $t_{end}$ of the ego vehicle's lane change. In the case that the start of the other vehicle's lane change is not in between $t_{start} - \Delta t_1$ and $t_{end} + \Delta t_2$, the distance to the middle of the interval is chosen. The template for timing looks as follows:

**Template 3:** (6)

$$f_{template,3} = \begin{cases} \left| \frac{t_{start} - \Delta t_1 + t_{end} + \Delta t_2}{2} - t_{event} \right|, & t_{event} \text{ not in between bounds} \\ 0, & \text{otherwise} \end{cases}$$

## 4 Combining Templates

We have presented several templates, each addressing a specific aspect. In the following, it is described how those templates can be combined to a fitness function that can be used by search-based techniques to yield complex scenarios. There are two possibilities: Combining the set of templates to a single fitness function allows the usage of single-objective optimizers, while for multi-objective search, the fitness functions stay separated.

### 4.1 Combination for Single-Objective Search

The templates are nested into each other with the help of case distinctions. The innermost level in the nesting is a template that measures the behavior of the system with respect to a safe operating envelope; it aims for a quantitative test goal. The outer levels of nesting are templates for qualitative test goals (e.g. positioning and timing), which need to be fulfilled for the inner ones. Each level consists of one template returning the measurement $m$ as described above. Instead of returning 0 in the case that the qualitative test goal is fulfilled, the measurement $m$ of the next inner level is returned. This structure causes the optimizer to approach the search in steps. First, scenarios are searched that are of the desired form. Among those, the best scenario is identified for testing against a safe operating envelope. To ensure the necessary total order of fitness values, offsets are added to all levels of nesting except for the most inner one.

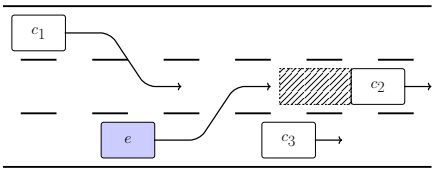
This offset needs to be greater than the maximum value of the next inner level. A simple overapproximation of the sum of the $m$ of the next inner level plus the offset of the next inner level is sufficient.

### 4.2 Combination for Multi-Objective Search

Most likely, there are some goals that are not dependent on each other, meaning that each of those independent goals can be fulfilled without the constraint that the others need to be fulfilled. For instance in the introductory example in Fig. 1, the goal that "$e$ performs its lane change behind $c_2$" can be fulfilled even though the goal that "$c_1$ performs its lane change simultaneously with $e$" is not fulfilled. In contrast to the usage of single-objective search, independent goals can be optimized simultaneously with multi-objective search. Multi-objective search optimizes a vector $x$ of fitness values $x_i$ instead of a single fitness value. The concept of Pareto optimization is used. A vector $x$ is better than another vector $y$ if all $x_i \leq y_i$ and at least one $x_i < y_i$. Each $x_i$ is computed by a single template $f_j$, which may depend on one or more $f_k, j \neq k$. The $f_j$ that are dependent on other $f_k, j \neq k$ need to be adjusted in the following way: In the case that at least one of the $f_k$ is not 0, which means that the qualitative test goal connected to at least one of the $f_k$ is not fulfilled, $x_j$ is set to a very bad, high value. Step by step, the preliminary qualitative test goals will get fulfilled before the remaining test goals are optimized.

## 5 Application of the Templates

Since many car manufacturers and suppliers are currently developing a highway pilot system or a comparable system, such a system is chosen for demonstration purposes. It has to cope with all possible situations on the highway and does not require the driver to take over in critical situations. Therefore, the highway pilot is considered to be an automated driving system of SAE's level 4 [14]. Many natural driving studies have been conducted to gather data for further understanding of road traffic and the driver's task (e.g. [8]). The database of the biggest one [8] got analyzed for near-collision and collision recordings on highways. The findings were grouped to 24 scenario types [18]. We used those as use cases for the presented templates. In fact, the example of Fig. 1 and Fig. 10 is one of those scenario types. The presented templates ensure that maneuvers happen at the right moment and objects are located correctly, while another template searches for violations of the safe operating envelope. Using these templates, we were able to create suitable fitness functions for all of those scenario types. Since those are the near-collision and collision scenarios, they are the most critical ones. By at least covering those 24 scenarios with the templates, we argue that the presented set of templates is sufficient for most of the critical highway traffic scenarios. The following depicts the ease of use of the templates by applying them to the most complex scenario of the 24, which is the introductory example of Fig. 1 and Fig. 10. A variety of other scenarios is contained in this one, e.g. a lane change of the ego vehicle behind another car without further surrounding cars.




| Parameter | Lower Bound | Upper Bound |
|---|---|---|
| Starting positions $s_{0,c_i}$ [m] | 0 | 300 |
| Starting times $t_{start,c_i}$ [s] | 0 | 3 |
| Target velocities $v_i$ [km/h] | 80 | 140 |
| Lane change time $t_{lc,c_1}$ [s] | 6 | 13 |

**Fig. 10.** Most complex (close-to-)collision scenario of the reduced set of scenarios from the database analysis [18]

For this scenario, several fitness functions are needed:

- The lane change of $e$ needs to happen, for which a constant template is used. For the given search space, a constant measurement $m$ is not problematic, since many candidates contain a lane change of $e$.

$$\alpha = \begin{cases} \infty, & e \text{ does not change lanes} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

- The lane change of $e$ needs to happen **behind** $c_2$, which indicates the use of the positioning template. Let the moment, when $e$ gets past the lane markings between the starting lane and the target lane, be denoted as $t_{e,start}$.

$$\beta = \begin{cases} s_e(t_{e,start}) - s_{c_2}(t_{e,start}), & s_{c_2}(t_{e,start}) < s_e(t_{e,start}) \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

- The lane change of $c_1$ needs to happen **behind** $e$. Again, the template for positioning is used. Let the moment, when $c_1$ gets past the lane markings between the starting lane and the target lane, be denoted as $t_{c_1,start}$.

$$\gamma = \begin{cases} s_{c_1}(t_{c_1,start}) - s_e(t_{c_1,start}), & s_e(t_{c_1,start}) < s_{c_1}(t_{c_1,start}) \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

- Lane changes of $e$ and $c_1$ need to be **simultaneously**, for which the timing template is used. Let the moment, when $e$ and $c_1$ are fully on the target lane, be denoted as $t_{e,end}$ and $t_{c_1,end}$. If either $t_{c_1,start} + \Delta t_1 < t_{e,start}$ or $t_{e,end} < t_{c_1,end} - \Delta t_2$ is true, the lane changes are not considered to be simultaneous anymore. $\Delta t_1$ is set to 1s to allow for an earlier start of the lane change of $c_1$, while $\Delta t_2$ is set to 0s such that $c_1$ does not finish the lane change before $e$ starts changing lanes.

$$\delta = \begin{cases} \left| \frac{t_{e,start}-1+t_{e,end}+0}{2} - \frac{t_{c_1,start}+t_{c_1,end}}{2} \right|, & \text{not simultaneous} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

- We need to search for a **violation of the safety distance**.

$$\epsilon = \min(s_{c_2}(t) - s_e(t) - safeDist(t)) \qquad [t_{e,start}, t_{e,end}] \tag{11}$$

The combined fitness function for single-objective search does look as follows. Powers of ten are used as offsets $o_i$, e.g. $o_1 = 10^3$ and $o_2 = 10^4$.

$$f_{single} = \begin{cases} \alpha + o_4, & e \text{ does not change lanes} \\ \begin{cases} \beta + o_3, & s_{c_2}(t_{e,start}) < s_e(t_{e,start}) \\ \begin{cases} \gamma + o_2, & s_e(t_{c_1,start}) < s_{c_1}(t_{c_1,start}) \\ \begin{cases} \delta + o_1, & \text{not simultaneous} \\ \epsilon, & \text{otherwise} \end{cases} \end{cases} \end{cases} \end{cases} \tag{12}$$

For an application of multi-objective search, the templates need to be changed, e.g. $\epsilon$ can only be computed if all qualitative test goals underlying the other templates are fulfilled.

$$\widetilde{\epsilon} = \begin{cases} \infty, & \alpha + \beta + \gamma + \delta > 0 \\ \epsilon \end{cases} \tag{13}$$

Incorporating the dependencies also in the other templates yields the final vector of fitness values. $\beta$, $\gamma$, and $\delta$ are independent of each other; they only depend on $\alpha$. In contrast to a combination for single-objective search as above, they can be optimized simultaneously when combined for multi-objective search. $\alpha$ stays unchanged as it does not depend on other templates.

$$f_{multi} = [\alpha \quad \widetilde{\beta} \quad \widetilde{\gamma} \quad \widetilde{\delta} \quad \widetilde{\epsilon}] \tag{14}$$

The actual technical application of search-based techniques is not the focus of this work as is has been done by various existing works. However, for interested readers, we provide supplementary material online at https://mediatum.ub.tum.de/1474281. Contained are two experiments that use the presented parameterized scenario and combined fitness function as well as videos of the worst-case scenarios identified by single- and multi-objective search during the experiments.

## 6 Related Work

Search-based techniques have been proposed for test scenario selection. The initial research presented the idea of applying search-based techniques for the functional testing of advanced driver assistance systems by testing a parking assistant [6] and a braking assistant [7]. Their setup is close to what we describe as scenario-based testing. Recently, machine learning was introduced to improve the performance of test case generation in this domain. For instance, with learning surrogate models the optimization speed may be improved [3] and with building decision-trees the test engineer receives information about the search space during test case selection [2]. Both works apply the presented techniques on an emergency braking system. For testing a feature interaction of an adaptive cruise control and an emergency braking system, search-based techniques are improved in a way that they search for multiple faulty interactions simultaneously [4].

While all these technical improvements are important and show great results, these works assume the fitness function to be given or create them ad-hoc, e.g. to test the interaction of some specific features [4]. This is, because those works focus on the technical aspect of the search-based techniques. Neither of them addresses the methodological aspect of how fitness functions are correctly created to allow for statements about safety, e.g. by testing against a safe operating envelope as for instance provided by recent works [15,11,13]. Additionally, the evaluation systems are rather reactive driver assistance systems of SAE level 1&2 [14] or combination of such. The provided fitness functions for those systems are mostly not applicable to higher automated systems (e.g. level 4&5) with decision making and active functionality (e.g. lane changing or overtaking) in complex dynamic traffic scenarios, which require the fulfillment of qualitative test goals. This motivates the need for methodological guidance when deriving fitness functions.

## 7 Conclusion

We started by describing the necessity of suitable fitness functions to identify "good" test cases within huge search spaces, described by parameterized scenarios for automated and autonomous driving. A correct derivation of such suitable functions is crucial, but difficult. For the application of search-based techniques at larger scale for testing automated and autonomous driving systems, guidance for test engineers is necessary. In this work, we provide such guidance in form of templates and the means to combine them to fitness functions for complex traffic scenarios. To test against thresholds of a safe operating envelope, we presented a specific template which provides the test engineer with an automated oracle. Additional templates for relative positioning in time and space ensure that the optimizer identifies scenarios that fulfill the qualitative test goals. For combining the templates, we presented both a single and a multi-objective approach which make use of case distinctions to provide a total ordering on scenario candidates such that better scenarios are assigned to better fitness values. As an evaluation, we presented the application of the templates on the most complex (close-to-)collision highway scenario contained in the biggest natural driving study database (identified by [18]). We conclude that the presented templates provide a structured way for test engineers to formulate fitness functions to identify "good" test cases. Thus, this work adds a much needed methodological angle to the otherwise technical solutions.

The application of search-based techniques requires both a fitness function and a search space. The derivation of the search space is not discussed in this work. Similarly to the fitness function derivation, methodological guidance for the derivation of search spaces (parameterized scenarios) is of high interest. Both are difficult the creation of a suitable skeleton of a parameterized scenario and the identification of suitable parameters and their domains. Further, in addition to the methodological guidance presented in this work, an automated fitness function derivation would be very useful to support test engineers. Using

a suitable scenario description as input, the described combination for single- and multi-objective techniques might be automated.

## References


1. OpenScenario 0.9.1 (2017). Tech. rep., OpenScenario Initiative - online at http://www.openscenario.org, retrieved 11th January 2019
2. Abdessalem, R.B., Nejati, S., Briand, L., Stifter, T.: Testing vision-based control systems using learnable evolutionary algorithms. In: Proceedings of the 40th International Conference on Software Engineering (ICSE 2018). ACM (2018)
3. Abdessalem, R.B., Nejati, S., Briand, L.C., Stifter, T.: Testing advanced driver assistance systems using multi-objective search and neural networks. In: 31st IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 63–74 (2016)
4. Abdessalem, R.B., Panichella, A., Nejati, S., Briand, L.C., Stifter, T.: Testing autonomous cars for feature interaction failures using many-objective search. In: 33rd ACM/IEEE International Conference on Automated Software Engineering. pp. 143–154 (2018)
5. Althoff, M., Koschi, M., Manzinger, S.: Commonroad: Composable benchmarks for motion planning on roads. In: Intelligent Vehicles Symposium (IV), 2017 IEEE. pp. 719–726. IEEE (2017)
6. Bühler, O., Wegener, J.: Evolutionary functional testing of an automated parking system. In: Proceedings of the International Conference on Computer, Communication and Control Technologies (CCCT) and the 9th. International Conference on Information Systems Analysis and Synthesis (ISAS) (2003)
7. Bühler, O., Wegener, J.: Evolutionary functional testing. Computers & Operations Research **35**(10), 3144–3160 (2008)
8. Hankey, J.M., Perez, M.A., McClafferty, J.A.: Description of the shrp 2 naturalistic database and the crash, near-crash, and baseline data sets. Tech. rep., Virginia Tech Transportation Institute (2016)
9. Koopman, P., Wagner, M.: Challenges in autonomous vehicle testing and validation. SAE International Journal of Transportation Safety **4**(1), 15–24 (2016)
10. Mullins, G.E., Stankiewicz, P.G., Gupta, S.K.: Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles. In: IEE International Conference on Robotics and Automation (ICRA). pp. 1443–1450 (2017)
11. Nister, D., Lee, H.L., Ng, J., Wang, Y.: The safety force field. online at https://www.nvidia.com/content/dam/en-zz/Solutions/self-driving-cars/safety-force-field/the-safety-force-field.pdf, retrieved 10th May 2019
12. Pretschner, A.: Defect-based testing. In: Dependable Software Systems Engineering (2015)
13. Rizaldi, A., Keinholz, J., Huber, M., Feldle, J., et al.: Formalising and monitoring traffic rules for autonomous vehicles in isabelle/hol. In: International Conference on Integrated Formal Methods. pp. 50–66. Springer (2017)
14. SAE: Definitions for terms related to on-road motor vehicle automated driving systems. J3016, SAE International Standard (2014)
15. Shalev-Shwartz, S., Shammah, S., Shashua, A.: On a formal model of safe and scalable self-driving cars. arXiv:1708.06374 (retrieved 5th May 2019)

16. Ulbrich, S., Schuldt, F., Homeier, K., Steinhoff, M., Menzel, T., Krause, J., Maurer, M.: Testing and validating tactical lane change behavior planning for automated driving. In: Automated Driving, pp. 451–471. Springer (2017)
17. Wachenfeld, W., Winner, H.: The release of autonomous vehicles. In: Autonomous Driving, pp. 425–449. Springer (2016)
18. Zhou, J., del Re, L.: Reduced complexity safety testing for adas & adf. IFAC-PapersOnLine **50**(1), 5985–5990 (2017)

# 6. Re-Using Concrete Test Scenarios Generally Is a Bad Idea

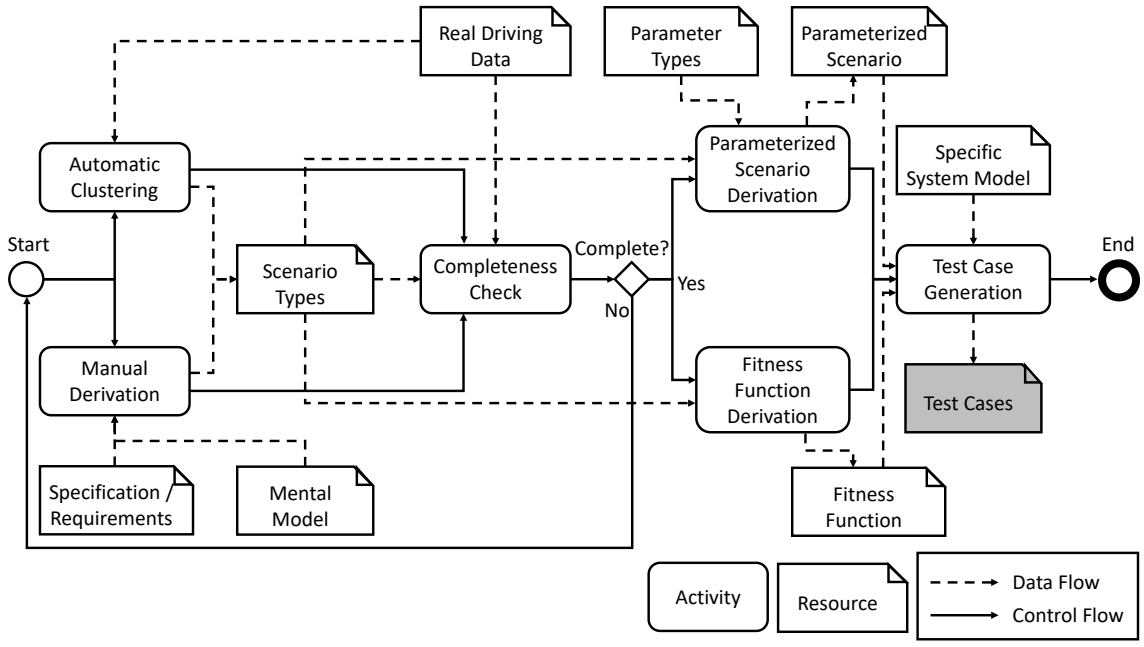



Figure 6.1.: Big Picture (previous versions appeared in [55, 59])

**Summary:** Once "good" test cases have been created, one might have the intention to save time and, thus, might want to re-use them for other versions and variants of the driving system. However, this requires that the quality of the test cases does not drop when re-used, meaning that "good" test cases stay "good". Since different versions and variants of a driving system may perform a maneuver like a lane change at different moments during a scenario and in different ways, it seems that a test case that is "good" for one system may turn bad for another system. However, many approaches in literature and industry are based on the assumption that concrete test scenarios, which are scenario instances and test cases, can generally be re-used [11, 95, 107, 145]. For instance, recording scenario instances in traffic and replaying them as test cases in simulation is an instance of re-using test cases. These perspectives are conflicting and, thus, analysis is required.

- Problem: Are test cases that have been generated or collected for one system generally re-usable for another system?
- Gap / Contribution: Re-usability of scenario instances among different variants and versions of driving systems is implicitly assumed by a variety of works. However, to the best of the author's knowledge there is no work actually providing evidence for or against this assumption.
- Solution: A counter example is experimentally derived: Experiments are conducted in which test cases are generated for different system versions and variants. The re-usability of these test cases for the respectively other systems is analyzed.
- Evaluation: The simulation setup consists of the high-fidelity physical simulation CarMaker [31] and a system under test that built according to a series of papers [117, 118, 119]. For a lane change scenario, test cases are generated for different versions of this system.
- Results: Test cases are system-specific and may turn "bad" when naïvely re-used.
- Limitations: This work solely provides a counterexample to the naïve re-usability. Apart from stating that new test cases should be generated, it is not constructive, e.g. by providing a prediction mechanism that tells whether a test case turns "bad" or not when re-used.

**Author Contribution:** F. Hauer, his supervisor A. Pretschner, and the industrial partner B. Holzmüller conceived the problem statement. F. Hauer implemented the experiment setup, including the implementation of an exemplary driving system. The experiments and result analysis was conducted by F. Hauer. The manuscript creation was done by F. Hauer in close discussion with A. Pretschner.

# Re-Using Concrete Test Scenarios Generally Is a Bad Idea

Accepted manuscript for the proceedings of the IEEE Intelligent Vehicles Symposium 2020
Published version: https://doi.org/10.1109/IV47402.2020.9304678

Florian Hauer, Alexander Pretschner, and Bernd Holzmüller

***Abstract*— Many approaches for testing automated and autonomous driving systems in dynamic traffic scenarios rely on the reuse of test cases, e.g., recording test scenarios during real test drives or creating "test catalogs." Both are widely used in industry and in literature. By counterexample, we show that the quality of test cases is system-dependent and that faulty system behavior may stay unrevealed during testing if test cases are naïvely re-used. We argue that, in general, system-specific "good" test cases need to be generated. Thus, recorded scenarios in general cannot simply be used for testing, and regression testing strategies needs to be rethought for automated and autonomous driving systems. The counterexample involves a system built according to state-of-the-art literature, which is tested in a traffic scenario using a high-fidelity physical simulation tool. Test scenarios are generated using standard techniques from the literature and state-of-the-art methodologies. By comparing the quality of test cases, we argue against a naïve re-use of test cases.**

## I. Introduction

Striving for highly automated and autonomous driving systems results in more and more complex and capable systems. The complexity of these systems as well as the complexity and sheer number of possible scenarios makes safety and functional correctness a crucial challenge [17]. Since testing by real test drives alone becomes practically infeasible [15], [33], the focus shifts to virtual test drives. For virtual testing of vehicle safety, scenario-based closed-loop testing in the form of X-in-the-loop settings is used [30]. Such scenarios usually contain dynamic traffic situations to test the behavior of automated and autonomous driving systems. An exemplary test scenario for testing a highway pilot is depicted in Fig. 1.

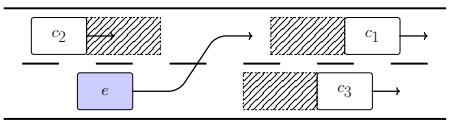



Fig. 1. Example test scenario for testing lane change functionality

The ego vehicle $e$ is driving on a two-lane highway behind the car $c_3$ and performs a lane change into the gap between the cars $c_1$ and $c_2$. It is tested whether the system keeps a sufficient safety distance (shaded areas in Fig. 1) to the surrounding cars during the lane change. In case the system violates the safety distance to e.g. $c_1$, because the gap between $c_1$ and $c_2$ is too small, a faulty behavior of the system is revealed. Now assume that the system is updated by "correcting" this faulty behavior and that the test scenario is re-run to test whether the system behaves correctly now. In this case, it may turn out that the system does not even perform a lane change anymore, since the planning component considers the gap too narrow. Thus, by slightly modifying the system, the previously useful test case now has become essentially useless for testing whether a safety distance is violated during a lane change. Even worse: The system may still violate a minimum safety distance threshold during a lane change. This stays unrevealed unless a new suitable test case is found instead of the re-used one. Since the consequences of a system change for its behavior in dynamic traffic in general cannot be known a-priori, we argue that this problem is of a general nature.

Conventional classical tests can usually be re-used, e.g. for testing the door locking systems. In contrast, the above consideration of driving scenarios has severe implications for the re-use of tests for regression testing as well as for the re-use of recorded test scenarios and tests of so-called "test catalogs." All these approaches rely on the idea of creating test cases once and re-using them later on. They are widely used in industry and literature. Unfortunately, as the simple example conveys, naïvely re-using test scenarios in general cannot directly provide the basis for an argumentation about safety and functional correctness.

The **contribution** of this paper is as follows. We provide a numerical counterexample to the re-usability of concrete test scenarios, heavily relied on by many approaches in industry and literature. Using standard techniques from literature, we generate test scenarios to test an autonomous driving system built according to state-of-the-art approaches. With the experimental results, we can show that naïvely re-using concrete test scenarios cannot guarantee the quality of a test as this test may not even trigger relevant behavior.

The remainder is structured as follows. §II explains scenario-based testing and what constitutes a "good" test case, before the methodology of creating the counterexample is described in §III. §IV explains the experiments and the re-usability issue, and §V provides an overview of related work. We conclude in §VI.

## II. "Good" Test Cases in Scenario-Based Testing

In scenario-based testing the goal is to mimic real traffic scenarios in simulation to test the safety of the driving

F. Hauer and A. Pretschner are with Department of Informatics, Technical University of Munich, Germany, e-mail: {florian.hauer,alexander.pretschner}@tum.de
B. Holzmüller is with ITK Engineering, Germany, e-mail: bernd.holzmueller@itk-engineering.de

behavior automated and autonomous driving systems. A variety of different scenario types (lane change, emergency brake, etc.) take place in real traffic. Lists of such scenario types are derived from experience [31] and real data [11], and the completeness of such lists is determined with statistical models [13]. For each scenario type, a parameterized scenario is created, called *logical scenario* [21]. The intention is to capture the variability of the real world with $n$ parameters $P$ and their domains $D_j \in D, j = 1..n$, e.g. the initial velocity of other traffic participants in a scenario is not set to a specific value of $100km/h$, but is assigned a parameter $v_{other}$ with domain $[80, 130]$. The domains span a space $A = D_1 \times D_2 \times ... \times D_n \subset \mathbb{R}^n$ of test cases. Assigning to each parameter a value from its domain yields a single, executable test case, which is called *concrete test scenario* [21]. Not every candidate in such a space $A$ is of the correct form (e.g. the ego vehicle should perform a lane change, but does not) and among those that have the correct form not all are interesting (e.g. all other vehicles are several hundred meters away from the ego vehicle) [12]. In other words, many test scenarios in such a search space are not the "good" test scenarios we are searching for. The concrete scenario describes the input and environment conditions of a test case. The expected behavior of continuous systems is described with the help a safe operating envelope (cf. Fig. 2). Inside the envelope, the system is allowed to freely optimize its performance [17], and as long as it does not leave the envelope it is considered safe. A variety of works present such safety envelopes [24], [26], [28].

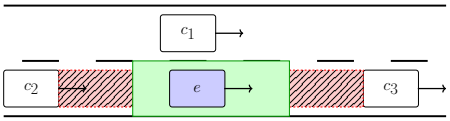



Fig. 2. Example of a safe operating envelope (green plain rectangle) bounded by the necessary safety distances (red shaded rectangle) and lane markings

In the spirit of limit testing, we define **"good" test cases** to test vehicle safety as follows [12], [25]: *A "good" test case can reveal* potential *faulty system behavior. In a "good" test scenario, a correct system approaches the limits of the safe operating envelope, and a faulty system violates them.*

## III. Creation of the Counterexample

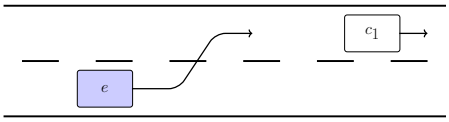
To argue against naïve re-use, we generate test cases for different variants of a system and then simulate "re-use" of these tests by applying each test to all variants. If concrete test scenarios were re-usable, the quality of a test case should not drop when re-used for another system. This requires that "good" test cases are generated and that the quality of the test cases can be measured.

Existing works suggest the use of search-based techniques for the selection of "good" test scenarios (detailed information in Sec V). Such techniques try to identify the best candidate (here: concrete test scenario) in a search space (here: logical scenario) with the help of a fitness function, which provides a quality measure for a test case of how good it is. In the following, the logical scenario, the fitness function, and the systems that are used in the experiments are explained.

### A. Logical Scenario

The logical scenario used for the experiments is shown in Fig. 3. It is a simplification of the exemplary scenario in Fig. 1. The cars $c_2$ and $c_3$ have been removed. When re-using concrete test scenarios for such a simple lane change like in Fig 3, we can expect that at least the form of the concrete scenario is still correct, i.e. the ego vehicle performs a lane change behind $c_1$. For more complex scenario types this is not the case as illustrated for the example in Fig. 1.

The ego car will start at the longitudinal position 0m and accelerate to the initial velocity $v_e$. As soon as starting time $t_{start,c_1}$ is reached, the other car starts accelerating from a longitudinal starting position $s_{0,c_1}$ to an initial velocity $v_{c_1}$. As soon as both cars reach their initial velocities, the lane change request is triggered after time $t_{trg}$. Those parameters provide the necessary search space to allow $c_1$ with $v_{c_1}$ to be located arbitrarily on the road at the point in time when $e$ is triggered to perform a lane change.




| Parameter | Lower Bound | Upper Bound |
|---|---|---|
| Ego vehicle $e$: | | |
| Initially reached velocity $v_e$ [m/s] | 22.22 (80km/h) | 36.11 (130kmh) |
| lane change trigger time $t_{trg}$ [s] | 0 | 5 |
| Other car $c_1$: | | |
| Longitudinal starting position $s_{0,c_1}$ [m] | 0 | 500 |
| Starting time $t_{start,c_1}$ [s] | 0 | 5 |
| Initially reached velocity $v_{c_1}$ [m/s] | 22.22 (80km/h) | 36.11 (130km/h) |

Fig. 3. Parameterized scenario used for the experiments

### B. Fitness Function

The fitness function assigns a quality to each concrete test scenario, which allows the search-based technique to select the "good" test cases. The goal is to test whether the system keeps sufficient distance during a lane change behind another car. Thus, the fitness function has to guide the search-based technique to identify test cases of the correct form, i.e. ego vehicle performs a lane change behind the other car. Among those with the correct form, the biggest violation of the safety distance is searched. We created the fitness function according to the literature [12] (the optimizer minimizes):
The fitness function with $t \in [t_{start}, t_{end}]$:

$$f = \begin{cases} \infty; & \text{no lane change happens} \\ \begin{cases} s_e(t_{start}) - s_{c_1}(t_{start}); & s_{c_1}(t_{start}) \leq s_e(t_{start}) \\ \min\{d - safeDist(t)\}; & s_{c_1}(t_{start}) > s_e(t_{start}) \end{cases} \end{cases}$$

The first part of the fitness function assigns $\infty$ as a very bad, high value to scenarios in which the ego vehicle does not perform a lane change. Instead, if such a lane change takes place, the longitudinal position of the ego vehicle at the start of the lane change $s_e(t_{start})$ is compared with the one of the other vehicle $s_{c_1}(t_{start})$. If the ego vehicle is in

front of the other car, the distance between the cars at this moment $d(t_{start}) = s_e(t_{start}) - s_{c_1}(t_{start})$ is assigned as bad fitness value. The smaller this distance is, the better the fitness value. In the third case, the lane change takes place behind the other car. The difference of distance $d(t)$ and the the safety distance $safeDist(t)$ is the remaining distance until the safety distance is violated. By computing the minimum of it throughout a lane change ($t \in [t_{start}, t_{end}]$), the least safe remaining distance until violation during this lane change is determined. During the search through the search space this yields the least safe remaining distance in the whole search space. If the system behavior is faulty, it will yield the biggest violation in the search space instead. The safety distance is computed according to a formalization [26] of the Vienna Road Convention. However, other safety models may be used as well, e.g. [24], [28].

### C. Driving Systems

The driving systems for the experiments are different versions and variants of a single system type for two reasons: First, we want to show that even small changes in the configuration already cause issues with the re-usability of concrete test scenarios. Changing the whole system design has a far bigger impact than small configuration changes. Second, testing different versions of a single system better represents regression testing, which is the common use case for re-using test cases.

The architecture of our demonstration system is shown in Fig. 4. It follows the general control paradigm, which contains the notions of sensing, long-term planning (or decision-making), short-term planning, tracking and actuation [4]. To ease the interpretation of the experimental results, we excluded sensing from the experiment setup. Also for the sake of simplicity, the car $c_3$ from the introductory example scenario has been removed as a trigger to the decision making to change lanes. Thus, also the decision making is removed. To preserve the variability of decision making caused by different relative positions of the ego vehicle and this $c_3$, the desire to change is triggered by a time trigger (cf. $t_{trg}$ in Sec. III-A). For the short-term planning, a state-of-the-art approach for lane change maneuvers of automated vehicles [23] was used. This system was chosen to closely resemble automated and autonomous driving systems of SAE levels 4 & 5 [27], e.g. a Highway Pilot of SAE level 4.

This model predictive control approach first predicts the positions $x_i$ and velocities $v_i$ of surrounding cars $c_i$ for each

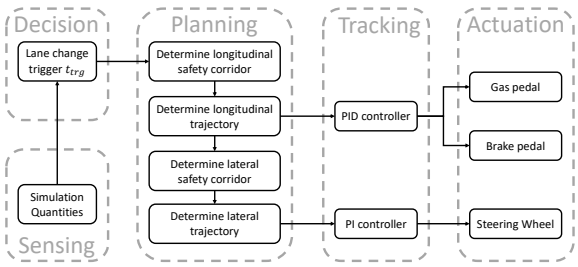



Fig. 4. System architecture overview

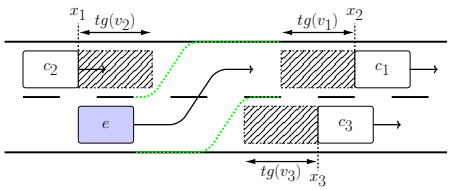



Fig. 5. Schematic simplified depiction of the predicted positions $x_i$ and time gaps $tg(v_i)$ at a specific prediction time $t_k$ of the system

sample time step $t_k$ over a short time horizon (cf. Fig. 5). For safety distance planning, a time gap $\tau = 0.5s$ is used as proposed in the paper series of [23]. The safety corridor for the lane change (dashed green lines) is bounded by the distance $tg(v_i) = \tau \cdot v_i$ to the predicted positions $x_i$ of other cars $c_i$. First, a safe longitudinal and afterwards a safe lateral trajectory is computed within these bounds. The objective is to keep the velocity at each time step of the trajectory as close to the velocity before the lane change (given by the scenario parameter "initially reached velocity" $v_e$) and the acceleration as low as possible. The tracking of the trajectories is done by classic control of the gas and brake pedals as well as the steering wheel. Both are tuned for a physical model of a sports car provided by the widely used physical simulation software CarMaker by IPG Automotive, which served as the simulation environment for this work.

## IV. Experiments

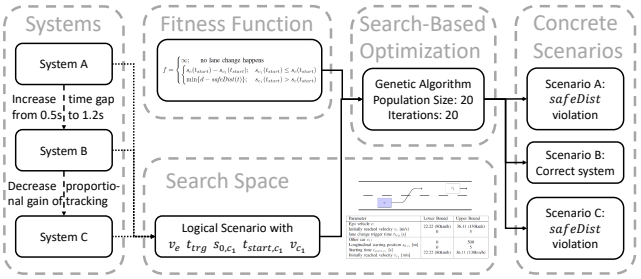


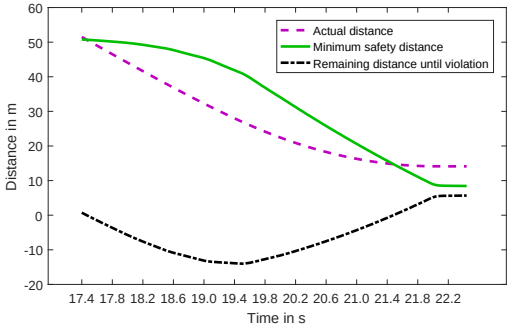

Fig. 6. Experiment Overview

In Fig. 6, an overview over the experiments is given. Three systems configurations are created (systems A,B,C). System A is the system described above, while system B is created by increasing the time gap $\tau$ from 0.5s to 1.2s, and system C is yielded by decreasing the proportional gain of the longitudinal PID controller in system B. The fitness function from Sec. III-B and the logical scenario as the search space from Sec. III-A are used for all three experiments. For the optimization, a single-objective genetic algorithm is applied. Note that the technological aspect is not the focus of this work and more advanced techniques could be used, as described in [9]. The population size and the number of generations were both set to 20, resulting in 400 simulation executions, which means each system is tested in 400 concrete scenarios during the optimization. The experiments were executed multiple times to rule out randomization effects. The scenario with the best fitness value of each experiment is presented in detail in the images of respective Fig. 7, 8, 9: (1) distances during lane change,

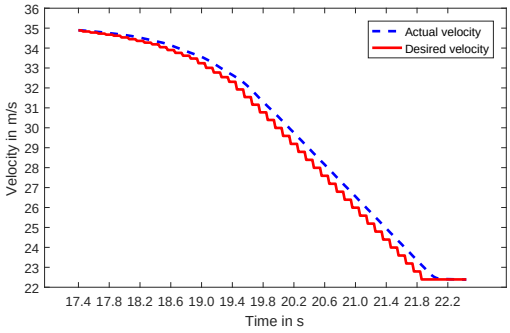
and (2) velocities during lane change. Experiment A shows that faulty behavior of the trajectory planner can be revealed, B presents a case where the system behavior is seemingly correct, and in C faulty behavior caused by slow tracking is detected.

### A. Experiment Results

In **scenario A**, which is the best concrete test scenario that could be found for system A, the ego vehicle performs a lane change behind the other car, which drives at lower velocity. In Fig. 7(1), the actual distance between both cars $d(t)$, the minimum safety distance $safeDist(t)$, as well as the remaining distance until this safety distance is violated are shown (a negative remainder means violation). Approximately between scenario time 17.5s and 21.5s, the safety distance is violated (see 7(1)). The biggest violation of 14m takes place at approximately 19.5s. Even though tracking will never be perfect, here it is considerably fast (cf. Fig. 7(2)), which leads to the conclusion that the planned trajectory does not consider sufficient safety distance in this scenario with respect to the safe distance computation of the fitness function. This faulty behavior gets addressed by increasing $\tau$ in the safety distance estimation $tg(v_i)$ of the planning algorithm to 1.2s to yield system B. The concrete value of 1.2s was determined experimentally. The results of the **scenario B** are shown in Fig. 8. It can be seen that even in the best concrete test scenario, the updated time gap causes the system to keep sufficient distance to the other vehicle to not violate the minimum safety distance. Therefore, it is considered to be safe with respect to this search space. To yield system C, the proportional gain of the velocity

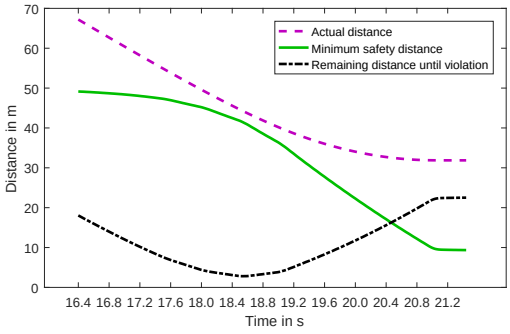



(1) Distances during the lane change of test scenario A



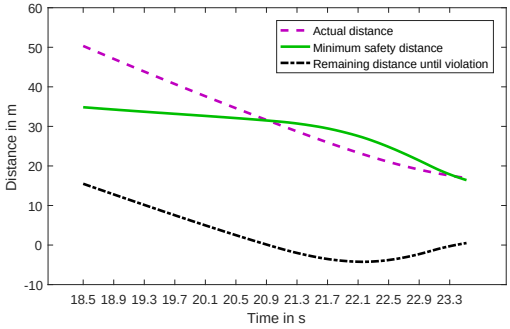

(2) Velocities during the lane change of test scenario A

Fig. 7. Results of scenario A: Faulty trajectory planner

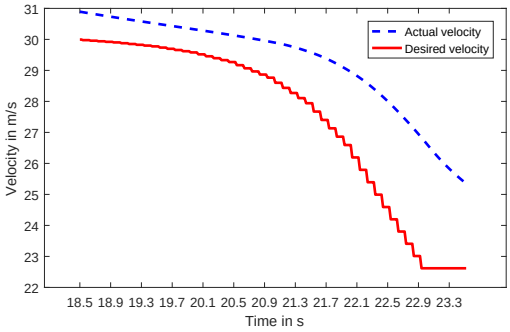



(1) Distances during the lane change oftest scenario B

Fig. 8. Results of scenario B: Seemingly safe system




(1) Distances during the lane change of test scenario C




(2) Velocities during the lane change of test scenario C

Fig. 9. Results of scenario C: Tracking too slow

tracking PID controller of system B is decreased to make the tracking more smooth, e.g. for passenger comfort. However, this also increases the rise time of the controller. The results of **scenario C** (cf. Fig. 9(2)) show that the tracking does not follow the planned trajectory close enough to decrease the ego vehicle's speed as fast as necessary, which causes a violation of the safety distance despite the increased time gap $\tau$.

### B. Why Re-Using Concrete Scenarios Is a Bad Idea

Each system (e.g. system A) is additionally tested using the best concrete test scenarios identified for the other systems (e.g. scenario B/ C). The resulting fitness values, which measure both the test case quality and the remaining buffer until violation of the safety distance in meters, are depicted in Tab. I.

| | System A | System B | System C |
|---|---|---|---|
| Scenario A | -14.001 | 5.604 | -3.065 |
| Scenario B | 2.595 | 2.812 | -3.037 |
| Scenario C | 20.153 | 20.484 | -4.238 |

TABLE I

FITNESS VALUES OF THE THREE IDENTIFIED BEST CONCRETE TEST SCENARIOS FOR THE THREE SYSTEMS

Every concrete test scenario has the best fitness value for testing the system it was created for (red cells). This confirms the intuition regarding the re-usability issue implied with the introductory example in Sec. I. The smallest remaining buffer until violation (positive fitness value) or biggest violation (negative fitness value) for distinct systems is found in different scenarios. Especially the first column of Tab. I illustrates the following. If the best concrete test scenarios B and C are "re-used" to test system A, the faulty behavior of system A is not revealed. In other words: By naïvely reusing test scenarios that have been created, or recorded, for other system versions or variants, the faulty behavior of a system might not be revealed. In this example, even the best concrete test scenario for another system is not sufficient, let alone scenarios of lower quality. Note that for this observation, the provenance of these concrete scenarios does not matter, e.g. they could have originated from test drive recordings or "test catalogs" of concrete test scenarios, showing the same result in terms of re-usability. This emphasizes the need for the identification of system-specific concrete test scenarios, since re-used test cases may not provide a strong foundation for a safety argument for the release of an automated or autonomous driving system.

Our argument does *not* imply that *no test case can ever be re-used*. Our experiments show, instead, that *there exist new system variants* for which the quality of existing test cases is *provably bad*. Our conclusion is hence that *in general*, the quality of test cases considered for re-use is *unknown*. If there are external means to ensure that the quality of existing test cases continues to be good, for instance because it is known that the driving behavior of the system under test has not been modified, then these tests may indeed be re-used. It is then the responsibility of the test engineer to argue that the quality of tests for an earlier version of a system carries over to a newer version.

## V. RELATED WORK

Several existing works are based on the idea of extracting concrete test scenarios from data according to a variety of different selection and filtering criteria. This data is collected with real test drives [5], [19], [22] or simulation setups [29]. Similar to manual test case generation based on experience, the result of those approaches are "test catalogs" of concrete test scenarios. These are subject to the presented re-usability issue. Additionally, the extracted scenarios are randomly encountered scenarios, which is problematic for safety argumentations, as such are hardly possible based on randomly encountered scenarios. An infeasible amount of driving hours or driven kilometer is necessary for each version of the system [33], [15] for each version and variant of the system.

A very popular idea to circumvent this issue is to extract all concrete test scenarios that occur in real data and filter for the "critical" ones, where a multitude of distinct metrics of criticality exist [14], [20], [32]. The resulting "test catalogs" are relatively small as they contain only the critical scenarios. When the amount of data is huge, there is the hope that the resulting "test catalog" overcomes the randomness of encountering certain concrete scenarios in real traffic. However, it is still subject to the re-usability issue. The critical concrete test scenarios are critical with respect to the behavior of the recording test vehicle or the driver maneuvering the recording vehicle. When such concrete test scenarios are re-used for testing another vehicle, e.g. the next generation automated or autonomous vehicle, it is not guaranteed that the concrete test scenarios are still critical. It might be an easy non-critical test scenario for the new system. Note that such criticality metrics might be used for targeted generation of concrete test scenario, e.g. as fitness function for the application of search-based techniques [10]. In this case, the usage of such criticality metrics is not an instance of the re-usability issue.

A variety of existing works suggest the use of search-based techniques for the selection of concrete test scenarios. We followed this idea to generate system-specific concrete test scenarios. Some of those works focus on the technical aspects of the scenario search [1], [2], [3], [6], some describe fitness functions for different specific purposes [7], [8], [16], and another one provides fitness function templates for testing against safe operating envelopes [12]. Even though these works implicitly advocate the generation of system-specific concrete test scenarios, neither of them discusses the re-usability issue or provides a numerical counterexample to the re-usability.

## VI. CONCLUSION

We started by sketching that re-using concrete test scenarios for the testing of automated and autonomous driving systems in general is problematic, since the quality of a test case is system-dependent and may become bad when the system changes. However, many approaches in industry and literature rely on this re-usability. We provided a numerical counterexample to the re-usability of concrete test scenarios. We used standard techniques to generate concrete test scenarios for three different configurations of an exemplary system. As scenario type, we chose an easy lane change. The experimental results showed that even the best concrete test scenario for one system configuration may not be a "good" concrete test scenario for the other. Note that even for a simple lane change and configurations of the identical system design, the re-usability issue could be shown. Systems with different designs (e.g. systems from different vendors) differ much more than configurations of a single design. Thus, re-usability may even be worse. While for this lane change scenario the re-used concrete test scenarios at least are of the correct form, i.e. the ego vehicle performs a lane change

behind the other car, for more complex scenario types, re-used scenarios might not even be of the correct form anymore. In the case of the introductory example of Fig. 1, almost certainly the lane changes will not be performed into the gap anymore as different systems decide to do their lane changes at different moments in time. In terms of re-usability, the provenance of the concrete test scenarios does not matter, e.g. they could be recorded test cases or reference tests from "test catalogs."

We have argued that this shows that the quality of existing tests *in general* cannot be predicted for new versions of a system. While this shows that there cannot be "canonical" reference tests for a product line *in general*, this does *not* mean that the quality of existing test cases *never* carries over to new versions of the system. There may well be situations where an engineer can argue that the re-use of tests is justified.

As a direct consequence, regression testing of automated and autonomous driving systems needs to be reviewed, and new methodologies are necessary. As a solution, we recommend the re-use of logical scenarios instead of concrete test scenarios. Then, for a new system, system-specific concrete test scenarios are generated for each logical scenario instead of re-using concrete test scenarios—which is the automated technique we used to generate all test cases in this paper. The generation of tests obviously is more costly than simple re-use, but our arguments suggest that this cost cannot be avoided unless one can argue that tests for earlier versions are safe to re-use. In addition to using recorded scenarios directly as test cases when this can be explicitly justified, we deem these recorded scenarios useful for the derivation of logical scenarios for test case generation, e.g. as done in [18], [34].

## REFERENCES


[1] R. B. Abdessalem, S. Nejati, L. Briand, and T. Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *International Conference on Software Engineering*. ACM, 2018.

[2] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter. Testing advanced driver assistance systems using multi-objective search and neural networks. In *IEEE/ACM International Conference on Automated Software Engineering*, pages 63–74, 2016.

[3] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter. Testing autonomous cars for feature interaction failures using many-objective search. In *ACM/IEEE International Conference on Automated Software Engineering*, pages 143–154, 2018.

[4] M. Aeberhard et al. Experience, results and lessons learned from automated driving on germany's highways. *IEEE Intelligent Transportation Systems Magazine*, 7(1):42–57, 2015.

[5] J. Bach, M. Holzäpfel, S. Otten, and E. Sax. Reactive-replay approach for verification and validation of closed-loop control systems in early development. Technical report, SAE Technical Paper, 2017.

[6] H. Beglerovic, M. Stolz, and M. Horn. Testing of autonomous vehicles using surrogate models and stochastic optimization. In *IEEE 2017 Intelligent Transportation Systems Conference*, pages 1–6, 2017.

[7] A. Calo, P. Arcaini, S. Ali, F. Hauer, and I. Fuyuki. Generating avoidable collision scenarios for testing autonomous driving systems. In *IEEE Int. Conf. on Software Testing, Verification and Validation*, 2020. to appear.

[8] A. Calo, P. Arcaini, S. Ali, F. Hauer, and I. Fuyuki. Simultaneously searching and solving multiple avoidable collisions for testing autonomous driving systems. In *Genetic and Evolutionary Computation Conference*, 2020. to appear.

[9] R. Feldt and S. Poulding. Broadening the search in search-based software testing: It need not be evolutionary. In *IEEE/ACM International Workshop on Search-Based Software Testing*, pages 1–7, 2015.

[10] 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*, 2019.

[11] F. Hauer, I. Gerostathopoulos, T. Schmidt, and A. Pretschner. Clustering traffic scenarios using mental models as little as possible. In *IEEE Intelligent Vehicles Symposium (IV)*, page to appear, 2020.

[12] F. Hauer, A. Pretschner, and B. Holzmüller. Fitness functions for testing automated and autonomous driving systems. In *International Conference on Computer Safety, Reliability, and Security*, pages 69–84, 2019.

[13] F. Hauer, T. Schmidt, B. Holzmüller, and A. Pretschner. Did we test all scenarios for automated and autonomous driving systems? In *IEEE Intelligent Transportation Systems Conference*, pages 2950–2955, 2019.

[14] P. Junietz, F. Bonakdar, B. Klamann, and H. Winner. Criticality metric for the safety validation of automated driving using model predictive trajectory optimization. In *IEEE Intelligent Transportation Systems Conference*, pages 60–65, 2018.

[15] 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*, 94:182–193, 2016.

[16] M. Klischat and M. Althoff. Generating critical test scenarios for automated vehicles with evolutionary algorithms. In *IEEE Intelligent Vehicles Symposium*, 2019.

[17] P. Koopman and M. Wagner. Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety*, 4(1):15–24, 2016.

[18] F. Kruber, J. Wurst, and M. Botsch. An unsupervised random forest clustering technique for automatic traffic scenario categorization. In *IEEE Intelligent Transportation Systems Conference*, pages 2811–2818, 2018.

[19] U. Lages, M. Spencer, and R. Katz. Automatic scenario generation based on laserscanner reference data and advanced offline processing. In *IEEE Intelligent Vehicles Symposium Workshops*, pages 146–148, 2013.

[20] M. Lehmann, M. Bäumler, G. Prokop, and D. Hamelow. Use of a criticality metric for assessment of critical traffic situations as part of sepia. In *19. Internationales Stuttgarter Symposium*, pages 1154–1167. Springer, 2019.

[21] T. Menzel, G. Bagschik, and M. Maurer. Scenarios for development, test and validation of automated vehicles. *arXiv:1801.08598*, 2018.

[22] P. Minnerup, T. Kessler, and A. Knoll. Collecting simulation scenarios by analyzing physical test drives. In *IEEE Intelligent Transportation Systems Conference*, pages 2915–2920, 2015.

[23] J. Nilsson, M. Brännström, E. Coelingh, and J. Fredriksson. Lane change maneuvers for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1087–1096, 2017.

[24] D. Nister, H.-L. Lee, J. Ng, and Y. Wang. The safety force field. `https://www.nvidia.com/content/dam/en-zz/Solutions/self-driving-cars/safety-force-field/the-safety-force-field.pdf`, retrieved 22nd January 2020.

[25] A. Pretschner. Defect-based testing. In: Dependable Software Systems Engineering, 2015.

[26] A. Rizaldi et al. Formalising and monitoring traffic rules for autonomous vehicles in isabelle/hol. In *International Conference on Integrated Formal Methods*, pages 50–66. Springer, 2017.

[27] SAE. Definitions for terms related to on-road motor vehicle automated driving systems. *J3016, SAE International Standard*, 2014.

[28] S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a formal model of safe and scalable self-driving cars. *arXiv:1708.06374*, 2019.

[29] C. Sippl et al. From simulation data to test cases for fully automated driving and adas. In *IFIP International Conference on Testing Software and Systems*, pages 191–206. Springer, 2016.

[30] S. Ulbrich et al. Testing and validating tactical lane change behavior planning for automated driving. In *Automated Driving*, pages 451–471. Springer, 2017.

[31] U.S. National Highway Traffic Safety Administration. A framework for automated driving systems. `https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/`

13882-automateddrivingsystems_092618_v1a_tag.pdf, retrieved 22nd January 2020.

[32] W. Wachenfeld, P. Junietz, R. Wenzel, and H. Winner. The worst-time-to-collision metric for situation identification. In *IEEE Intelligent Vehicles Symposium*, pages 729–734, 2016.

[33] W. Wachenfeld and H. Winner. The release of autonomous vehicles. In *Autonomous Driving*, pages 425–449. Springer, 2016.

[34] J. Zhou and L. del Re. Identification of critical cases of adas safety by fot based parameterization of a catalogue. In *Control Conference (ASCC), 2017 11th Asian*, pages 453–458. IEEE, 2017.

# Part III.

# Related Work and Conclusion

# 7. Related Work

*This chapter discusses the related work in the domain of scenario-based testing. Parts of this chapter have previously appeared in peer-reviewed publications [54, 56, 57, 59], co-authored by the author of this thesis.*

Existing surveys [18, 27, 66, 78, 115, 132, 149, 150] provide an overview of the many facets and perspectives on the domain of verification and validation of automated and autonomous driving systems. In the following, we discuss in depth the existing works relevant for this thesis. We align the structure of this chapter to the methodology of this work as depicted in Fig. 7.1. Note that Sec. 7.2, 7.3, 7.5, and 7.7 reflect on the state-of-the-art to which this work is compared to, while Sec. 7.1, 7.4, and 7.6 name related works that focus on other parts of the overall test case generation process.

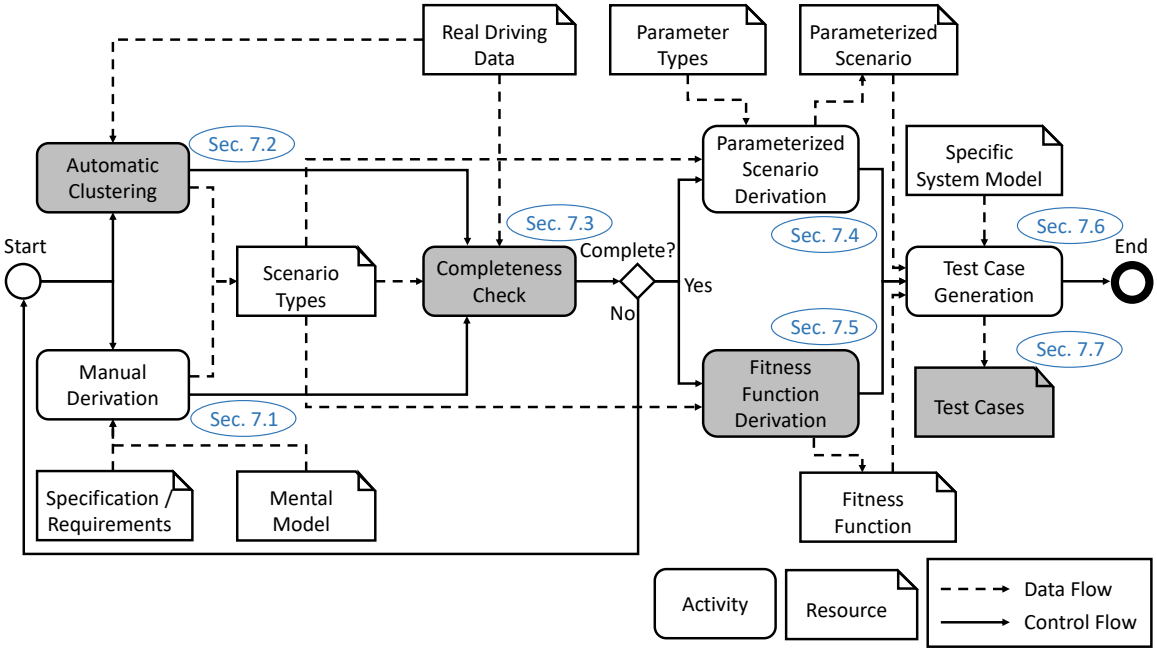



Figure 7.1.: Big Picture (previous versions appeared in [55, 59])

## 7.1. Manual Derivation

Literature suggests many different ways that support experts in the manual derivation of scenario types. They mostly aim at structuring and formalizing the experts' knowledge and often times exploit the formalism to infer (additional) scenario types.

In [143, 14], a layer structure for scenarios is suggested that is widely used. On the first layer, the *road network* is described, followed by the *traffic infrastructure* on level two and *temporal manipulations* of both previous levels on level three. Static and dynamic *objects* including their interactions are detailed on level four. The *environment*, meaning weather and lighting, is specified on level five.

Next to ad-hoc meta-models that allow the direct description of each individual scenario type [13] or a set of those with the classification tree method [12], various ontology-based formalisms have been proposed. In [37], an ontology is presented based on categories, e.g. *pedestrian crossing*, which are similar to scenario types. This formalism encourages to directly specify parameterized scenarios instead of the more abstract scenario types. In [99], ontologies are based on categorical enumerations, e.g. road surface may be dry, wet, or icy. By n-wise combination of such enumerations, scenario types are generated. Based on the layer structure, an ontology is created in [14] from which a large amount of different scenes can be automatically inferred. The scenes then serve as input for scenario type derivation.

In [7, 60, 150], so-called situation coverage is suggested, which translates to coverage of scenario types. Manually specified street layout snippets are combined to reach coverage.

Various technical reports present lists of scenario types [8, 32, 36, 112, 123, 158, 166, 178]. Those are of differing levels of abstraction and granularity as well as for various kinds of different operational design domains.

**Summary:** All these works help the expert in (manually) deriving scenario types by formalization of the expert's mental model.

## 7.2. Automated Clustering

A variety of works exist that present traffic data clustering with various different goals, e.g. traffic flow analysis, accident analysis, and understanding the driving task. This is reflected in the data, which ranges from abstract keyword databases to high resolution time series.

In the domain of traffic engineering, the goal is to understand the usage and demand of the road network [19] or of single road sections [33]. Both works cluster two-dimensional GPS position time series of individual vehicle trips from start to destination. Since the position time series of a single vehicle does not contain information about surrounding vehicles and since such a trip is composed of a multitude of scenario instances, their approach is not suitable to cluster single scenario instances with traffic interactions to scenario types. From a technical perspective, these approaches are limited to the two-dimensional GPS position time series.

In [98, 165], so-called "driving encounters" between two vehicles are clustered based on

vehicle position trajectories. The approach yields four [98] and ten [165] clusters where one cluster contains driving encounters at crossings, another clusters contains driving encounters in which vehicles approach each other on opposing lanes, and so on. Arguably, this level of granularity does not provide a sufficient level of detail to yield fine-grained scenario types. For instance, it ignores the various different ways how two or more vehicles may interact at a crossing. From a technical perspective, the approach is limited to two-vehicle interactions, while in reality scenario instances take place with more than two vehicles.

In [121], an approach is presented that clusters collision data to identify different types of collisions. Abstract, categorical information is used as input for the clustering, e.g. the gender of the driver or three categories of injuries. Intuitively, this approach is limited to the specific use case of collision data composed of categorical data. It is not applicable to (non-collision) driving data presented as time series. Similar to that, in [146], clustering is presented based on keywords for crash data specific to crossings, which has the same shortcomings. In [141], a combination of categorical and numeric values is used, e.g. the collision angle, of crash data for clustering. Since these (handcrafted) features are specific to crash data, the approach is limited to this kind of data.

The goal of [93] is to extract traffic scenario types from simulated driving data. This approach is limited to scenario instances of two seconds' length and interactions between two vehicles. The clustering is thereby based on handcrafted features, such as aggregations and characteristic points within the time series, e.g. whether or not a braking maneuver took place as well as the velocity of both vehicles at the start and at the end of the two second time span. In reality there are traffic scenarios (1) with more than two interacting vehicles and (2) usually such scenarios are longer than two seconds. Furthermore, handcrafted features come with the discussed shortcomings (see Sec. 1.1.1). In their follow-up work [94], they suggest an adapted approach that clusters scenario instances of varying length, but still on handcrafted features, e.g. whether the ego vehicle performed a lane change or not.

The approach of [152] avoids the use of handcrafted features. Scenario instances are represented as time series of fixed length and are clustered based on difference between those time series. Their approach requires the expert to manually select values for technique-specific parameters that are used to compute the amount and structure of the clusters. Thus, the structure and level of granularity of the scenario types is chosen by the expert based on the mental model, which contrasts the goal of reducing the influence of the mental model.

**Summary:** The existing works are either (1) technically limited, i.e. restrictions in the number of vehicles, the total duration of the scenario, and the type, number, and length of time series, or (2) heavily depend on the mental model of the expert, i.e. using handcrafted features or relying on the expert to compute shape and number of clusters.

## 7.3. Completeness Check (and Test Exit Criteria)

The existing works about completeness of testing and test exit criteria for automated and autonomous driving systems are roughly divided into two groups, namely (1) works that raise the concern that verification and validation solely by real test drives are not feasible and (2) works that provide constructive methods to assess testing completeness.

In [161], a statistical calculation is presented to show that verification and validation solely by real test drives is infeasible. The average driven kilometers between two fatal accidents on German highways are used to derive that $6.61$ billion kilometers need to be driven to ensure to encounter at least one of these scenarios. Similarly, in [63], fatal accidents are used as well. It is stated that tens of billions of kilometers are needed for direct measurement of sufficient events for statistical analysis. It is therefore suggested to use virtual scenarios instead. In another work [176] it is stated that millions or even billions of miles have to be driven to arrive at an acceptable level of certainty. Instead, it is suggested to accelerate testing by using mainly critical scenarios. Another work in this line [73] states that millions to billions of miles need to be driven. All these works did the important task of raising the awareness that verification and validation by real drive testing alone is not feasible. However, they do not provide a practically applicable test exit criterion.

In other works [137, 138] it is suggested that the driving system needs to be at least as good as the human driver. The driving behavior of a human driver is analyzed in all scenario types of a specific list of scenario types. An expected system behavior for those scenario types is derived. Then, the system performance is measured with respect to this expected behavior. This might be used as a test exit criterion for those specific scenario types: *Stop testing once it can be shown that the driving system is better than a human driver.* However, it cannot be used as a general test exit criterion, since the list of scenario types might not be complete. In this case, it cannot be argued that a system is safe, because it only performed better than a human driver in some scenarios. Such a comparative argument needs to be complemented by an argument that the list of scenario types is complete.

There exist works [38, 130] that analyze real drive data and generate for each scenario type a histogram of occurring instances. For example, a cut-in scenario is happening with different relative positions of the vehicles. Those distributions are then used for test case generation by using parameterized scenarios and selecting concrete values for parameters according to the distributions. However, they do not present a test exit criterion. They are missing an argument that the list of scenario types is complete for which this histogram-based testing is performed.

Constructive completeness argumentations for whether or not one has collected sufficient amounts of data for a single scenario type are presented in [39] and [164]. It is suggested to describe scenario types with the help of scenario type-specific parameters and deriving a kernel density function along those dimensions. These works focus on completeness on the level of single scenario types. Thus, they need to be complemented by a completeness argument on the level of a list of scenario types.

**Summary:** The one group of existing works that states that testing solely with real test

drives is not feasible do not provide a practically applicable test exit criterion. The other works lack an argument that all scenario types are known.

## 7.4. Parameterized Scenario Derivation

The derivation of parameterized scenarios comprises two steps: (1) the formalization of scenario types including the choice of the types of parameters and (2) the identification of the boundaries of the parameters' domains.

Various existing works [105, 167] provide abstract guidelines for the mostly manual process of formalizing scenario types and choosing the "appropriate" parameter types. Both works base their thoughts on the scenario layer model of [142] that tries to structure the way the parameter types are selected (see Sec. 7.1). In [50], it is suggested to use those parameter types the systems is sensitive to.

A variety of works [38, 70, 91, 170, 177] present data-driven approaches to identify the parameter domains' boundaries. In [38], parameters are used to describe the braking of a vehicle in front of the ego vehicle. Similarly, in [177], parameters are used to describe a lane change. For more complex models, in [91], an approach is presented to encode many scenario parameters into less scenario parameters with the help of autoencoders. Other works focus on complex crossing scenarios [70] and on complex highway scenarios with many cars [170].

**Summary:** All these works provide approaches and suggest a way to yield parameterized scenarios that best resemble and cover reality.

## 7.5. Fitness Function Derivation (and Test Case Selection With Search-Based Techniques)

Since the space of test cases spanned by the parameters of a parameterized scenario is huge, search-based techniques have been proposed for targeted test case selection. An abundance of works has been published, suggesting a variety of different approaches to test driving systems of all SAE levels [139] against various different (safety) criteria.

First works on search-based test scenario generation presented the idea of applying search-based techniques for the functional testing of advanced driver assistance systems, e.g. parking assistants [23] and braking assistants [24, 25]. This initial research focused on the technical aspects of the application of search-based techniques to these very specific driver assistance systems, using ad-hoc fitness functions that do not generalize and are not applicable to other or more complex driving systems.

Similarly, an approach is presented in [4] using an ad-hoc fitness function to test the interaction of a braking system and an adaptive cruise control system. In [147], an ad-hoc fitness function for another breaking system is suggested and, in [50], an experience report

is presented using ad-hoc fitness functions to test an adaptive cruise control system and a lane keeping assistance system.

For more complex driving systems of higher automation levels, a series of works [9, 79, 81] suggests to use search to minimize the safe drivable space during the scenario such that the driving system has a hard time obtaining a safe motion plan. Intuitively, such test cases are challenging. However, ensuring that scenario of a desired form is not the focus of these works. Note that in a later work of the same authors [80], which has been published after this thesis' publication on fitness functions [57], language snippets like "on lane" or "is behind" are provided to create specifications, which are translated to fitness functions ensuring correct scenario form.

Other works use ad-hoc fitness functions to guide the search to test cases in which the time to collision is small [28, 43, 44] or in which collisions occur, e.g. collisions in crossings [5], or on usual city roads [46], or on highways [155]. These ad-hoc fitness functions do not ensure that the selected test cases are of the desired form. Additionally, a collision or a small time to collision does not necessarily mean that the driving system behaves incorrectly or unsafe. Thus, the generated test cases are not necessarily "good".

To overcome this issue, two different methodologies are suggested by literature. On the one hand, searching for avoidable collisions is suggested [29, 30]. Knowing that a collision is avoidable means that in case of a collision, the system could be improved to actually avoid the collision. On the other hand, other works [61, 34] present an ad-hoc fitness function for a single scenario type. Those fitness functions make use of one safety aspect of the so-called responsibility-sensitive safety model [144]. By testing specifically against such a safety criterion, the safety of the driving system's behavior can be directly argued for. However, all those works do not ensure a desired scenario form. Additionally, these works [61, 34] provide their approaches only for a single scenario type without providing methodological guidance on how to create fitness functions that search for safety model-violating test cases for other scenario types.

Even though the concept of targeted search promises to be more efficient than random test selection, it is still very costly to run all the test cases as part of the optimization. Several technical improvements have been suggested. In [3] and [17], so-called surrogate models are used to learn the fitness value space. This way, running test cases can be avoided if they are predicted to yield a bad fitness value. Another idea is to create decision trees during the search [2] to provide information to the engineer about which regions of the parameter space spanned by the parameterized scenario results in interesting test cases. Other works [88, 108, 109, 153] focus on reducing the simulation time of the test cases by executing test cases only partially and re-using such parts for future test cases. This way, only missing parts of future test cases need to be run. While all of these technical improvements are important and show great results [97], these works assume the fitness function to be given or create them ad-hoc. Neither of them addresses the methodological aspect of how fitness functions are correctly created.

Another line of research [10, 58, 65, 100, 103, 104, 127, 159, 171] aims at directly generating the input signals to controllers (and driver assistance systems) instead of generating full-

blown test scenarios. They are not applicable to scenario-based testing of complex driving systems of higher automation levels.

Scenario-based testing is also used in other domains, e.g. to test unmanned aerial vehicles [20, 180], cleaning robots [116], and unmanned underwater vehicles [110, 111]. Obviously, those systems are vastly different from automated driving systems.

**Summary:** Many of the existing works are not applicable to complex driving systems. Other works either provide ad-hoc fitness functions for a specific system or scenario type or focus on the technical aspects of search-based test scenario generation assuming the fitness function to be given. Both leave the methodological aspect unconsidered of how fitness functions generally should be created. Especially, ensuring a desired scenario form with the help of fitness functions is left unaddressed. Note that there is an exception: In [80], which has been published after this thesis' publication on fitness functions [57], language snippets like "on lane" or "is behind" are provided to create specifications, which are translated to fitness functions ensuring correct scenario form.

## 7.6. Test Case Generation (Without Search)

Next to search-based test scenario generation, a variety of other approaches have been suggested that transfer various ideas of classic software testing to driving systems.

In [12], parameter space coverage is suggested, meaning that the parameter space is discretized and all variations are covered. A similar approach is to perform uniform random sampling on the parameter space [76, 102] until, e.g., a certain number of test cases is reached or some sort of coverage of the parameter space is achieved. In addition to uniformly random sampling, other works [6, 168] suggest to use statistical sampling to yield a distribution of test scenarios that is similar to the distribution of scenario instances in reality. To reduce the number of necessary test cases needed for simple parameter space coverage, some works [136, 154, 173] suggest to discretize the parameter space and select test cases to yield n-wise combinatorial coverage. However, parameter space coverage is very similar to gathering miles in real drive test cases: Run test cases until all possible traffic situations are covered. The necessary number of test cases needed to achieve such a coverage is just too large to be practically feasible. Note that the amount of necessary test cases depends on the granularity of the discretization, meaning that a very coarse discretization requires only very few test cases. However, the coarser the discretization, the less meaningful is the coverage.

**Summary:** Test case selection to achieve some sort of parameter space coverage is practically infeasible. This is why many works suggest to use targeted test case selection using search-based techniques.

## 7.7. (Re-Usability of) Test Cases

Several existing works are based on the idea of extracting scenario instances, also called concrete test scenarios, from data according to a variety of different selection and filtering criteria. This data is collected with real test drives [11, 35, 95, 107, 179] or simulation setups [145]. Similar to manual test case generation based on experience, the result of those approaches are "test catalogs" of test cases [8]. These are subject to the presented re-usability issue. Additionally, the extracted scenarios are randomly encountered scenarios, which is problematic for safety argumentations, as such are hardly possible based on randomly encountered scenarios. An infeasible amount of driving hours or driven kilometer is necessary for each version and variant of the system [63, 73, 161, 176].

A very popular idea to circumvent this issue is to extract all scenario instances that occur in real traffic data and filter for the "critical" ones, where a multitude of distinct metrics of criticality exist [71, 96, 122, 160, 162, 163, 169]. The resulting "test catalogs" are relatively small as they contain only the critical scenario instances. When the amount of data is huge, there is the hope that the resulting "test catalog" overcomes the randomness of encountering certain scenario instances in real traffic. However, it is still subject to the re-usability issue. The critical scenario instances are critical with respect to the behavior of the recording test vehicle or the driver maneuvering the recording vehicle. When such scenario instances are re-used for testing another vehicle, e.g. the next generation automated or autonomous vehicle, it is not guaranteed that the scenario instances are still critical. It might be an easy non-critical test scenario for the new system. Note that such criticality metrics might be used for targeted generation of scenario instances, e.g. as fitness function for the application of search-based techniques. In this case, the usage of such criticality metrics is not an instance of the re-usability issue.

There is one work [22] that discusses the general re-use of test cases, which has been published after the publications of this thesis. In [22], it is stated that test cases may yield different results when used in different simulation environments, since simulation environments may use different simulation models. However, they do not discuss the re-usability of test cases among different system versions and variants.

**Summary:** Re-usability of concrete test scenarios among different system version and variants is implicitly assumed by a variety of works. However, to the best of author's knowledge there is no work actually providing evidence for or against this assumption.

# 8. Conclusion and Outlook

*This chapter concludes this work by summarizing the results and their limitations. After presenting lessons learnt, potential future steps are discussed. Parts of this chapter have previously appeared in peer-reviewed publications [54, 56, 57, 59], co-authored by the author of this thesis.*

## 8.1. Summary of Results and Limitations

In this work, an approach is shown for scenario-based testing of automated and autonomous driving systems while considering completeness. The goal is to test the driving system in the most difficult scenario instances of "all" scenario types. This requires that "all" scenario types are known (**Problem 1**) and that the systems is tested in "good" test cases (**Problem 2**), which are instances of those scenario types.

**Addressing Problem 1:**

Scenario types are usually manually derived by experts. To complement this manual process, it is suggested to use automated clustering as redundancy. However, existing works are technically limited or rely on the expert's mental model (**Gap 1**). The presented clustering approach clusters traffic scenario instances solely based on syntactic similarity, i.e. by comparing the time series that describe dynamic aspects of the scenario instances like the position of traffic participants. This reduces the influence of the expert's mental model to the choice of what kind of data is recorded. Note that this influence can inherently not be avoided. By application of the approach to a data set of highway scenario instances with two lanes with 346 instances and three lanes with 414 instances, 57 clusters representing 38 manually labeled scenario types and 78 clusters representing 67 manually labeled scenario types could be found. Comparing the granularity of the automatically derived scenario types with literature's lists of scenarios types, we found the automatically derived scenario types to be more granular than [67, 158, 166] and similar granular as [36, 178]; we could not find a more granular list. **Limitations:** As it is in the nature of recorded data, the presented approach depends on the expert's choice of what kind of data is recorded (and used) as for the feature computation during the clustering. Thus, the influence of the expert's mental model is not completely avoided. Here, relative positions of vehicles to each other have been used as data for feature computation. From a technical perspective, there may be more appropriate techniques to determine the number and content of the clusters than the ones used for this approach. Further, the evaluation is limited to data from a 420m section

of straight highway, which is not representative for the whole driving task. Finally, the approach is limited in its scalability, since the clusters require manual inspection.

Once the derivation of scenario types is finished, the resulting list of scenario types needs to be assessed for completeness. Since an absolute measurement is infeasible, the goal is an assessment relative to collected traffic data. However, existing works either provide arguments that the classic test exit criterion, namely *gathering miles in real traffic*, is not feasible or analyze completeness of data for a single scenario type (**Gap 2**). With the described statistical model, lists of such scenario types can be analyzed for completeness relative to a given traffic data set. An adapted version of the Coupon Collector's Problem considers the existence of an unknown scenario type and estimates based on a histogram of occurrence probabilities (extracted from recorded traffic data) whether this hypothetical unknown scenario type should have been seen already. Depending on the statistical guarantees one would like to have, this approach provides the amount of data that is necessary to assume completeness relative to this data set. The statistical model is applied to hand-crafted, (artificial) distributions of scenario type occurrence rates as well as to one suggested by literature. **Limitations:** The histogram of occurrence probabilities changes with the level of granularity of the scenario types. Instead of a very abstract scenario type *lane change*, one could also use two — still very abstract — scenario types *lane change to the left* and *lane change to the right*, or many more granular ones. As a result, the histogram contains more or less scenario types with respective occurrence probabilities, which results in different amounts of necessary data to be considered complete. Thus, granularity influences the assessment result. However, it is generally unclear what an adequate level of granularity is. Further, the provided approach assumes the existence of this histogram of occurrence probabilities of scenario types, which is difficult to acquire. Additionally, the application of the presented model requires the specification of the occurrence probability up to which it is desired for scenario types to be known. Choosing an adequate value is left to the expert.
**Addressing Problem 2:**

The scenario types are fundamentally important for test case generation. Based on those types, search spaces and fitness functions are created for the application of search-based techniques to identify "good" test cases. However, existing works either focus on technical aspects of the search or assume the fitness function to be given or create it ad-hoc. Especially, ensuring a desired scenario form with the help of fitness functions is left unconsidered (**Gap 3**). The presented approach provides methodological guidance for the creation of fitness functions. Three templates for space, time, and safe operating envelope violation are provided that can be combined to ensure that the identified test case is of the desired form and is interesting, meaning that the system under test approaches the boundaries of the safe operating envelope. The templates are evaluated for highway scenario types suggested by literature [178] by generating test cases for those types. A prototypical highway pilot driving system based on [117, 118, 119] served as system under test. **Limitations:** The templates are evaluated using 24 scenario types of [178], which are straight-line highway scenario types. Not contained are special highway scenario types like exist ramps or construction sites as well as city center scenario types. It is expected that the templates may need slight

adaption for such other highway scenario types or inner city scenario types, e.g. crossings. Further, the idea of testing the driving systems in the most challenging test case assumes that the search-based techniques actually identify the global optimum in the search space or at least some candidate close to it. However, heuristic search is not perfect and may get stuck in local optima. Finally, since the fitness function is used to ensure that test cases are of the desired form, it depends on the structure and level of granularity of the scenario types, which is an open problem.

Once a system is tested and potential problems are resolved, it needs to be tested again. One might want to naïvely re-use the already generated test cases. Many existing works implicitly rely on the assumption that this is possible. However, no one shows evidence for or against this assumption (**Gap 4**). The presented counter-example conveys that such a naïve re-use is generally a bad idea if there is no evidence that a "good" test case stays "good" when re-used. We apply the presented fitness function templates to generate "good" test cases for three system versions. Then, the best test case identified for each of those versions is "re-used" for the other two. By showing that a re-used test case is not able to reveal faulty behavior, while a newly generated test case does so, the system-dependence of test cases is shown and a counter-example to naïve re-use is provided. **Limitations:** The presented experiments are a single counterexample. It does not provide constructive analysis how it can be determined which existing "good" test cases are not naïvely re-usable because of a change in the system.

## 8.2. Lessons Learned

As stated in [64], testing is not only about revealing faulty behavior, but also to increase the confidence that the behavior of the system under test is correct. While this statement is made in a different context, it holds for the domain of scenario-based testing of automated and autonomous driving systems. It is the fundamental principle of several lessons learned throughout working on this thesis, which are named in the context of, but – of course – are not necessarily limited to scenario-based testing:

Having an explicit definition of the meaning of a "good" test case is key. Only then can the quality of a test case and the quality of the system behavior be measured quantitatively (according to the definition). This is fundamental for two reasons: (1) It usually allows for an automated oracle and (2) without knowing how good or bad a test case is, it is generally unclear how much it increases the confidence that the systems behaves correctly.

Since those test cases as the result of the test case generation process are needed for the release argumentation of the system under test, one should aim for a structured and simple to interpret methodology and process. An unstructured approach that, e.g., uses trial and error until the results look "as expected" is not providing any confidence. Similarly, even if failing test cases are generated for some scenario types, no confidence is gained if the approach is hardly interpretable.

Completeness and adequacy are fundamental. A good testing approach needs to incor-

porate validity checks for the inputs to the test case generation process, intermediate results in between sequential steps, and the final test cases. The best techniques integrated into the test case generation process will not yield useful results if they are applied on bad inputs. For instance the recorded traffic data that serves as input to the scenario clustering needs to be of high quality; otherwise the results will be meaningless. Analogously, if the fitness function or search space - both input to the search-based test case generation - are bad, the search cannot find any "good" test case and the identified test cases are meaningless.

## 8.3. Outlook and Future Work

While this work addresses a variety of problems and gaps, several issues stay unsolved and may be aimed at by future work. The majority of the following open problems are fundamental not only for the approaches presented in this work, but also for many other existing approaches as well.

**Suitable level of scenario granularity** The level of granularity determines the number of scenario types, which means that most of the steps in the test case generation process are affected by the granularity. Many of the statistical models for completeness assessment are influenced by the number of scenario types. Similarly, the granularity directly affects the parameter space of parameterized scenarios as well as the test selection criterion (i.e. the fitness functions in this work) and, thus, indirectly the test case generation. Further investigation is necessary to identify a suitable level of scenario granularity.

**Extended and integrated statistical assessment** Ideally, one would like to receive an overall, sound statistical guarantee that a driving system is behaving safe in every possible traffic situation it may encounter. While this might be too ambitious, several steps in the test case generation process may be enhanced with statistical analysis. For instance one would like to estimate how well the test case generation technique performed in identifying the relevant test cases in the parameter space of a parameterized scenario. Such statistical analyses may then be combined to an overall statistical assessment.

**Adequacy of a parameterized scenarios** Many existing works in the domain of scenario-based testing rely on parameterized scenario for test case generation. Their creation usually involves a lot of manual effort. Intuitively, the test case selection can only select test cases that are actually contained in the parameter space of a parameterized scenario. Thus, validation of the parameterized scenarios' completeness and adequacy is necessary.

**Fitness function templates for other highway and city center scenarios** More specific to this work's approach is the idea to extend the presented fitness function templates to other highway scenario types, e.g. exit ramps and construction sites, as well as to non-highway

scenarios, e.g. crossings or roundabouts in city centers. While it is expected that the core notion of time, space, and safety (distance) violation does not change, the templates may require adaption to be applicable to such scenario types.

**Investigation of the re-usability issue** While this work presents a counterexample to the naïve re-usability of test cases, it may still be very well possible to utilize historical test data or system change information to (1) assess the re-usability of test cases or (2) improve the cost-effectiveness of the generation of new test cases.

**Adaptions for future driving systems** Future driving systems may render some parts of the presented test case generation process obsolete or inapplicable. For instance, the notion of safety may change when driving systems cooperate with each other. Thus, adaptions to the presented approaches may be necessary, i.e. new scenario types are added and refined parameterized scenarios and fitness functions are required.

# Bibliography


[1] Road vehicles - functional safety (iso 26262). Standard, International Organization for Standardization (ISO), 2018.

[2] Raja Ben Abdessalem, Shiva Nejati, Lionel Briand, and Thomas Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *Proceedings of the 40th International Conference on Software Engineering (ICSE)*. ACM, 2018.

[3] Raja Ben Abdessalem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Testing advanced driver assistance systems using multi-objective search and neural networks. In *31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 63–74, 2016.

[4] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Testing autonomous cars for feature interaction failures using many-objective search. In *33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 143–154, 2018.

[5] Yasasa Abeysirigoonawardena, Florian Shkurti, and Gregory Dudek. Generating adversarial driving scenarios in high-fidelity simulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8271–8277. IEEE, 2019.

[6] Yasuhiro Akagi, Ryosuke Kato, Sou Kitajima, Jacobo Antona-Makoshi, and Nobuyuki Uchida. A risk-index based sampling method to generate scenarios for the evaluation of automated driving vehicle safety. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 667–672. IEEE, 2019.

[7] Rob Alexander, Heather Rebecca Hawkins, and Andrew John Rae. Situation coverage–a coverage criterion for testing autonomous robots. 2015.

[8] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 719–726. IEEE, 2017.

[9] Matthias Althoff and Sebastian Lutz. Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1326–1333. IEEE, 2018.

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

[11] Johannes Bach, Marc Holzäpfel, Stefan Otten, and Eric Sax. Reactive-replay approach for verification and validation of closed-loop control systems in early development. Technical report, SAE Technical Paper, 2017.

[12] Johannes Bach, Jacob Langner, Stefan Otten, Eric Sax, and Marc Holzäpfel. Test scenario selection for system-level verification and validation of geolocation-dependent automotive control systems. In *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 203–210. IEEE, 2017.

[13] Johannes Bach, Stefan Otten, and Eric Sax. Model based scenario specification for development and test of automated driving functions. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1149–1155, 2016.

[14] Gerrit Bagschik, Till Menzel, and Markus Maurer. Ontology based scene creation for the development of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1813–1820. IEEE, 2018.

[15] Gerrit Bagschik, Till Menzel, Andreas Reschka, and Markus Maurer. Szenarien für entwicklung, absicherung und test von automatisierten fahrzeugen. In *11. Workshop Fahrerassistenzsysteme. Hrsg. von Uni-DAS e. V*, pages 125–135, 2017.

[16] Gerrit Bagschik, Andreas Reschka, Torben Stolte, and Markus Maurer. Identification of potential hazardous events for an unmanned protective vehicle. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, pages 691–697. IEEE, 2016.

[17] Halil Beglerovic, Michael Stolz, and Martin Horn. Testing of autonomous vehicles using surrogate models and stochastic optimization. In *IEEE 2017 Intelligent Transportation Systems Conference (ITSC)*, pages 1–6, 2017.

[18] Klaus Bengler, Klaus Dietmayer, Berthold Farber, Markus Maurer, Christoph Stiller, and Hermann Winner. Three decades of driver assistance systems: Review and future perspectives. *IEEE Intelligent Transportation Systems Magazine*, 6(4):6–22, 2014.

[19] Philippe C Besse, Brendan Guillouet, Jean-Michel Loubes, and François Royer. Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3306–3317, 2016.

[20] Kevin M Betts and Mikel D Petty. Automated search-based robustness testing for autonomous vehicle software. *Modelling and Simulation in Engineering*, 2016, 2016.

[21] Julian Bock, Robert Krajewski, Tobias Moers, Lennart Vater, Steffen Runde, and Lutz Eckstein. The ind dataset: A drone dataset of naturalistic vehicle trajectories at german intersections. 2019.

[22] Markus Borg, Raja Ben Abdessalem, Shiva Nejati, Francois-Xavier Jegeden, and Donghwan Shin. Generating avoidable collision scenarios for testing autonomous driving systems. In *IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 2021. to appear.

[23] Oliver Bühler and Joachim Wegener. Evolutionary functional testing of an automated parking system. In *Proceedings of the International Conference on Computer, Communication and Control Technologies (CCCT) and the 9th. International Conference on Information Systems Analysis and Synthesis (ISAS)*, 2003.

[24] Oliver Bühler and Joachim Wegener. Evolutionary functional testing of a vehicle brake assistant system. In *6th Metaheuristics International Conference, Vienna, Austria*, 2005.

[25] Oliver Bühler and Joachim Wegener. Evolutionary functional testing. *Computers & Operations Research*, 35(10):3144–3160, 2008.

[26] Simon Burton, Lydia Gauerhof, and Christian Heinzemann. Making the case for safety of machine learning in highly automated driving. In *International Conference on Computer Safety, Reliability, and Security*, pages 5–16. Springer, 2017.

[27] Simon Burton and Richard Hawkins. Assuring the safety of highly automated driving: state-of-the-art and research perspectives. online at `https://www.york.ac.uk/assuring-autonomy/news/publications/highly-automated-driving-assurance/`, retrieved 5th October 2020.

[28] Andreas Bussler, Lukas Hartjen, Robin Philipp, and Fabian Schuldt. Application of evolutionary algorithms and criticality metrics for the verification and validation of automated driving systems at urban intersections. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 128–135, 2020.

[29] A Calo, P Arcaini, S Ali, F Hauer, and I Fuyuki. Simultaneously searching and solving multiple avoidable collisions for testing autonomous driving systems. In *Genetic and Evolutionary Computation Conference*, 2020.

[30] Alessandro Calo, Paolo Arcaini, Shaukat Ali, Florian Hauer, and Ishikawa Fuyuki. Generating avoidable collision scenarios for testing autonomous driving systems. In *IEEE International Conference on Software Testing, Verification and Validation (ICST)*.

[31] CarMaker. *version 8.1*. IPG Automotive GmbH, Karlsruhe, Germany, 2019.

[32] Catapult Transport Systems. Taxonomy of scenarios for automated driving. online at `https://s3-eu-west-1.amazonaws.com/media.ts.catapult/wp-content/uploads/2017/04/25114137/ATS34-Taxonomy-of-Scenarios-for-Automated-Driving.pdf`, retrieved 5th October 2020.

[33] Mei Yeen Choong et al. Modeling of vehicle trajectory clustering based on lcss for traffic pattern extraction. In *IEEE International Conference on Automatic Control and Intelligent Systems*, pages 74–79, 2017.

[34] Anthony Corso, Peter Du, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Adaptive stress testing with reward augmentation for autonomous vehicle validatio. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 163–168. IEEE, 2019.

[35] Erwin de Gelder, Jeroen Manders, Corrado Grappiolo, Jan-Pieter Paardekooper, Olaf Op den Camp, and Bart De Schutter. Real-world scenario mining for the assessment of automated vehicles. *arXiv preprint arXiv:2006.00483*, 2020.

[36] Erwin de Gelder, Olaf Op den Camp, and Niels de Boer. Scenario categories for the assessment of automated vehicles. online at `http://cetran.sg/publications/`, retrieved 5th October 2020.

[37] Erwin De Gelder, J-P Paardekooper, A Khabbaz Saberi, H Elrofai, J Ploeg, L Friedmann, B De Schutter, et al. Ontology for scenarios for the assessment of automated vehicles. *arXiv preprint arXiv:2001.11507*, 2020.

[38] Erwin de Gelder and Jan-Pieter Paardekooper. Assessment of automated driving systems using real-life scenarios. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 589–594. IEEE, 2017.

[39] Erwin de Gelder, Jan-Pieter Paardekooper, Olaf Op den Camp, and Bart De Schutter. Safety assessment of automated vehicles: how to determine whether we have collected enough field data? *Traffic injury prevention*, 20(sup1):S162–S170, 2019.

[40] Christian Domsch and Herbert Negele. Einsatz von referenzfahrsituationen bei der entwicklung von fahrerassistenzsystemen. *Aktive Sicherheit durch Fahrerassistenz*, 3:7–8, 2008.

[41] Economic Commission for Europe - Inland Transport Committee. Convention on road traffic, 1968. Vienna.

[42] Federal Republic of Germany. Straßenverkehrsordnung (English: Road Traffic Regulation), 2017.

[43] Shuo Feng, Yiheng Feng, Haowei Sun, Shan Bao, Yi Zhang, and Henry X Liu. Testing scenario library generation for connected and automated vehicles, part ii: Case studies. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[44] Shuo Feng, Yiheng Feng, Chunhui Yu, Yi Zhang, and Henry X Liu. Testing scenario library generation for connected and automated vehicles, part i: Methodology. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[45] Bundesanstalt für Straßenwesen. Rechtsfolgen zunehmender fahrzeugautomatisierung. *Forschung Kompakt*, 2012.

[46] Briti Gangopadhyay, Siddartha Khastgir, Sumanta Dey, Pallab Dasgupta, Giovanni Montana, and Paul Jennings. Identification of test cases for automated driving systems using bayesian optimization. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1961–1967. IEEE, 2019.

[47] Douglas Gettman and Larry Head. Surrogate safety measures from traffic simulation models. *Transportation Research Record: Journal of the Transportation Research Board*, (1840):104–115, 2003.

[48] Olaf Gietelink, Jeroen Ploeg, Bart De Schutter, and Michel Verhaegen. Testing advanced driver assistance systems for fault management with the vehil test facility. In *Proceedings Of The 7th International Symposium On Advanced Vehicle Control (AVEC'04)*, pages 579–584, 2004.

[49] Olaf Gietelink, Jeroen Ploeg, Bart De Schutter, and Michel Verhaegen. Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations. *Vehicle System Dynamics*, 44(7):569–590, 2006.

[50] Christoph Gladisch, Thomas Heinz, Christian Heinzemann, Jens Oehlerking, Anne von Vietinghoff, and Tim Pfitzer. Experience paper: Search-based testing in automated driving control applications. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 26–37. IEEE, 2019.

[51] VIRES Simulationstechnologie GmbH. OpenDrive 1.5. Technical report, online at `http://www.opendrive.org/`, retrieved 5th October 2020, 2019.

[52] Junyao Guo, Unmesh Kurup, and Mohak Shah. Is it safe to drive? an overview of factors, metrics, and datasets for driveability assessment in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[53] Jonathan M Hankey, Miguel A Perez, and Julie A McClafferty. Description of the shrp 2 naturalistic database and the crash, near-crash, and baseline data sets. Technical report, Virginia Tech Transportation Institute, 2016.

[54] Florian Hauer, Ilias Gerostathopoulos, Tabea Schmidt, and Alexander Pretschner. Clustering traffic scenarios using mental models as little as possible. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1007–1012. IEEE.

[55] Florian Hauer and Bernd Holzmüller. A sound approach to scenario-based testing as the basis for safety argumentations. *Automotive Testing Technology Magazine*, September:93–94, 2020.

[56] Florian Hauer, Alexander Pretschner, and Bernd Holzmüller. Re-using concrete test scenarios generally is a bad idea. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1305–1310. IEEE.

[57] Florian Hauer, Alexander Pretschner, and Bernd Holzmüller. Fitness functions for testing automated and autonomous driving systems. In *International Conference on Computer Safety, Reliability, and Security*, pages 69–84. Springer, 2019.

[58] Florian Hauer, Alexander Pretschner, Maximilian Schmitt, and Markus Groetsch. Industrial evaluation of search-based test generation techniques for control systems. In *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 5–8. IEEE, 2017.

[59] Florian Hauer, Tabea Schmidt, Bernd Holzmüller, and Alexander Pretschner. Did we test all scenarios for automated and autonomous driving systems? In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2950–2955, 2019.

[60] Heather Hawkins and Rob Alexander. Situation coverage testing for a simulated autonomous car–an initial case study. *arXiv preprint arXiv:1911.06501*, 2019.

[61] Mohammad Hekmatnejad, Bardh Hoxha, and Georgios Fainekos. Search-based test-case generation by monitoring responsibility safety rules. *arXiv preprint arXiv:2005.00326*, 2020.

[62] Philipp Helle, Wladimir Schamai, and Carsten Strobel. Testing of autonomous systems–challenges and current state-of-the-art. In *INCOSE International Symposium*, volume 26, pages 571–584. Wiley Online Library, 2016.

[63] Thomas Helmer, Lei Wang, Klaus Kompass, and Ronald Kates. Safety performance assessment of assisted and automated driving by virtual experiments: Stochastic microscopic traffic simulation as knowledge synthesis. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2019–2023, 2015.

[64] Dominik Holling. *Defect-based Quality Assurance with Defect Models*. PhD thesis, Technische Universität München, 2016.

[65] Dominik Holling, Alvin Stanescu, Kristian Beckers, Alexander Pretschner, and Matthias Gemmar. Failure models for testing continuous controllers. In *Software Reliability Engineering (ISSRE), 2016 IEEE 27th International Symposium on*, pages 365–375. IEEE, 2016.

[66] WuLing Huang, Kunfeng Wang, Yisheng Lv, and FengHua Zhu. Autonomous vehicles testing methods review. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 163–168. IEEE, 2016.

[67] Hardi Hungar, Frank Köster, and Jens Mazzega. Test specifications for highly automated driving functions: Highway pilot. 2017.

[68] OpenScenario Initiative. OpenScenario 1.0.0. Technical report, online at `http://www.openscenario.org`, retrieved 5th October 2020, 2020.

[69] Stefan Jesenski, Jan Erik Stellet, Wolfgang Branz, and J Marius Zöllner. Simulation-based methods for validation of automated driving: A model-based analysis and an overview about methods for implementation. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1914–1921. IEEE, 2019.

[70] Stefan Jesenski, Jan Erik Stellet, Florian Schiegg, and J Marius Zöllner. Generation of scenes in intersections for the validation of highly automated driving functions. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 502–509. IEEE, 2019.

[71] Philipp Junietz, Farid Bonakdar, Björn Klamann, and Hermann Winner. Criticality metric for the safety validation of automated driving using model predictive trajectory optimization. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 60–65, 2018.

[72] Philipp Junietz, Jan Schneider, and Hermann Winner. Metrik zur bewertung der kritikalität von verkehrssituationen und -szenarien. In *11. Workshop Fahrerassistenzsysteme und automatisiertes Fahren*, 2017.

[73] Nidhi Kalra and Susan M Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016.

[74] Yue Kang, Hang Yin, and Christian Berger. Test your self-driving algorithm: An overview of publicly available driving datasets and virtual testing environments. *IEEE Transactions on Intelligent Vehicles*, 4(2):171–185, 2019.

[75] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[76] Siddartha Khastgir, Gunwant Dhadyalla, Stewart Birrell, Sean Redmond, Ross Addinall, and Paul Jennings. Test scenario generation for driving simulators using constrained randomization technique. Technical report, SAE Technical Paper, 2017.

[77] Yeeun Kim, Sehyun Tak, Jeongyun Kim, and Hwasoo Yeo. Identifying major accident scenarios in intersection and evaluation of collision warning system. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2017.

[78] Christian King, Lennart Ries, Jacob Langner, and Eric Sax. A taxonomy and survey on validation approaches for automated driving systems. In *2020 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–8. IEEE.

[79] Moritz Klischat and Matthias Althoff. Generating critical test scenarios for automated vehicles with evolutionary algorithms. In *Proc. of the IEEE Intelligent Vehicles Symposium*, 2019.

[80] Moritz Klischat and Matthias Althoff. Synthesizing traffic scenarios from formal specifications for testing automated vehicles. In *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020.

[81] Moritz Klischat, Edmond Irani Liu, Fabian Höltke, and Matthias Althoff. Scenario factory: Creating safety-critical traffic scenarios for automated vehicles. page to appear, 2020.

[82] Alessia Knauss, Jan Schroder, Christian Berger, and Henrik Eriksson. Software-related challenges of testing automated vehicles. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 328–330. IEEE, 2017.

[83] Philip Koopman, Aaron Kane, and Jen Black. Credible autonomy safety argumentation. In *27th Safety-Critical Systems Symposium*, 2019.

[84] Philip Koopman, Beth Osyk, and Jack Weast. Autonomous vehicles meet the physical world: Rss, variability, uncertainty, and proving safety (expanded version). *arXiv preprint arXiv:1911.01207*, 2019.

[85] Philip Koopman and Michael Wagner. Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety*, 4(1):15–24, 2016.

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

[87] Philip Koopman and Michael Wagner. Toward a framework for highly automated vehicle safety validation. Technical report, SAE Technical Paper, 2018.

[88] Markus Koschi, Christian Pek, Sebastian Maierhofer, and Matthias Althoff. Computationally efficient safety falsification of adaptive cruise control systems. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2879–2886. IEEE, 2019.

[89] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2118–2125, 2018.

[90] Robert Krajewski, Tobias Moers, Julian Bock, Lennart Vater, and Lutz Eckstein. The round dataset: A drone dataset of road user trajectories at roundabouts in germany. submitted.

[91] Robert Krajewski, Tobias Moers, Dominik Nerger, and Lutz Eckstein. Data-driven maneuver modeling using generative adversarial networks and variational autoencoders for safety validation of highly automated vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2383–2390. IEEE, 2018.

[92] Annkathrin Krämmer, Christoph Schöller, Dhiraj Gulati, and Alois Knoll. Providentia - a large scale sensing system for the assistance of autonomous vehicles. *Robotics Science and Systems Workshops (RSS Workshops)*, 2019.

[93] Friedrich Kruber, Jonas Wurst, and Michael Botsch. An unsupervised random forest clustering technique for automatic traffic scenario categorization. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2811–2818, 2018.

[94] Friedrich Kruber, Jonas Wurst, Eduardo Sánchez Morales, Samarjit Chakraborty, and Michael Botsch. Unsupervised and supervised learning with the random forest algorithm for traffic scenario clustering and classification. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2463–2470. IEEE, 2019.

[95] Ulrich Lages, Martin Spencer, and Roman Katz. Automatic scenario generation based on laserscanner reference data and advanced offline processing. In *IEEE Intelligent Vehicles Symposium Workshops*, pages 146–148, 2013.

[96] Matthias Lehmann, Maximilian Bäumler, Günther Prokop, and Diana Hamelow. Use of a criticality metric for assessment of critical traffic situations as part of sepia. In *19. Internationales Stuttgarter Symposium*, pages 1154–1167. Springer, 2019.

[97] Guanpeng Li, Yiran Li, Saurabh Jha, Timothy Tsai, Michael Sullivan, Siva Kumar Sastry Hari, Zbigniew Kalbarczyk, and Ravishankar Iyer. Av-fuzzer: Finding safety violations in autonomous driving systems. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 25–36. IEEE, 2020.

[98] Sisi Li, Wenshuo Wang, Zhaobin Mo, and Ding Zhao. Cluster naturalistic driving encounters using deep unsupervised learning. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1354–1359, 2018.

[99] Yihao Li, Jianbo Tao, and Franz Wotawa. Ontology-based test generation for automated and autonomous driving functions. *Information and Software Technology*, 117:106200, 2020.

[100] Felix Lindlar, Andreas Windisch, and Joachim Wegener. Integrating model-based testing with evolutionary functional testing. In *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on*, pages 163–172. IEEE, 2010.

[101] Waymo LLC. Waymo open dataset. online at `https://waymo.com/open/`, retrieved 5th October 2020.

[102] Rupak Majumdar, Aman Mathur, Marcus Pirron, Laura Stegner, and Damien Zufferey. Paracosm: A language and tool for testing autonomous driving systems. *arXiv preprint arXiv:1902.01084*, 2019.

[103] Reza Matinnejad, Shiva Nejati, Lionel C Briand, and Thomas Bruckmann. Automated test suite generation for time-continuous simulink models. In *Proceedings of the 38th international conference on software engineering*, pages 595–606. ACM, 2016.

[104] Reza Matinnejad, Shiva Nejati, Lionel C Briand, and Thomas Bruckmann. Test generation and test prioritization for simulink models with dynamic behavior. *IEEE Transactions on Software Engineering*, 45(9):919–944, 2018.

[105] Till Menzel, Gerrit Bagschik, Leon Isensee, Andre Schomburg, and Markus Maurer. From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2383–2390. IEEE, 2019.

[106] Till Menzel, Gerrit Bagschik, and Markus Maurer. Scenarios for development, test and validation of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1821–1827. IEEE, 2018.

[107] Pascal Minnerup, Tobias Kessler, and Alois Knoll. Collecting simulation scenarios by analyzing physical test drives. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2915–2920, 2015.

[108] Pascal Minnerup and Alois Knoll. Testing autonomous driving systems against sensor and actuator error combinations. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 561–566. IEEE, 2014.

[109] Pascal Minnerup and Alois Knoll. Temporal logic for finding undesired behaviors of autonomous vehicles in a state space explored by dynamic analysis. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, pages 1248–1253. IEEE, 2016.

[110] Galen E Mullins, Paul G Stankiewicz, and Satyandra K Gupta. Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles. In *IEE International Conference on Robotics and Automation (ICRA)*, pages 1443–1450, 2017.

[111] Galen E Mullins, Paul G Stankiewicz, R Chad Hawthorne, Jordan D Appler, Michael H Biggins, Kevin Chiou, Melissa A Huntley, Johan D Stewart, and Adam S Watkins. Delivering test and evaluation tools for autonomous unmanned vehicles to the fleet. *Johns Hopkins APL technical digest*, 33(4):279–288, 2017.

[112] Wassim G Najm, John D Smith, Mikio Yanagisawa, et al. Pre-crash scenario typology for crash avoidance research. Technical report, United States. National Highway Traffic Safety Administration, 2007.

[113] Mirko Nentwig, Maximilian Miegler, and Marc Stamminger. Concerning the applicability of computer graphics for the evaluation of image processing algorithms. In *Vehicular Electronics and Safety (ICVES), 2012 IEEE International Conference on*, pages 205–210. IEEE, 2012.

[114] Mirko Nentwig and Marc Stamminger. Hardware-in-the-loop testing of computer vision based driver assistance systems. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 339–344. IEEE, 2011.

[115] Christian Neurohr, Lukas Westhofen, Tabea Henning, Thies de Graaff, Eike Möhlmann, and Eckard Böde. Fundamental considerations around scenario-based testing for automated driving. *arXiv preprint arXiv:2005.04045*, 2020.

[116] Cu D Nguyen, Simon Miles, Anna Perini, Paolo Tonella, Mark Harman, and Michael Luck. Evolutionary testing of autonomous software agents. *Autonomous Agents and Multi-Agent Systems*, 25(2):260–283, 2012.

[117] Julia Nilsson, Mattias Brännström, Erik Coelingh, and Jonas Fredriksson. Longitudinal and lateral control for automated lane change maneuvers. In *American Control Conference (ACC), 2015*, pages 1399–1404. IEEE, 2015.

[118] Julia Nilsson, Mattias Brännström, Erik Coelingh, and Jonas Fredriksson. Lane change maneuvers for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1087–1096, 2017.

[119] Julia Nilsson, Jonatan Silvlin, Mattias Brannstrom, Erik Coelingh, and Jonas Fredriksson. If, when, and how to perform lane change maneuvers on highways. *IEEE Intelligent Transportation Systems Magazine*, 8(4):68–78, 2016.

[120] David Nister, Hon-Leung Lee, Julia Ng, and Yizhou Wang. The safety force field. online at `https://www.nvidia.com/content/dam/en-zz/Solutions/`

self-driving-cars/safety-force-field/the-safety-force-field.pdf, retrieved 5th October 2020.

[121] Philippe Nitsche, Pete Thomas, Rainer Stuetz, and Ruth Welsh. Pre-crash scenarios at road junctions: A clustering method for car crash data. *Accident Analysis & Prevention*, 107:137–151, 2017.

[122] Jan-Pieter Paardekooper, S Montfort, Jeroen Manders, Jorrit Goos, Erwin de Gelder, Olaf Op den Camp, Annie Bracquemond, and Gildas Thiolon. Automatic identification of critical scenarios in a public dataset of 6000 km of public-road driving. In *26th International Technical Conference on the Enhanced Safety of Vehicles (ESV)*. Mira Smart, 2019.

[123] Christian Pek, Vitaliy Rusinov, Stefanie Manzinger, Murat Can Üste, and Matthias Althoff. Commonroad drivability checker: Simplifying the development and validation of motion planning algorithms. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1013–1020. IEEE, 2020.

[124] Christian Pek, Peter Zahn, and Matthias Althoff. Verifying the safety of lane change maneuvers of self-driving vehicles based on formalized traffic rules. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 1477–1483. IEEE, 2017.

[125] Oliver Pink, Jan Becker, and Sören Kammel. Automated driving on public roads: Experiences in real traffic. *it-Information Technology*, 57(4):223–230, 2015.

[126] Alexander Poddey, Tino Brade, Jan Erik Stellet, and Wolfgang Branz. On the validation of complex systems operating in open contexts. *arXiv preprint arXiv:1902.10517*, 2019.

[127] Hartmut Pohlheim, Mirko Conrad, and Arne Griep. Evolutionary safety testing of embedded control software by automatically generating compact test data sequences. Technical report, SAE Technical Paper, 2005.

[128] Alexander Pretschner. Defect-based testing. In: Dependable Software Systems Engineering, 2015.

[129] Andreas Pütz, Adrian Zlocki, Julian Bock, and Lutz Eckstein. System validation of highly automated vehicles with a database of relevant traffic scenarios. *12th ITS European Congress*, 2017.

[130] Andreas Pütz, Adrian Zlocki, Jörg Küfen, Julian Bock, and Lutz 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.

[131] Rodrigo Queiroz, Thorsten Berger, and Krzysztof Czarnecki. Geoscenario: An open dsl for autonomous driving scenario representation. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 287–294. IEEE, 2019.

[132] Stefan Riedmaier, Thomas Ponn, Dieter Ludwig, Bernhard Schick, and Frank Diermeyer. Survey on scenario-based safety assessment of automated vehicles. *IEEE Access*, 8:87456–87477, 2020.

[133] Albert Rizaldi and Matthias Althoff. Formalising traffic rules for accountability of autonomous vehicles. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 1658–1665. IEEE, 2015.

[134] Albert Rizaldi et al. Formalising and monitoring traffic rules for autonomous vehicles in isabelle/hol. In *International Conference on Integrated Formal Methods*, pages 50–66. Springer, 2017.

[135] Albert Rizaldi, Fabian Immler, and Matthias Althoff. A formally verified checker of the safe distance traffic rules for autonomous vehicles. In *NASA Formal Methods Symposium*, pages 175–190. Springer, 2016.

[136] Elias Rocklage, Heiko Kraft, Abdullah Karatas, and Jörg Seewig. Automated scenario generation for regression testing of autonomous vehicles. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 476–483. IEEE, 2017.

[137] Christian Roesener, Felix Fahrenkrog, Axel Uhlig, and Lutz Eckstein. A scenario-based assessment approach for automated driving by using time series classification of human-driving behaviour. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1360–1365. IEEE, 2016.

[138] Christian Roesener, Jan Sauerbier, Adrian Zlocki, Felix Fahrenkrog, Lei Wang, András Várhelyi, Erwin de Gelder, Joris Dufils, Sandra Breunig, Pablo Mejuto, et al. A comprehensive evaluation approach for highly automated driving. In *25th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration*, 2017.

[139] SAE. Definitions for terms related to on-road motor vehicle automated driving systems. *J3016, SAE International Standard*, 2018.

[140] Khaled Saleh, Mohammed Hossny, and Saeid Nahavandi. Kangaroo vehicle collision detection using deep semantic segmentation convolutional neural network. In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7. IEEE, 2016.

[141] Ulrich Sander and Nils Lubbe. The potential of clustering methods to define intersection test scenarios: Assessing real-life performance of aeb. *Accident Analysis & Prevention*, 113:1–11, 2018.

[142] Fabian Schuldt. *Ein Beitrag für den methodischen Test von automatisierten Fahrfunktionen mit Hilfe von virtuellen Umgebungen*. PhD thesis, TU Braunschweig, 2017.

[143] Fabian Schuldt, Falko Saust, Bernd Lichte, Markus Maurer, and S Scholz. Effiziente systematische testgenerierung für fahrerassistenzsysteme in virtuellen umgebungen. *Automatisierungssysteme, Assistenzsysteme und Eingebettete Systeme Für Transportmittel*, 2013.

[144] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv:1708.06374*, retrieved 5th October 2020.

[145] Christoph Sippl et al. From simulation data to test cases for fully automated driving and adas. In *IFIP International Conference on Testing Software and Systems*, pages 191–206. Springer, 2016.

[146] Jaehyun Jason So, Inseon Park, Jeongran Wee, Sangmin Park, and Ilsoo Yun. Generating traffic safety test scenarios for automated vehicles using a big data technique. *KSCE Journal of Civil Engineering*, 23(6):2702–2712, 2019.

[147] Michal Sroka, Roman Nagy, and Dominik Fisch. Genetic algorithms in test design automation. In *Applied Mechanics and Materials*, volume 693, pages 153–158. Trans Tech Publ, 2014.

[148] Markus Steimle, Gerrit Bagschik, Till Menzel, Jan Wendler, and Markus Maurer. Ein Beitrag zur Terminologie für den szenarienbasierten Testansatz automatisierter Fahrfunktionen. In *AAET - Automatisiertes und vernetztes Fahren*, 2018.

[149] Jan Erik Stellet, Marc René Zofka, Jan Schumacher, Thomas Schamm, Frank Niewels, and J Marius Zöllner. Testing of advanced driver assistance towards automated driving: A survey and taxonomy on existing approaches and open questions. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 1455–1462. IEEE, 2015.

[150] Zaid Tahir and Rob Alexander. Coverage based testing for v&v and safety assurance of self-driving autonomous vehicle: A systematic literature review. In *The Second IEEE International Conference On Artificial Intelligence Testing*. York, 2020.

[151] Ömer Şahin Taş, Florian Kuhnt, J Marius Zöllner, and Christoph Stiller. Functional system architectures towards fully automated driving. In *2016 IEEE Intelligent vehicles symposium (IV)*, pages 304–309. IEEE, 2016.

[152] Pavlo Tkachenko, Jinwei Zhou, and Luigi del Re. Unsupervised clustering of highway motion patterns. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2337–2342. IEEE, 2019.

[153] Cumhur Erkan Tuncali and Georgios Fainekos. Rapidly-exploring random trees for testing automated vehicles. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 661–666. IEEE, 2019.

[154] Cumhur Erkan Tuncali, Georgios Fainekos, Danil Prokhorov, Hisahiro Ito, and James Kapinski. Requirements-driven test generation for autonomous vehicles with machine learning components. *IEEE Transactions on Intelligent Vehicles*, 5(2):265–280, 2019.

[155] Cumhur Erkan Tuncali, Theodore P Pavlic, and Georgios Fainekos. Utilizing s-taliro as an automatic test generation framework for autonomous vehicles. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1470–1475. IEEE, 2016.

[156] Simon Ulbrich, Till Menzel, Andreas Reschka, Fabian Schuldt, and Markus Maurer. Defining and substantiating the terms scene, situation, and scenario for automated driving. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 982–988. IEEE, 2015.

[157] U.S. Department of Transportation — National Highway Traffic Safety Administration (NHTSA). Automated driving systems - a vision for safety. online at `https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf`, retrieved 5th October 2020.

[158] U.S. Department of Transportation — National Highway Traffic Safety Administration (NHTSA). A framework for automated driving systems — testable cases and scenarios. online at `https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13882-automateddrivingsystems_092618_v1a_tag.pdf`, retrieved 5th October 2020.

[159] Tanja EJ Vos, Felix F Lindlar, Benjamin Wilmes, Andreas Windisch, Arthur I Baars, Peter M Kruse, Hamilton Gross, and Joachim Wegener. Evolutionary functional black-box testing in an industrial setting. *Software Quality Journal*, 21(2):259–288, 2013.

[160] Walther Wachenfeld, Philipp Junietz, Raphael Wenzel, and Hermann Winner. The worst-time-to-collision metric for situation identification. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 729–734, 2016.

[161] Walther Wachenfeld and Hermann Winner. The release of autonomous vehicles. In *Autonomous Driving*, pages 425–449. Springer, 2016.

[162] Sebastian Wagner, Korbinian Groh, Thomas Kuhbeck, Michael Dorfel, and Alois Knoll. Using time-to-react based on naturalistic traffic object behavior for scenario-based risk assessment of automated driving. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1521–1528. IEEE, 2018.

[163] Sebastian Wagner, Korbinian Groh, Thomas Kühbeck, and Alois Knoll. Towards cross-verification and use of simulation in the assessment of automated driving. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1589–1596. IEEE, 2019.

[164] Wenshuo Wang, Chang Liu, and Ding Zhao. How much data are enough? a statistical approach with case study on longitudinal driving behavior. *IEEE Transactions on Intelligent Vehicles*, 2(2):85–98, 2017.

[165] Wenshuo Wang, Aditya Ramesh, and Ding Zhao. Clustering of driving scenarios using connected vehicle datasets. *arXiv:1807.08415*, 2018.

[166] Waymo LLC. On the road to fully self-driving — waymo safety report. online at `https://www.mtfchallenge.org/wp-content/uploads/2017/02/waymo-safety-report-2017-10.pdf`, retrieved 5th October 2020.

[167] Hendrik Weber, Julian Bock, Jens Klimke, Christian Roesener, Johannes Hiller, Robert Krajewski, Adrian Zlocki, and Lutz Eckstein. A framework for definition of logical scenarios for safety assurance of automated driving. *Traffic injury prevention*, 20(sup1):S65–S70, 2019.

[168] Nico Weber, Dirk Frerichs, and Ulrich Eberle. A simulation-based, statistical approach for the derivation of concrete scenarios for the release of highly automated driving functions. In *AmE 2020-Automotive meets Electronics; 11th GMM-Symposium*, pages 1–6. VDE, 2020.

[169] Bowen Weng, Sughosh J Rao, Eeshan Deosthale, Scott Schnelle, and Frank Barickman. Model predictive instantaneous safety metric for evaluation of automated driving systems. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1899–1906. IEEE, 2020.

[170] Tim A Wheeler, Mykel J Kochenderfer, and Philipp Robbel. Initial scene configurations for highway traffic propagation. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 279–284. IEEE, 2015.

[171] Andreas Windisch and Noura Al Moubayed. Signal generation for search-based testing of continuous systems. In *Software Testing, Verification and Validation Workshops, 2009. ICSTW'09. International Conference on*, pages 121–130. IEEE, 2009.

[172] Hermann Winner, Stephan Hakuli, Felix Lotz, and Christina Singer. *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. Springer-Verlag, 2015.

[173] Qin Xia, Jianli Duan, Feng Gao, Qiuxia Hu, and Yingdong He. Test scenario design for intelligent driving system ensuring coverage and effectiveness. *International Journal of Automotive Technology*, 19(4):751–758, 2018.

[174] Hang Yin and Christian Berger. When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1–8, 2017.

[175] Ding Zhao, Yaohui Guo, and Yunhan Jack Jia. Trafficnet: An open naturalistic driving scenario library. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2017.

[176] Ding Zhao and Huei Peng. From the lab to the street: Solving the challenge of accelerating automated vehicle testing. online at `https://mcity.umich.edu/wp-content/uploads/2017/05/Mcity-White-Paper_Accelerated-AV-Testing.pdf`, retrieved 5th October 2020.

[177] Jinwei Zhou and Luigi del Re. Identification of critical cases of adas safety by fot based parameterization of a catalogue. In *Control Conference (ASCC), 2017 11th Asian*, pages 453–458. IEEE, 2017.

[178] Jinwei Zhou and Luigi del Re. Reduced complexity safety testing for adas & adf. *IFAC-PapersOnLine*, 50(1):5985–5990, 2017.

[179] Marc René Zofka, Florian Kuhnt, Ralf Kohlhaas, Christoph Rist, Thomas Schamm, and J Marius Zöllner. Data-driven simulation and parametrization of traffic scenarios for the development of advanced driver assistance systems. In *Information Fusion (Fusion), 2015 18th International Conference on*, pages 1422–1428. IEEE, 2015.

[180] Xueyi Zou, Rob Alexander, and John McDermid. Testing method for multi-uav conflict resolution using agent-based simulation and multi-objective search. *Journal of Aerospace Information Systems*, 2016.