Applying CT-FLA for AEB Function Testing: A Virtual Driving Case Study

Ludwig Kampel
MATRIS Research Group
SBA Research
Vienna, Austria
lkampel@sba-research.org

Michael Wagner
MATRIS Research Group
SBA Research
Vienna, Austria
mwagner@sba-research.org

Dimitris E. Simos
MATRIS Research Group
SBA Research
Vienna, Austria
dsimos@sba-research.org

Mihai Nica
AVL List GmbH
Graz, Austria
mihai.nica@avl.com

Dino Dodig
AVL List GmbH
Maribor, Slovenia
dino.dodig@avl.com

David Kaufmann
AVL List GmbH
Graz, Austria
david.kaufmann@avl.com

Franz Wotawa
Institute for Software Technology
Graz University of Technology
Graz, Austria
wotawa@ist.tugraz.at

***Abstract*—The advancements of automated and autonomous vehicles requires virtual verification and validation of automated driving functions, in order to provide necessary safety levels and to increase acceptance of such systems. The aim of our work is to investigate the feasibility of combinatorial testing fault localization (CT-FLA) in the domain of virtual driving function testing. We apply CT-FLA to screen parameter settings that lead to critical driving scenarios in a virtual verification and validation framework used for automated driving function testing. Our first results indicate that CT-FLA methods can help to identify parameter-value combinations leading to crash scenarios.**

***Index Terms*—Combinatorial testing, Combinatorial fault localization, AEB, autonomous driving, test scenario generation**

I. Introduction

The advances in applications of artificial intelligence in the field of automated and autonomous driving systems leads to more complex systems and functions. Advanced driver assistant systems and automated driving systems are safety critical, i.e. a failure or unintended system behavior can cause accidents with severe consequences. This leads inevitable to challenges in terms of quality and safety assurance of automated driving functions. In order to ensure that in every possible situation the an advanced driver assistant or AD system decides for a correct and safe behavior, it is necessary to test related systems and functions extensively under different environmental and traffic conditions. However, the high complexity of the systems in use and the intractable number of input traffic scenarios and environmental conditions makes exhaustive testing impossible, or infeasible to say the least. Rendering validation and verification of AD functions an important and challenging problem [1]. The authors of [2] present a summary of the major challenges in autonomous vehicle testing. They argue that it would require hundreds of a million kilometers driven of road testing, in order to statistically show that an automated vehicle is as safe as a humanly driven vehicle. They thereby point out the that ironically, the safer the vehicle driving is, the higher the number of required test kilometers is. Similarly, Karla and Paddock [3] give an estimation of more than 275 million miles required in order to perform such vehicle testing when underlying the fatality rate of driving in the US. This amounts to a testing time of 400 years assuming that there are 100 cars in use, driving continuously at an average speed of 25 miles per hour.

These numbers stress that distance based validation of autonomous vehicles is not an acceptable solution. Since automated vehicles need to be tested in different traffic situations and in interactions with other vehicles, scenario-based approaches are considered as proper methods for the development of automated driving functions [4], [5], [6]. This however raises the question which scenarios to generate, respectively how to automatically generate scenarios in order to test automated driving functions. Thereby the focus lies on generating *critical scenarios*, i.e. scenarios that lead to a crash or nearly a crash involving the vehicle. In existing works, combinatorial testing (CT) was used as one approach to address this question [7], [8].

Contribution. In this paper we report the – to the best of our knowledge – first application of *combinatorial testing fault localization* (CT-FLA) methods for autonomous vehicle testing in an industrial setting, focusing on a detailed case study based on an autonomous emergency braking system from AVL List GmbH (AVL). Particularly we address the problem of evaluating automatically generated scenarios for virtual validation and verification of automated driving functions. We use CT-FLA to screen parameter settings that lead to critical scenarios in a virtual verification validation framework used for automated driving function testing, in the following

A part of the work has been performed in the project ArchitectECA2030. ArchitectECA2030 project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 877539. The JU receives support from the European Union's Horizon 2020 research and innovation programme. It is co-funded by the consortium members and grants from Germany, Netherlands, Czech Republic, Austria and Norway. A part of this work was funded by SBA-K1 (COMET Center) within the framework of COMET - Competence Centers for Excellent Technologies Programme which is funded by BMK, BMDW, and the federal state of Vienna and managed by FFG.

referred to as *virtual driving test platform*. For the domain experts, it is important to find out which parameter settings, and hence which parameters play an essential role in crash scenarios. By means of CT-FLA we are able to identify t-way interactions in the scenario specifications, that (are likely to) result in a crash. The obtained information of (potential) crash inducing t-way interactions can be used in further test scenario generation in order to generate crash and near crash scenarios more targeted. These efforts, must be understood as part of the larger undertaking of verification and validation of automated driving functions.

The automatic emergency braking (AEB) function strongly contributes to the active safety properties of automotive vehicles. Proper functioning of AEB increases vehicle and road safety and further contributes to the acceptance of automated driving in general. Conversely, misbehavior of this function, e.g. by triggering the brake too late or groundlessly, can cause severe road accidents with fatal consequences. It is hence key to test the AEB functionality extensively. Extending previous efforts, our case study focuses on this central and representative function of automated driving.

This paper is structured as follows: Section II reviews related work, Section III describes the general framework conditions and the CT-FLA method we used. Thereafter, in Section IV we give a detailed description of the set-up of our case study, i.e. of the system under test (SUT), the used input-parameter model (IPM), the virtual driving test platform and the testing oracle. Our results are presented and discussed in Section V, before we conclude this paper.

II. Related Work

A. Automated Driving Functions and Combinatorial Testing

The present work can also be seen as an extension of the application of the combinatorial testing cycle (see Fig. 1) to the domain of automated driving function testing where the following works have appeared in the literature. The work in [9] presents an algorithm that converts ontologies to input parameter models (see [10] for more details on IPMs) and further to combinatorial test suits. In [7] the authors discuss main challenges of testing automated and autonomous vehicles and propose to use CT based on automatically extracted IPMs from ontologies as a solution. The main contribution of [7] is the introduction of a general testing methodology for automated and autonomous vehicles, that uses the algorithm of [9] for mapping ontologies of the environment of the vehicle to IPMs for CT. Thereafter, in [8] the developed ontology based CT methods are improved (CT_ONT2) and used in an industrial setting for testing autonomous driving functions. More specifically, the AEB function system from AVL was tested in a concrete case study based using an ontology-based representation of the domain and using CT for test case generation. In [11] the focus was to use a genetic algorithm for test parameter optimization in order to efficiently derive critical driving scenarios for virtual testing. So far these efforts culminate in the study presented in [12], where search based testing, combinatorial testing and random testing have been empirically compared for AEB function testing. The study compares the scenarios generated from these methods, with respect to the quantity and quality of the crashes recorded during test execution. Thereby CT leads to the highest total number of crashes and notably was the only method that lead to all different types of crashes that where considered, this holds for strength $t = 3$ *and* $t = 2$. However, in terms of crash frequency, i.e. number of crashes per test scenario, CT was inferior to search based testing.

The revisited works above all realize or refine the automated testing methodology presented in [6]. Our present work augments this methodology by adding the aspect of combinatorial fault localization to it.

Also other authors have applied CT in the automotive domain. For example Dhadyalla, Kumari and Snell [13] thoroughly present the application of CT to a real hybrid electric vehicle control system. Thereby two electronic control units are modelled in terms of CT and their controller area network messages and replaced with tests coming from 2-way, 3-way and variable strength up to 4-way combinatorial test suites. Lastly we want to mention that also sequential CT in the form of sequence covering arrays have been used in the automotive domain [14].

B. Combinatorial Fault Localization

Combinatorial testing fault localization (CT-FLA) can be subdivided into *adaptive* and *non-adaptive* approaches. Non-adaptive approaches do not depend on repeated test execution. Fault location merely relies on the properties of the applied combinatorial test suite and the execution results in form of a pass/fail assignment to the tests. Combinatorial objects with these desired properties were initially described in [15], and have been generalized in several works, e.g. in [16] or in [17]. These structures however are difficult to find or generate, and the absence of powerful effective generation algorithms, as of this writing, renders them impractical for applications. However, they are not unusable by any means as the work in [18] shows. Adaptive approaches, on the other hand, do rely on an alternation of test execution and test generation, where test results of earlier tests can shape the generation of later tests, e.g. [19] and [20]. In multiple rounds of test execution and test generation, the set of potential failure inducing t-way interactions can be gradually concertized.

While CT in general has been applied successfully in several industrial settings, see e.g. [21], [22] or [23], there are only a few works known on the application of CT-FLA in such settings.

In [19] the BEN tool, adaptive approach to CT-FLA has been proposed, that generates new tests based on statistical ranking of *suspicious* t-way interactions. When BEN has access to the source code of a program, it can output a ranking of the likelihood of statements to be faulty. In [24] an extensive experimental evaluation showed that the proposed adaptive CT-FLA method is applicable to real world software artifacts. Later, in [25] the BEN tool has been used for evolving an input model for security testing for XSS vulnerabilities, where

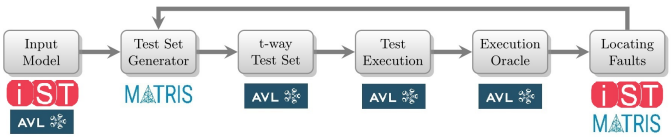


Fig. 1: Visualization of the combinatorial testing cycle, indicating the working areas of the contributing organizations AVL List GmbH (AVL), Institute for Software Technology, Graz University of Technology (IST) and the MATRIS Research Group from SBA Research (MATRIS) in this case study.

the suspicious t-way interactions returned from BEN represent input combinations suspected to enable XSS vulnerabilities. Hence, the applied CT-FLA method is not particularly used for *combinatorial fault localization*, but rather for identifying input combinations that trigger a certain behaviour of the SUT.

In a similar way we can understand the work in [26] and [27], where the concept of using CT-FLA for explaining categorizations in data bases, or decisions of artificial intelligence systems by means of t-way interactions is elucidated and discussed. This line of research culminated so far in the application of BEN to produce explanations for decisions of image classifiers in [28].

The authors of [20] propose an interleaving approach, where a failing test of an initial combinatorial test suite is immediately examined by executing new tests directly after it, aiming to identify the failure inducing t-way interaction causing the original test to fail. The rational behind this strategy is to identified failure-inducing interactions early and guide the remaining test case generation. The proposed method is also extensively evaluated experimentally by applying it to five subject programs. It is worthwhile to mention that such an approach requires a high frequent communication between the SUT and the combinatorial test generation method, especially when there are many failing tests, or failure inducing t-way interactions.

The authors of [29] have integrated CT-FLA into an industrial framework, as part of the commercial tool IBM FOCUS. Roughly summarized, the followed approach consists of replacing each position of a failing test (coming from a CA of strength t) with a *safe-value* that does not appear in any failing test. Under the assumed preconditions the tool always returns a failure inducing t-way interaction. They report on a promising initial experience in applying it to an operating system where the existence of a bug was known due to previous CT. Thanks to their end to end automation, the failure inducing t-way interactions were correctly identified and reported to a test engineer without any manual intervention.

III. Methodology

In the following paragraphs we first describe the interpretation of the considered testing problem as a combinatorial fault localization problem; second we describe the workflow of the testing procedure; and third the CT-FLA method used for test suite generation and analysis.

Automated driving functions are tested on virtual driving environment platforms for their behaviour in different driving scenarios. See for example [30] for a review of simulation-based verification and validation methods. The execution of a scenario can be understood as a simulation of a real world traffic situation from the point of view of a vehicle - referred to as the *ego vehicle* in the following - that is controlled using automated driving functions, such as an AEB function. The aim of our work is to get a better understanding why certain scenarios simulated on the virtual driving platform result in a *crash*, while others do not. To that extent in the testing problem at hand, we can understand the AEB function as the SUT. However its input is not modelled directly, but instead the driving scenario that the virtual vehicle undergoes, or more precisely the scenario specification, is specified by means of an IPM that models the traffic situation.

For example, the speed and the type of the ego vehicle as well as the number of other road users, their position and their speed, is captured by the IPM and potential constraints defined between the parameters. A more detailed description of the used IPM will be given in Section IV. The individual parameter-values are entered automatically to a virtual driving platform so that the simulation can be performed. The individual driving scenarios thus represent a test case, which can be derived for example from the rows of a covering array (CA). In order to identify critical scenarios we rely the time-to-collision (TTC), a well-known time-based safety indicator, since several decades [31]. The TTC is defined for every point in time, as the time span left until two vehicles would collide, if no evasive action would be taken. In short, the lower the TTC value, the more critical the driving situation, see [32] and [33]. The TTC is collected throughout the whole simulation of a scenario and if it gets dangerously small, i.e. falls below a given threshold TTC_{crit}, at any point in time, the scenario is considered to be critical. A TTC equal or close to zero represents a crash. Finally, using the above interpretation, parameter settings that are likely responsible or essential for critical scenarios map to the concept of failure inducing t-way interactions (FITs) from CT, see e.g. [34] but also [19] and [20] where this notion is referred to as *failure causing schemas*, or *suspicious combinations* respectively. Table I summarizes the described interpretation of notions from automated driving function testing in terms of CT. In the following the terms *failure inducing t-way interaction*, abbreviated as FIT, and *crash inducing parameter setting* can be understood synonymous.

The structural and infrastructural circumstances in the documented case study are as follows. In the following we refer to the main phases of a (simplified) combinatorial testing cycle depicted in Fig. 1; and refer the interested reader to [10] for more details on the *combination strategies testing process*. The phases are split between the contributing organizations AVL List GmbH (AVL), Institute of Software Technology at Graz University of Technology (IST) and the MATRIS Research Group from SBA Research (MATRIS) in the following way. AVL operates the virtual driving test platform, and the IPM

TABLE I: Mapping between concepts from virtual driving function testing and those of combinatorial testing.

Virtual driving function testing	Combinatorial testing
AEB function	System under test (SUT)
Driving scenario specification	Input-parameter model (IPM)
Driving scenario	Test vector
Time-to-collision (TTC)	Testing oracle (oracle)
Driving scenario simulation	Test execution
Non-crash scenario (TTC$> TTC_{crit}$)	Passing test
Crash scenario (TTC$\leq TTC_{crit}$)	Failing test
Crash inducing parameter setting	Failure inducing t-way interaction (FIT)

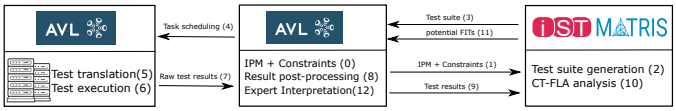


Fig. 2: The structure of the CT workflow in the considered scenario, the numbers indicate the order in which the steps are performed. The schematics shows leftmost the computation unit from AVL, in the middle the development and testing unit of AVL, and rightmost the research groups of IST and MATRIS. The numbers (i) for $i = 0, \ldots, 12$ indicate the order in which the steps are performed.

for the same was also created jointly by AVL and IST. The test suite generation is performed by MATRIS based on the IPM obtained from AVL. The t-way test sets are translated to executable tests using AVL's internally developed tool. Test execution is also performed by AVL, however the computationally costly simulations need to be performed on a powerful computing environment. In our case this is a server from AVL where the test scenarios are executed on the virtual driving test platform. The sever however is used by multiple in-house clients and requires to be scheduled in advance in order to perform computations, reserving a time slot and the required computation time. Once AVL receives the testing results to a test suite, they are post-processed and forwarded to MATRIS where the analysis of the results is performed. The transmission of test suites and the testing results between the two parties is not (yet) automated, as the small number of testing cycles did not demand for it so far. The structure of the workflow of the combinatorial testing cycle in the considered case study is depicted in Fig. 2.

The given structural and infrastructural circumstances entail certain hurdles in our joint undertaking. Particularly the fact that test generation needs to be performed separate from the test execution platform, and the fact that test execution requires booking of computational resources timely in advance, required us to follow an adequate CT-FLA method. Prominent adaptive CT-FLA methods such as the one presented in [20] do rely on a frequent communication between test execution unit and the test generation unit. In order to keep the overhead introduced through test suite scheduling low, the used CT-FLA method should not depend on a frequent communication between the test generation unit and the test execution unit. However, application of a non-adaptive CT-FLA method is also not possible, since such are (currently) only applicable

Algorithm 1 CT-FLA Approach

Require: initial test strength t_0, maximal test strength t_{max} or testing budget
1: model SUT via IPM and constraints
2: generate initial test suite $\mathcal{T}_t$ of strength $t \leftarrow t_0$
3: **while** $t \leq t_{max}$ and testing budget not exceeded **do**
4: execute tests in $\mathcal{T}_t \rightsquigarrow$ oracle results o_t
5: $potentialFITs \leftarrow$ all i-way interactions for $i \leq t$ that *do not*
6: appear in any passing tests
7: $\mathcal{T}_{t+1} \leftarrow TestSuiteGeneration(t, potentialFITs_t)$
8: $t \leftarrow t + 1$
9: **end while**
10: **return** $potentialFITs$

TestSuiteGeneration$(t, potentialFITs_t)$
11: $\mathcal{T}_{t+1} = \emptyset$
12: **for all** t-way interactions $\tau \in potentialFITs_t$ **do**
13: construct separating test r_τ for τ, containing minimized number of other elements of $potentialFITs_t$
14: add test r_τ to $\mathcal{T}_{t+1}$
15: **end for**
16: extend $\mathcal{T}_{t+1}$ to a CA of strength $t + 1$
17: **return** $\mathcal{T}_{t+1}$

and available for a small number of FITs with a small strength, see e.g. [35] or [36] and references therein for descriptions of non-adaptive CT-FLA methods.

Hence we opted to realize an adaptive CT-FLA approach (Algorithm 1), that allows to have a low frequent communication between test execution and generation unit, which works as follows. The initial test suite is derived from a CA of strength t for the given IPM and respecting the constraints. The tests are executed and the execution oracle, which is realized through measuring the minimal occurring TTC in a scenario, categorizes them in failing tests (respectively crash scenarios) or passing tests (respectively non-crash scenarios). Based on the testing results we analyze the test suite and identify all t-way interactions that appear in at least one passing test. All remaining t-way interactions constitute the potentially failure inducing t-way interactions (potential FITs), also known as *suspicious combinations* as they are referred to in [19]. Based on this, the test suite for the next round is computed, which is derived from a CA of strength $t+1$. Thereby we aim to enrich the generated CA with the additional *separation property*, that each potential FIT appears in a test case that covers no other potential FIT. In some cases this might not be possible for all potential FITs, for example due to given constraints in the IPM, or simply due to the sheer number of potential FITs (as is the case this case study). Therefore for each potential FIT we generate a test that *minimizes* the number other FITs covered by the test. Algorithm 1 gives a high level description of the realized CT-FLA approach.

IV. Case Study

Following we give a description of the testing infrastructure, i.e. the virtual driving test platform, the considered AEB function, and the translation of test cases to executable scenarios. We further describe the IPM with constraints used to model the input to the virtual driving test platform as well as the considered testing oracles and the test execution environment.

a) System Under Test: In our case study we consider an AEB test procedure as defined by the European New Car Assessment Programme [37], also known as Euro NCAP, a well-established vehicle safety assessment organization.

Euro NCAP introduced a test protocol for AEB test scenario classes based on frequent accident situations. The set-up for these scenario classes is provided in detail, together with relevant parameters and value ranges, which allow to define concrete test scenarios that are executable on test site or in simulation.

In order to translate parameter-value assignments to executable test scenarios, we first convert it into an XML format according to the standardized OpenScenario specification [38], an open file format for the description of dynamic contents for vehicle driving simulations. The scenario descriptions are finally input to the automated test execution and evaluation framework. Following, we briefly describe the setup of the framework and the comprised models enabling to run and evaluate the scenario simulations. As the main environment we use AVL's Model.Connect [39] co-simulation platform in order to integrate and connect required models. For executing the given test scenarios, the virtual driving environment Virtual Test Drive from Hexagon AB [40] is used. In addition the AVL VSM tool to represent realistic vehicle dynamics, an AEB function, and a control model to select the target object during scenario execution are implemented. These models and tools are linked in the platform, to enable the exchange of information during run-time. Further a crash reporting tool is connected to automatically evaluate the results related to the executed scenario criticality.

The AEB function is designed to automatically perform a braking manoeuvre upon detection of a potential collision. The investigated prototypical AEB function was developed in the course of internal research activities from AVL and is solely used for demonstration purpose. The system comprises two main components: the vision system and the brake control system. The first is based on a radar sensor model and the second comprises a target object selection algorithm and the AEB function. As soon as an object (e.g. a leading vehicle, or a pedestrian walking orthogonal to the traffic lane) enters the radar's cone-shaped field of view (see Fig. 3), the sensor (radar) will detect the object and identify its position and its relative speed compared to the ego vehicle. The radar in this setup is an ideal sensor, no influence of outside noise or other physical phenomena will have any influence on detection. The radar will always detect an object within its cone-shaped detection area. The information from the radar is fed into the target object selection algorithm, which selects the target object based on criticality of its influence on the ego vehicle. The target object information is then forwarded to the AEB function where the minimum required braking time towards that object is calculated, and a velocity-dependent time safety margin is added. When the calculated TTC is equal to or goes below the bound $TTC_{critical}$, a braking maneuver is initiated.

b) IPM: The IPM modelling the input to the virtual driving test platform was previously derived in [7] using methods presented in [9], and was already used in previous works [11] and [8]. We refer the interested reader to [7] for the details on the IPM derivation. The IPM consists of 39 parameters, with the following domain sizes

$$(3, 3, 31, 3, 5, 1, 6, 4, 12, 12, 12, 10, 14, 12, 12, 12, 10, 14, \quad (1)$$
$$31, 4, 3, 20, 9, 3, 3, 31, 4, 3, 20, 9, 3, 3, 31, 4, 3, 20, 9, 3, 3).$$

When sorting these in descending order, we obtain $(31^4, 20^3, 14^2, 12^6, 10^2, 9^3, 6, 5, 4^4, 3^{12}, 1)$, where we use an exponent notation, i.e. x^y means that there are y parameters that can take x values. Additionally there are 42 constraints formulated on the parameters of the IPM, which are expressed as quantifier-free Boolean linear arithmetic formulas. Each constraint uses between two and six parameters of the IPM. When deriving the set of minimal forbidden s-way interactions, that are implicitly forbidden by these constraints, we obtain a list of 1225 s-way interactions of lengths $1 \leq s \leq 4$.

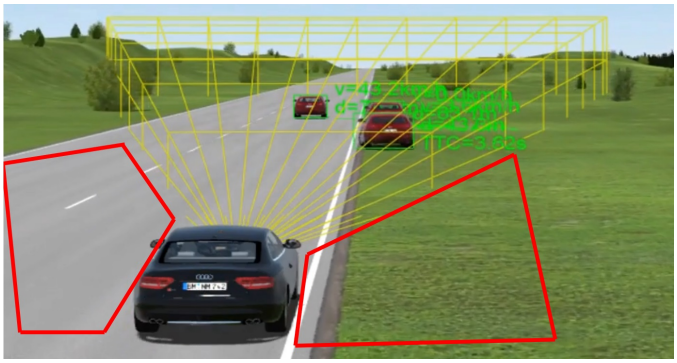


Fig. 3: Excerpt from the visualization of a driving scenario, visualizing the Ego vehicle's cone-shaped radar field of view in yellow, and the blind area in the red.

c) Oracle: As mentioned in Section III the testing oracle is realized through a method that distinguishes critical from non-critical scenarios. This method is based on the time-to-collision measured throughout a scenario. In this case study we are interested in identifying the most critical scenarios, i.e. collisions, which represent AEB failure. This is realized by the oracle function CRS (for CRash Scenario), which assesses any scenario with a TTC equal to or almost zero as a *crash*.

d) Execution environment: The test execution in form of driving scenario co-simulation is using two locally connected machines. The first is an Intel Xeon Silver 4214 with a CPU clocked at 2.2GHz and an NVIDIA Quadro RTX 4000 graphics card (running Windows 10 as operating system), which runs the co-simulation platform Model.CONNECT, simulates the vehicle dynamic using AVL VSM, and the AEB function, as well as performing pre-processing of the test cases. The second machine has an Intel Core i7-9700K CPU clocked at 3.6GHz and an NVIDIA Geforce RTX 2070 SUPER graphics card (running Linux Ubuntu 18.04 as operating system) and is used for the driving environment and sensor simulation using Virtual Test Drive by Hexagon AB [40].

TABLE II: Overview of the results after two rounds of testing of the virtual driving test platform following Algorithm 1.

Test suite	Strength t	# Scenarios # Tests	CRS oracle	potential 2-FITs	potential 3-FITs
$\mathcal{T}_2$	2	994	178	5680	395804
$\mathcal{T}_3$	3	39061	7928	0	125789

TABLE III: The first row shows the quantities of potential 3-way FITs, appearing in the number of failing tests, i.e. crash scenarios, given in the second row.

# pot. 3-FITs	1	1	1	4	8	19	117	387	1634	6500	25872	91245
# failing tests they appear	60	53	46	9	8	7	6	5	4	3	2	1

In this execution environment, the simulation of a single scenario took on average 38 seconds including pre-processing. The exact execution time depends on the scenario outcome, e.g. if the ego vehicle brakes and stops, the scenario simulation is considered completed and is terminated, leading to a shorter execution time.

e) Combinatorial Test Suites: Following the CT-FLA approach given in Algorithm 1, the initial test strength was set to be two, $t_0 = 2$. There was no upper bound for the strength specified, i.e. $t_{max} = 39$ as there are 39 parameters, for the reason that the limiting factor was the testing budget, given by the available time on the test execution unit, which main capacities are required for other projects. The initial test suite $\mathcal{T}_2$ was based on a CA of strength two respecting the given constraints, having 994 rows, i.e. yielding that many driving scenarios. The test suite $\mathcal{T}_3$ for the second iteration of the work-flow was based on a CA of strength three, optimizing separation of all potential 2-way FITs derived from the first iteration; it consists of 39061 rows. There where not enough resources, time and computing capacity, in order to perform a third round of testing.

V. Results

Following we detail the results of the simulation of the driving scenarios performed in the course of this case study, reporting the number of potential FITs identified in the individual testing rounds. Thereafter we evaluate and discuss these results, where we also quantify over the appearance of parameters involved in these FITs. We further give some impressions from the visualization of the potential FITs that are most likely responsible (from a CT-FLA point of view) for crashes in the scenarios they appear in, and report on their assessment through a domain expert.

An overview of the simulation results is given in Table II. The simulation of the 994 scenarios derived from the strength $t = 2$ test suite $\mathcal{T}_2$, resulted in 178 crash scenarios, respectively failing tests. In the parameter specifications of these scenarios we identified 5680 potential 2-way FITs, respectively potential 2-way crash inducing interactions. For the sake of completeness we also report that based on this test suite we identified 395804 potential 3-way FITs, respectively 3-way crash inducing interactions. However, that this number has little to no informative value becomes clear when considering, that the test suite $\mathcal{T}_2$ misses to cover 5659100 different 3-way interactions that respect the given constraints.

According to Algorithm 1 we generated the test suite $\mathcal{T}_3$, by generating tests aiming for separation of the 5680 potential 2-way FITs, i.e. covering each if them in an individual test case. This resulted in 192 tests, where 5417 of the potential 2-way FITs appeared with no other potential 2-way FIT in a test, and the remaining 263 where covered in 64 additional tests. We note again, that a full separation was not possible due to the high number of failing tests, almost 18% of tests, the high number of potential 2-way FITs and the given constraints.

These tests for separating the potential 2-FITs where then extended to a CA of strength $t = 3$, which in sum constitute the test suite $\mathcal{T}_3$. From the 39061 scenarios 7928 resulted in a crash, which are more than 20%. Notably, based on these results it was possible to rule out that any 2-way interaction is crash inducing, as each appears in at least one of the passing tests of $\mathcal{T}_2 \cup \mathcal{T}_3$. However, a very high number 125789 of potential 3-way FITs remains. Although we drastically reduced the number of potential 3-way FITs compared the first round of testing, this number remains too high, rendering another test set of separating tests not executable due to test budget constraints. Recall that the execution consumes about 38 seconds, thus a test suite with over 100000 tests would roughly take one and a half month to execute. Even more so, testing with a test suite based on a CA of strength $t = 4$ is not possible, as such a test suite would contain roughly one million of scenarios – recall that the four largest parameter domains contain 31 values, which causes a strength 4 CA to have at least $31^4 = 923521$ rows.

Given the time constraints, it was also not possible to perform another round of testing. This is why we focused on a more detailed analysis of the potential 3-way interactions obtained so far.

A. Further Analysis Based on Potential 3-way FITs

Investigating the results obtained from the the second round of testing closer, we can make the following qualitative differentiation. For the 125789 potentially 3-way FITs, we count for each in how many failing test it appears. This leads to the distribution given in Table III. These numbers show that the largest fraction of potential 3-way FITs appear only in a small number of failing tests, while on the other hand there are three potential 3-way FITs that are outstanding from all the others, appearing in 60, 53, and 46 failing tests respectively. This difference by itself suggests that these potentially 3-way FITs are of special interest. For us this is reason enough to investigate them even closer, documented in the following subsection.

Aside from the frequencies of appearances of potential 3-way FITs in failing tests, we also considered the frequencies of parameters appearing in potential 3-way FITs, i.e. for each parameter we count how often it appears (with any value) in the set of potential 3-way FITs. Referring to the IPM domain

sizes given in equation (2), the third parameter has the highest number of appearances, contributing to 62557 potential 3-way FITs. Dividing by this maximal value, we get the following vector of frequencies normalized to 62557:

$$\begin{aligned}(&0.02, 0.01, \mathbf{1.00}, 0.00, 0.00, 0.00, 0.00, 0.01, 0.08,\\ &0.08, 0.07, 0.03, 0.12, 0.10, 0.10, 0.10, 0.05, 0.13, \mathbf{0.99},\\ &0.00, 0.00, \textit{0.33}, 0.01, 0.00, 0.00, \mathbf{0.99}, 0.00, 0.00, \textit{0.36},\\ &0.02, 0.00, 0.00, \mathbf{1.00}, 0.00, 0.00, \textit{0.40}, 0.02, 0.00, 0.00)\end{aligned} \tag{2}$$

We see that the parameters with large domain sizes coincide with the parameters that have the highest frequency of appearances in potential 3-way FITs, where it is worth to mention that these frequencies ≥ 0.99 (displayed **bold** in (3)) are roughly three times larger than the frequencies of a second group of parameters appearing ≥ 0.33 (displayed *italics* in (3)), and larger by almost one or multiple orders of magnitude compared to the other frequencies. A threat to validity for this presented measure is that the applied test suites are CAs, i.e. all 2-way and 3-way interactions are covered, hence it may penalize some parameters based on their large domain size. Assume there was a 2-way FIT between two Boolean parameters, in a CA of strength $t = 3$ a parameter with v different values appears in at least v different failing tests; for a v very large compared to the other domain sizes, this can cause a skew distribution. Therefore we also consider for each parameter how often it appears in the set of potential 3-way FITs relative to its domain size. Again the third parameter achieves the highest measure, where each of the 31 values appears on average $2017.97 = 62557/31$ in potential 3-way FITs. Dividing by this maximal value, we get the following vector of frequencies per parameter value normalized to 2017.97 appearances (again referring to the IPM domain sizes given in equation (2)):

$$\begin{aligned}(&0.18, 0.06, \mathbf{1.0}, 0.00, 0.01, 0.00, 0.00, 0.08, 0.20,\\ &0.21, 0.19, 0.10, 0.28, 0.26, 0.27, 0.26, 0.14, 0.29, \mathbf{0.99},\\ &0.00, 0.00, \textit{0.52}, 0.04, 0.00, 0.00, \mathbf{0.99}, 0.00, 0.00, \textit{0.56},\\ &0.05, 0.00, 0.00, \mathbf{1.00}, 0.02, 0.04, \textit{0.61}, 0.07, 0.00, 0.04)\end{aligned} \tag{3}$$

The frequencies per parameter given in equation (4) exhibit a very similar pattern to the one in equation (3). In fact it is the same set of parameters that achieve a frequency ≥ 0.99. When we look at the input-values that are modelled by these parameters, we find that all four of them model the start speed of road users:

```
EgoVehicleStartSpeed
Object1StartSpeed
Object2StartSpeed
Object3StartSpeed
```
(4)

a result that seems reasonable when considering that our investigations pertain traffic and crash scenarios.

1) Comments on IPM Reduction: Finally we also counted how often each parameter-value appears in the potential 3-way FITs. Clearly, the previously identified parameters that appear with a high frequency in the potential 3-way FITs, also have values with a high presence in those. But more interestingly is the insight that there are also parameter-values that do not appear in any of the 125789 potential 3-way FITs. We therefore refer to them as 3-*way safe values*. The 3-way safe values contain potential for reducing the IPM for future scenario generation, by removing any value from it that does not appear in any of the potential 3-way FITs. Such an IPM reduction can allow to derive critical scenarios more efficiently in future testing cycles – A potentially valuable input for practitioners.

Generalizing, the above concept gives rise to the notion of t-*way safe values*, that are parameter-values that do not appear in any failing test of a given t-way test set. We plan to explore this notion in its general form more in the future.

B. Review of a Domain Expert

In order to get a better understanding of the results obtained from Algorithm 1, and to get an insight of the meaning of CT-FLA for automated driving function testing, we asked a domain expert from the field of autonomous driving testing for an opinion on our results. Since it is infeasible to give feedback on each individual of the 125789 potential 3-way FITs, our domain expert considered the three potential 3-way FITs that most likely cause crash scenarios, i.e. the ones that appear in 60, 53 respectively 46 failing tests, see Table III. Listing these in the given order, in the format $(parameter_a = value_a, parameter_b = value_b, parameter_c = value_c)$ they reveal as:

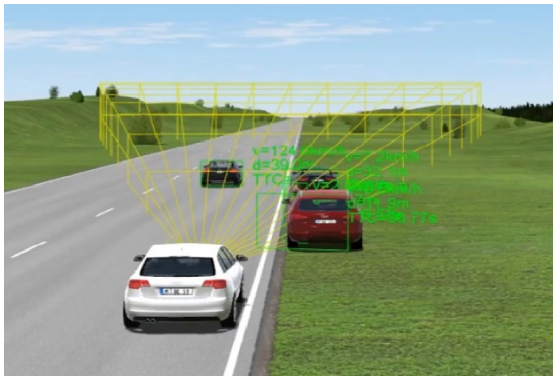
$(EgoVehicleOffset = 2, Pedestrain1StartSpeed = 0.56, Pedestrain1Offset = 4)$,
$(EgoVehicleOffset = 2, Pedestrain1StartSpeed = 0.28, Pedestrain1Offset = 4)$,
$(EgoVehicleOffset = 2, Pedestrain1StartSpeed = 0.56, Pedestrain1Offset = 3)$.

First, we note that interestingly the parameters appearing in these potential 3-way FITs are different from the four parameters given in equation (5) that appear with highest frequencies in the potential 3-way FITs. *This fact gives a valuable first indication for the meaningfulness of CT-FLA in the domain of virtual validation and verification of automated driving functions.*

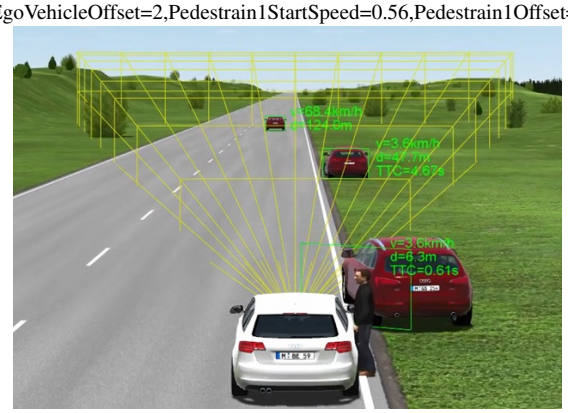
For each of these 3-way interactions, we arbitrarily selected five (failing) tests where they appear in. Recall, since they are potential FITs, all tests that cover the 3-way interactions must be failing. The corresponding driving scenarios, 15 in total, have been simulated and evaluated visually by the domain expert. In Figure 4 we show excerpts from one of the five scenarios for each of the three 3-way interactions.

The assessment of the domain expert after evaluating the 15 scenarios can be summarized as follows. All scenarios provided, covering the identified 3-way interactions have the same root cause for the crash: The vehicle is unable to detect an object, or it loses an object during operation, thus not reacting on time or not even reacting to an imminent collision. Further, a large majority of crashes in the scenarios are sidewards collisions with the ego vehicle.

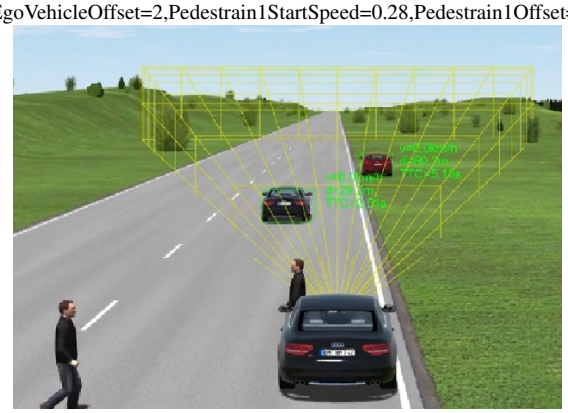
This lead the domain expert to give the following explanation of these outcomes: Since an ideal object sensor is used



(EgoVehicleOffset=2,Pedestrain1StartSpeed=0.56,Pedestrain1Offset=4)



(EgoVehicleOffset=2,Pedestrain1StartSpeed=0.28,Pedestrain1Offset=4)



(EgoVehicleOffset=2,Pedestrain1StartSpeed=0.56,Pedestrain1Offset=3)

Fig. 4: Excerpts from driving scenarios derived from tests that cover the three potential 3-way FITs that appear in the highest number of failing tests, i.e. crash scenarios.

in the simulations, there are no detection issues if an object is within the radar field of view (see the yellow cone-shape in Fig. 3), while objects outside this area, the red area in Fig. 3, will not be detected. Scenario specifications, i.e. parameter values that impact the position (offset) of the ego vehicle and relevant objects to the side of the ego vehicle outside of the cone-shaped detection area will contribute more towards outcomes with a collision.

VI. Threats to Validity

There are some clear threats to validity regarding the results of this case study. First and foremost, it is unfortunate, that we could not perform further rounds of combinatorial testing. To elaborate, it is not clear in how far identified (potential) FITs are responsible for crash scenarios in the virtual driving test platform. In our analysis of the testing results obtained so far we therefore had to rely on relative frequencies of potential FITs, instead of *precisely* identified FITs in terms of CT-FLA. Further, we applied our CT-FLA approach in a specific testing set-up - the virtual driving test platform at AVL - which is based on the framework AVL Model.Connect [39] as co-simulation platform which assembles the three parts (1) AVL VSM for vehicle dynamic simulation, (2) VTD from VIRES [40] as a virtual driving environment and (3) the AEB function developed by AVL for internal research activities. Albeit, this is an important first step in applying CT-FLA methods for virtual testing of automated driving functions, this certainly does not allow for generalization to arbitrary virtual driving test platforms and functions.

VII. Conclusion

In this case study we have demonstrated that CT-FLA methods are applicable to screen parameter settings to identify (potential) crash inducing value combinations in a virtual driving function test platform at AVL List GmbH. In our study we were able to drastically reduce the number of potential crash inducing t-way interactions: our results show that there are no such 2-way interactions, and we reduced the potential crash inducing 3-way interactions by two thirds. Further, we have identified a set of 3-way safe values, i.e. parameter-values that certainly do not appear in any crash inducing 3-way interaction, which bears the potential for an IPM reduction. While it was not possible to identify t-way interactions that certainly lead to a crash, we were able to identify three very likely crash inducing 3-way interactions, which can serve as valuable input to test engineers. A first review by a domain expert indicates that the three identified 3-way interactions lead to scenarios that result in similar types of crashes.

VIII. Outlook

The application of other CT-FLA methods, e.g. ones known in the literature [19], [20], to the same SUT will be of interest in order to compare the obtained results. However, in order to do so it will be required to enhance the test execution setup to allow for a better communication between the test execution unit and the test generation unit, see also the discussion in Section III. This however will require changes in the infrastructure underlying the setup and is therefore kept for onward future work. As part of our immediate future work we will consider multiple oracle functions that differentiate between different types of crashes. Further, we want to execute more tests coming from CT. In particular, as testing with a CA of strength $t = 4$ is infeasible due to test suite size and execution time, we plan to follow a different approach to further reduce the number of potential 3-way FITs, or to increase certainty that specific 3-way interactions lead to crash scenarios. Finally, it is possible to consider also other automated driving functions in the experimental setup.

References

[1] W. Wachenfeld and H. Winner, *The Release of Autonomous Vehicles*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 425–449.

[2] 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. [Online]. Available: http://www.jstor.org/stable/26167741

[3] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965856416302129

[4] F. Schuldt, A. Reschka, and M. Maurer, *A Method for an Efficient, Systematic Test Case Generation for Advanced Driver Assistance Systems in Virtual Environments*. Cham: Springer International Publishing, 2018, pp. 147–175.

[5] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1821–1827.

[6] F. Wotawa, *Testing Autonomous and Highly Configurable Systems: Challenges and Feasible Solutions*. Cham: Springer International Publishing, 2017, pp. 519–532.

[7] F. Klück, Y. Li, M. Nica, J. Tao, and F. Wotawa, "Using ontologies for test suites generation for automated and autonomous driving functions," in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2018, pp. 118–123.

[8] J. Tao, Y. Li, F. Wotawa, H. Felbinger, and M. Nica, "On the industrial application of combinatorial testing for autonomous driving functions," in *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2019, pp. 234–240.

[9] F. Wotawa and Y. Li, "From ontologies to input models for combinatorial testing," in *Testing Software and Systems*, I. Medina-Bulo, M. G. Merayo, and R. Hierons, Eds. Cham: Springer International Publishing, 2018, pp. 155–170.

[10] M. Grindal and J. Offutt, "Input parameter modeling for combination strategies," in *Proceedings of the 25th Conference on IASTED International Multi-Conference: Software Engineering*, ser. SE'07. Anaheim, CA, USA: ACTA Press, 2007, pp. 255–260.

[11] F. Klück, M. Zimmermann, F. Wotawa, and M. Nica, "Genetic algorithm-based test parameter optimization for adas system testing," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, 2019, pp. 418–425.

[12] F. Klück, Y. Li, J. Tao, and F. Wotawa, "An empirical comparison of combinatorial testing and search-based testing in the context of automated and autonomous driving systems," 2023, under review.

[13] G. Dhadyalla, N. Kumari, and T. Snell, "Combinatorial testing for an automotive hybrid electric vehicle control system: A case study," in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, 2014, pp. 51–57.

[14] G. Dhadyalla, C. P. Yang, J. Marco, and P. Jennings, "Real-time sequence testing of an automotive electric machine control systems," SAE Technical Paper 2018-01-0004, Tech. Rep., 2018.

[15] C. J. Colbourn and D. W. McClary, "Locating and detecting arrays for interaction faults," *Journal of Combinatorial Optimization*, vol. 15, no. 1, pp. 17–48, Jan 2008.

[16] C. Martínez, L. Moura, D. Panario, and B. Stevens, "Locating errors using ELAs, covering arrays, and adaptive testing algorithms," *SIAM Journal on Discrete Mathematics*, vol. 23, no. 4, pp. 1776–1799, 2009.

[17] H. Jin, C. Shi, and T. Tsuchiya, "Constrained detecting arrays for fault localization in combinatorial testing," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, ser. SAC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1971–1978.

[18] R. Compton, M. T. Mehari, C. J. Colbourn, E. De Poorter, and V. R. Syrotiuk, "Screening interacting factors in a wireless network testbed using locating arrays," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2016, pp. 650–655.

[19] L. S. Ghandehari, J. Chandrasekaran, Y. Lei, R. Kacker, and D. R. Kuhn, "BEN: A combinatorial testing-based fault localization tool," in *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, April 2015, pp. 1–4.
interaction identification," *IEEE Transactions on Software Engineering*, vol. 46, no. 6, pp. 584–615, 2020.

[20] X. Niu, C. Nie, H. Leung, Y. Lei, X. Wang, J. Xu, and Y. Wang, "An interleaving approach to combinatorial testing and failure-inducing

[21] J. Petke, M. B. Cohen, M. Harman, and S. Yoo, "Practical combinatorial interaction testing: Empirical findings on efficiency and early fault detection," *IEEE Transactions on Software Engineering*, vol. 41, no. 9, pp. 901–924, 2015.

[22] L. Hu, W. E. Wong, D. R. Kuhn, and R. N. Kacker, "How does combinatorial testing perform in the real world: an empirical study," *Empirical Software Engineering*, vol. 25, no. 4, pp. 2661–2693, Jul 2020.

[23] D. E. Simos, L. Kampel, and M. Ozcan, "Combinatorial methods for testing communication protocols in smart cities," in *Learning and Intelligent Optimization*, R. Battiti, M. Brunato, I. Kotsireas, and P. M. Pardalos, Eds. Cham: Springer International Publishing, 2019, pp. 437–440.

[24] L. Sh. Ghandehari, Y. Lei, R. Kacker, R. Kuhn, T. Xie, and D. Kung, "A combinatorial testing-based approach to fault localization," *IEEE Transactions on Software Engineering*, vol. 46, no. 6, pp. 616–645, 2020.

[25] B. Garn, M. Radavelli, A. Gargantini, M. Leithner, and D. E. Simos, "A fault-driven combinatorial process for model evolution in xss vulnerability detection," in *Advances and Trends in Artificial Intelligence. From Theory to Practice*, F. Wotawa, G. Friedrich, I. Pill, R. Koitz-Hristov, and M. Ali, Eds. Cham: Springer International Publishing, 2019, pp. 207–215.

[26] D. R. Kuhn, R. N. Kacker, Y. Lei, and D. E. Simos, "Combinatorial methods for explainable AI," in *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2020, pp. 167–170.

[27] L. Kampel, D. E. Simos, D. R. Kuhn, and R. N. Kacker, "An exploration of combinatorial testing-based approaches to fault localization for explainable ai," *Annals of Mathematics and Artificial Intelligence*, vol. 90, no. 7, pp. 951–964, 2022. [Online]. Available: https://doi.org/10.1007/s10472-021-09772-0

[28] J. Chandrasekaran, Y. Lei, R. Kacker, and D. Richard Kuhn, "A combinatorial approach to explaining image classifiers," in *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2021, pp. 35–43.

[29] D. Blue, A. Hicks, R. Rawlins, and R. Tzoref-Brill, "Practical fault localization with combinatorial test design," in *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2019, pp. 268–271.

[30] F. Wotawa, B. Peischl, F. Klück, and M. Nica, "Quality assurance methodologies for automated driving," *e & i Elektrotechnik und Informationstechnik*, vol. 135, no. 4, pp. 322–327, 2018.

[31] J. C. Hayward, "Near miss determination through use of a scale of danger," *Highway Research Record*, 1972.

[32] K. Vogel, "A comparison of headway and time to collision as safety indicators," *Accident Analysis and Prevention*, vol. 35, no. 3, pp. 427–433, 2003.

[33] A. Svensson, *A method for analysing the traffic process in a safety perspective, Doctoral Dissertation*. Lund, Sweden: University of Lund, 1998.

[34] C. Nie and H. Leung, "The minimal failure-causing schema of combinatorial testing," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 4, Sep. 2011. [Online]. Available: https://doi.org/10.1145/2000799.2000801

[35] H. Jin, C. Shi, and T. Tsuchiya, "Constrained detecting arrays: Mathematical structures for fault identification in combinatorial interaction testing," *Information and Software Technology*, vol. 153, p. 107045, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584922001574

[36] C. J. Colbourn and V. R. Syrotiuk, *There Must be Fifty Ways to Miss a Cover*. 1st ed., Florida, USA, Chapman and Hall/CRC, 2019, pp. 319–333.

[37] European New Car Assessment Programme (EuroNCAP). Test Protocol - AEB Car-to-Car systems, Version 3.0.3. Accessed on: Jan. 11, 2023. [Online]. Available: https://cdn.euroncap.com/media/62794/euro-ncap-aeb-c2c-test-protocol-v303.pdf

[38] AVL List GmbH. ASAM OpenSCENARIO. Accessed on: Jan. 11, 2023. [Online]. Available: http://www.openscenario.org/

[39] AVL List GmbH. Model.CONNECT. Accessed on: Jan. 11, 2023. [Online]. Available: https://www.avl.com/-/model-connect-

[40] Hexagon. Virtual Test Drive. Accessed on: Jan.24, 2023. [Online]. Available: https://hexagon.com/de/products/virtual-test-drive