Analysing Experimental Results Obtained when Applying Search-based Testing to Verify Automated Driving Functions

Florian Klück, Franz Wotawa
Christian Doppler Laboratory for Quality Assurance Methodologies for Autonomous Cyber-Physical Systems
Institute for Software Technology, Graz University of Technology
Graz, Austria
{fklueck, wotawa}@ist.tugraz.at

Gerhard Neubauer, Jianbo Tao, Mihai Nica
AVL List GmbH
Graz, Austria
{gerhard.neubauer, jianbo.tao, mihai.nica}@avl.com

2021 8th International Conference on Dependable Systems and Their Applications (DSA) | 978-1-6654-4391-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/DSA52907.2021.00033

***Abstract*—Assuring safety in case of automated and autonomous driving is of uttermost importance requiring exhaustive search for critical scenarios allowing to reveal faults in current implementations. Different approaches like search-based testing have been already used to come up with test cases that allow to detect situations where the automated or autonomous driving function reacts in an unwanted way. In this paper, we contribute to the corresponding research and provide an in-depth analysis of results obtained using search-based testing applied to two different automatic emergency braking systems. We primarily focus on answering the question regarding the number of parameters required to find crashes. An answer to this question has implications for practice as well as for other testing techniques like combinatorial testing, where there is a need to identify the combinatorial strength in advance for assuring the detection of faults. Our analysis revealed important parameters of tests and we observed that interactions of at least 4 parameters are required to obtain critical scenarios of a high probability.**

Keywords: search-based testing, test automation, ADAS testing, V&V of ADAS

I. Introduction

Safety-critical systems, i.e., systems that may harm people when failing, like cars or airplanes, require exhaustive verification in order to prevent from harm. In case of automated and autonomous driving functions, we have to assure not only that the corresponding implementation is safe in case of a fault occurring inside of the system, but also that interactions with the environment do not lead to situations where crashes or other unwanted behavior happens. Therefore, it is of uttermost importance to come up with testing and other verification methodologies providing guarantees to identify such critical scenarios. In previous work, researchers have reported on the use of combinatorial testing [1] and search-based testing [2] for this purpose.

Automated and autonomous driving functions may rely on techniques and methods originating from Artificial Intelligence (AI) like machine learning. For systems based on AI despite safety other properties have to be assured, e.g., that ethical or moral rules and related regulations are fulfilled. However, we focus solely on the safety properties in this work. For a more general discussion on this topic we refer the interested reader to [3].

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

In this paper, we elaborate on the use of search-based testing for obtaining test cases revealing faults in automated driving functions. In this context, we focus on system testing, considering the implementation of the driving function required sensors and the whole vehicle. The reason behind is to assure that testing takes care of interactions between different parts of the implementation, the sensors, and as well as external objects like other cars or pedestrians crossing the street. A test case provides a scenario for the function, i.e., preconditions like the initial position and speed of the car we want to test, and a sequence of interactions with its environment. A test case (or scenario) is said to be critical if it causes a crash or is close to it.

In contrast to previous work, we do not focus on the underlying testing methodology but on the analysis of experimental results obtained when using search-based testing for verifying two different automatic emergency braking system (AEB) implementations. The objective behind the analysis is to obtain information regarding the influence on the interaction of parameters or actions required to cause a crash. In particular, we are interested in the number of parameters needed because this impacts other approaches like combinatorial testing as well. For the latter, we need information regarding the maximum number of parameters required to interact for obtaining a crash to assure that we have considered enough tests.

Furthermore, we are interested in gaining further insights regarding the obtained test cases. For this purpose, we outline the use of statistics as well as the use of decision trees for analysing the test suite we obtained using search-based testing. Such an analysis can serve as blueprint for other test suites as well.

We organize this paper as follows: First, we discuss the

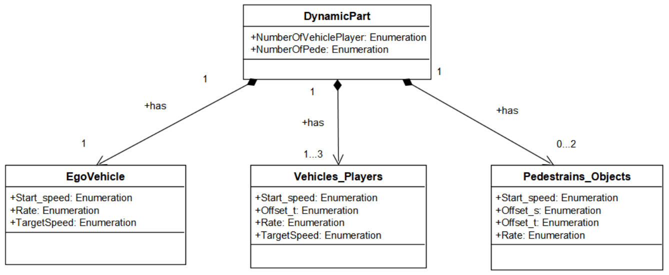

Fig. 1. Constructed AEB ontology based on EuroNCAP scenarios using UML

underlying foundations, including the origins of the database of test cases used, and the statistical methods used. Afterwards, we present the experimental results, which includes the analysis of the test cases obtained using two AEB implementations, and discuss the observations. Finally, we introduce related research and conclude the paper.

II. Foundations

In this section, we outline the foundations including the test case database, which originates from using search-based testing. Furthermore, we discuss statistical methods used for analysing the obtained data.

A. Test Case Database

The test case database for this study has been compiled as part of a study currently conducted for the comparison of search-based and combinatorial test case generation in the context of Advanced Driver Assistance Systems and Autonomous Driving (ADAS/AD) testing. Based on a shared ontology, different types of test scenarios were automatically generated by each method and executed in simulation against two different AEB system implementations. The applied ontology shown in Figure 1 comprises four concepts and three compositional relations to describe the relevant car-to-car breaking (C2Cb) and pedestrian vulnerable road users (VRU) scenarios for AEB testing based on the European New Car Assessment Programme (Euro NCAP 2017) test protocols [4], [5]. The top concept *Number of vehicles* and *Number of pedestrians* holds parameters to define the number of scenario members, which are selected randomly during the scenario generation process. Specifically, the *Number of vehicles* can vary from one to three and the *Number of pedestrians* can vary from zero to two. Scenarios with Zero pedestrians are considered in accordance with the C2Cb test protocol, where no vulnerable road users are included. In total there are nine scenario types as summarized in Table I.

For instance, the scenario type E3V2P comprises the ego-vehicle which carries the AEB system under test, three vehicle players (one driving and two parking aside) as well as two pedestrians crossing the street one from the right-side and the other from the left-side. The dynamic and positional properties of theses scenario members are defined in the three composer concepts. The parameter and value sequences required to form

TABLE I
Scenario Types

Scenario Types	3 Vehicles	2 Vehicles	1 Vehicle
2 Pedestrians	E3V2P	E2V2P	E1V2P
1 Pedestrians	E3V1P	E2V1P	E1V1P
0 Pedestrians	E3V0P	E2V0P	E1V0P

TABLE II
Generated input model

Parameter	R name	Range	Unit
NumberOfVehiclePlayer	veh	$1, 2, 3$	
NumberOfPede	ped	$1, 2$ or $NULL$	
EgoVehicle1_Start_speed	v	$(0, 5, \ldots, 150)/3.6$	m/s
EgoVehicle1_Rate	r	$5, 6, \ldots, 10$	m/s^2
EgoVehicle1_Target_Speed	t	$14, 28, 42, 55$	m/s
Pedestrains_Objects1_Start_speed	vp1	$(0, 1, \ldots, 10)/3.6$ or $NULL$	m/s
Pedestrains_Objects1_Offset_s	osp1	$3, 4, \ldots, 13$ or $NULL$	m
Pedestrains_Objects1_Offset_t	otp1	$100, 110, \ldots, 200$ or $NULL$	m
Pedestrains_Objects1_Rate	rp1	$0, 1, \ldots, 8$ or $NULL$	m/s^2
Pedestrains_Objects2_Start_speed	vp2	$(0, 1, \ldots, 10)/3.6$ or $NULL$	m/s
Pedestrains_Objects2_Offset_s	osp2	$-19, -18, \ldots, -9$ or $NULL$	m
Pedestrains_Objects2_Offset_t	otp2	$100, 110, \ldots, 200$ or $NULL$	m
Pedestrains_Objects2_Rate	rp2	$0, 1, \ldots, 8$	m/s^2
Vehicles_Players1_Start_speed	vv1	$0, 1, \ldots, 41$	m/s
Vehicles_Players1_Offset_t	otv1	$100, 105, \ldots, 200$ or $NULL$	m
Vehicles_Players1_Rate	rv1	$3, 4, \ldots, 10$ or $NULL$	m/s^2
Vehicles_Players1_Target_Speed	tv1	$0, 1, 2$	m/s
Vehicles_Players2_Offset_t	otv2	$100, 105, \ldots, 200$ or $NULL$	m
Vehicles_Players2_Start_speed	vv2	$NULL$	
Vehicles_Players2_Rate	rv2	$NULL$	
Vehicles_Players2_Target_Speed	tv2	$NULL$	
Vehicles_Players3_Offset_t	otv3	$100, 105, \ldots, 200$ or $NULL$	m
Vehicles_Players3_Start_speed	vv3	$NULL$	
Vehicles_Players3_Rate	rv3	$NULL$	
Vehicles_Players3_Target_Speed	tv3	$NULL$	

a concrete scenario instantiation are shown in Table II in accordance with the EuroNCAP test protocols.

Since *Vehicle1* is always the leading vehicle on the street, both other vehicles have no dynamic information, therefore the parameter range is $NULL$. For search-based testing we used a genetic algorithm as explained in detail in our previous work [2], [6], however this time we generated the seed population directly from the input model. Each individual in the seed population represents a separate test scenario and encodes six genes, one for the Ego vehicle, three for the vehicle players and two for the pedestrian players. Every gene furthermore comprises a set of chromosomes, which hold concrete parameter values that define the positional and dynamic properties of every scenario member. If two individuals are selected for crossing, they swap parts of their gene sequences to form new offspring for the next generation. If an individual is selected for mutation, a new parameter value from the input model is assigned to a gene's chromosome, based on an independent probability. Within every test run, we restarted the genetic algorithm after five generations and created a new seed with 40 individuals from the input model.

As database for the present study we consider the test results obtained from ten separate search-based test runs as summarized in Table III for AEB1 and Table IV for AEB2. In total we generated 11277 test scenarios for AEB1 and 11214 test

TABLE III
AEB1 TEST RESULTS

AEB1	Total	Unique	Pass	Fail	FCV	CP1	CP2
0	1,126	1,024	806	218	0	218	1
1	1,130	983	857	126	0	125	1
2	1,144	1,002	802	200	0	199	1
3	1,132	1,010	743	267	0	266	1
4	1,119	1,005	807	198	3	192	5
5	1,116	988	884	104	0	98	7
6	1,128	971	743	228	0	227	5
7	1,132	990	771	219	0	217	2
8	1,115	963	764	199	0	199	0
9	1,135	1,021	801	220	2	216	2
Total	11,277	9,957	7,978	1,979	5	1,957	25

TABLE IV
AEB2 TEST RESULTS

AEB2	Total	Unique	Pass	Fail	FCV	CP1	CP2
0	1,139	976	867	109	96	14	0
1	1,109	916	287	629	581	82	0
2	1,123	897	309	588	537	116	17
3	1,131	990	746	244	218	71	0
4	1,115	923	285	638	581	123	1
5	1,108	931	282	649	602	116	0
6	1,123	925	264	661	589	97	16
7	1,113	907	677	230	188	44	0
8	1,120	1,006	896	110	69	37	8
9	1,133	946	837	109	98	12	0
Total	11,214	9,417	5,450	3,967	3,559	712	42

scenarios for AEB2. After removal of duplicates, 9957 unique test scenarios remain for AEB1 and 9417 for AEB2. If during scenario execution a vehicle crash flag (FCV) or pedestrian crash flag (CP1, CP2) is triggered, the scenario is considered failed, otherwise passed. Here, it is worth mentioning that for certain scenario constellations more than one crash flag gets triggered, which explains the deviation between sum of crash flags and sum of failing test cases in both tables.

For AEB1 we observed a reasonable high number of crashes with *pedestrian1* in every single test run, in total 1957. In contrast, only 25 crashes were observed with *pedestrian2*. For AEB1 only 5 *vehicle* crashes were observed, whereas for AEB2 the majority of failing scenarios resulted in a *vehicle* crash. For the pedestrians we observed 712 crashes with *pedestrian1* and 42 scenarios resulted in a crash with *pedestrian2*. Considering both the input model and the presented results for AEB1 and AEB2, we are especially interested to understand which parameter interactions and parameter value ranges have a high probability to result in a crash. In the following section we present the statistical method for data analysis.

B. Statistical methods for data analysis

The data situation here is characterized by a binary dependent variable and a set of input parameters. The binary variable captures the occurrence of an accident, and the input parameters characterize the experimental setting in the simulation. The main research question here is: Are there special areas in the design space where accidents are very likely. As the target variable is binary

$$Y = \begin{cases} 1 & \text{if an accident occurs} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

and the corresponding distribution is Bernoulli, i.e. $Y \sim \text{Bernoulli}(\pi)$ the question can be reformulated as: Are there special areas in the design space where π is very large. In terms of hypothesis testing we have a null hypothesis H_0 which states that π is the same over the design space - the target variable is homogeneous, and a alternative hypothesis H_1 which is simply the negation of H_0. The data analysis should produce reliable answers to the following questions:

1) Is the distribution of the target variable homogeneous over the feature space?
2) If the distribution of the target variable is not homogeneous, how many groups of homogeneous areas are prevailing?
3) How can these groups be characterized in terms of the design space?

For a binary dependent variable, logistic regression is a method that estimates the relationship between explanatory variables and a binary variable. Once this relationship is determined one can identify regions in the design space, where the probability of crash is high. This procedure requires to estimate a regression model first and then to find out where the high probability areas are located in the design space. This two-step approach can be avoided if CART (Classification and Regression Trees, [7]) is used. CART is a methodology that gives answers to the questions above. It uses a recursive partitioning algorithm together with a performance measure to identify groups in the data. For classification target variables the performance measures are the information index

$$I = -\pi \log \pi \tag{2}$$

and the Gini index

$$G = \pi(1 - \pi) \tag{3}$$

which coincide for the binary case almost always. By default the Gini index is used in package rpart of the software R [8].

The empirical approach in this study is exploratory, and, hence, techniques like pruning and cross-validation, which aim at model confirmation, are beyond the scope of this paper. In future research that will be based on the findings here, confirmation will certainly be of interest, but at the moment the data base does not support the data requirements of the implied methods.

Nevertheless, we aim at a parsimonious model that has explanatory power. We use the Bayesian Information Criterion (BIC) as a likelihood based performance measure for model selection. It is defined as

$$BIC = -2\ell + \log(n)p, \tag{4}$$

where $\ell = \log L$ and

$$L = \prod_j \prod_i \pi_j^{y_{ij}} (1 - \pi_j)^{1-y_{ij}} \quad (5)$$

is the likelihood of the data under some model with $j = 1, \ldots, m$ groups in the data, $p = \sum_j p_j = k \times m = m$ is the number of parameters in the model, and n is the number of observations in the data. For the Bernoulli distribution $k = 1$ and hence in this setting we have $p = m$. The BIC considers model complexity through the term $\log(n)p$. For competing models the first term in BIC decreases monotonously with increasing p, while the second term increases. Hence the criterion shows a u-shaped curve when plotted over a wide range of p, and the model with $\min(BIC)$ is often selected.

We use the default settings of the R function `rpart()` to obtain proposals for parsimonious models. These models are then compared to the null model ($p = 1$) and the full model by using the corresponding BIC values.

III. Experimental results

From the Tables III and IV we find that of the six possible crash variables FCV, CP1, CP2 for AEB1 and AEB2 only four show a substantial frequency of occurrence. For the AEB1 database, the number of observed FCV and CP2 are so low that a meaningful modelling with CART is not possible.

The upper part of Table V shows the BIC values of the remaining four binary variables for the null, the proposed and the full model. In the middle of this table we see the values of the relative BIC obtained as $(1 - b_j/b_0) \times 100$, where b_0 denotes the BIC value for the null model and the index j runs over the null, the proposed and the full model.

For AEB1 we have a very good performance of the proposed model for CP1. The same holds for the AEB2 models for CP1 and CP2. The worst result is observed for FVC in AEB2. Here we have a value of BIC=11,626.64, which is roughly in the middle between the corresponding values for the null and the full model: BIC_{null}=14,060.69, BIC_{full}=8,426.38. Hence we argue that even for the worst case the performance of the proposed model is roughly average and we keep the proposed models for interpretation.

Table VI shows the 19 decision rules we obtained from the AEB1 model for crashes with *pedestrian1*. In the second column $\hat{\pi}_j$ describes the probability for a concrete scenario to result in a crash, considering the conditions defined by each rule. The conditions comprise between one and seven parameters and their relevant value ranges. Parameters are ordered from left to right based on the number of rules they appear in. In the last column n_j shows the number of observations in the data for each rule. In the AEB1 model for crashes with *pedestrian1*, the most frequently used parameter is *osp1* that describers the distance of *pedestrian1* towards the street and is included in all decision rules.

The rule with the lowest parameter interactions is *Rule 4*, which only relies on *osp1* and describes that if the *pedestrian1* starting position is greater than or equal to six meters away from the street we have a close to zero probability to observe a crash. This rule is supported by 4,353 observations in the data. However, even if *osp1* is smaller than six, we still have a close to zero crash probability if the *ego vehicle* target speed *t* is smaller than 21, as described in *Rule 3*.

TABLE V
Model Performance

			BIC	
SUT	Response	null	proposed	full
AEB1	CP1	11,532.81	4,455.58	2,819.52
AEB2	CP1	6,257.57	2,701.63	2,076.68
AEB2	CP2	588.22	128.70	128.70
AEB2	FCV	14,060.69	11,626.64	8,426.38
			Relative BIC [%]	
SUT	Response	null	proposed	full
AEB1	CP1	0	61.37	75.55
AEB2	CP1	0	56.83	66.81
AEB2	CP2	0	78.12	78.12
AEB2	FCV	0	17.31	40.07
			Number of parameters	
SUT	Response	null	proposed	full
AEB1	CP1	1	19	114
AEB2	CP1	1	23	87
AEB2	CP2	1	6	6
AEB2	FCV	1	9	306

The regions of high crash probability (above 80%) require an interaction of at least four parameters as in *Rule 18*, which is supported by 1,114 observations in the data. Here, we reach a 95% probability for a crash with *pedestrian1*, if *osp1* is between four and five, *t* greater than or equal to 21, *pedestrian1* start speed *vp1* is greater than or equal to 1.3 and the *vehicle1* start speed *vv0* is greater than or equal to nine. If *pedestrian1* is standing very close to the street for instance when *osp1* is smaller than 4 it early enters the sensor cone and the braking system can quickly react.

However, there are still two conditions that result in a high crash probability, either *Rule 19* with an interaction of five parameters or *Rule 17* with an interaction of six parameters. In *Rule 19* the distance between *pedestrian1* and *ego vehicle* must be very large with *otp1* greater than or equal to 195, presumably required for *ego vehicle* to sufficiently accelerate to high velocity. Whereas, if the distance between *ego vehicle* and *pedestrian1* is lower as in *Rule 17*, the *ego vehicle* acceleration rate *r* must be greater than or equal to six. Overall, the impact of *otp1* on the scenario outcome is significant if we compare *Rule 19* with *Rule 1*, which share the exact same conditions except for *otp1*, with completely different outcome. Table VII shows the 23 decision rules we obtained from the AEB2 model for crashes with *pedestrian1*. Here the rules comprise between one and nine parameters and again *osp1*

TABLE VI
DECISION RULES FOR THE MODEL AEB1 CP1

Rule	$\hat{\pi}_j$	osp1	t	vp1	vv0	otp1	r	otv0	n_j
1	0.00	< 4	≥ 21	≥ 0.7	≥ 9	$165 - 195$			127
2	0.00	< 4	≥ 21	$0.7 - 2.1$	≥ 9	$135 - 165$	< 6		30
3	0.02	< 6	< 21						1,219
4	0.02	≥ 6							4,353
5	0.06	< 6	≥ 21	< 0.7					583
6	0.10	< 4	≥ 21	≥ 2.4	< 9	≥ 145	≥ 6	≥ 148	49
7	0.14	< 6	≥ 21	≥ 0.7	< 9			< 148	211
8	0.15	$4 - 6$	≥ 21	≥ 0.7	< 9	≥ 145	≥ 6	≥ 148	200
9	0.21	$5 - 6$	≥ 21	≥ 1.3	≥ 9	≥ 155			99
10	0.24	< 4	≥ 21	≥ 0.7	≥ 9	< 135			224
11	0.25	$4 - 6$	≥ 21	$0.7 - 1.3$	≥ 9				154
12	0.33	< 4	≥ 21	≥ 2.1	≥ 9	$135 - 165$			178
13	0.57	< 6	≥ 21	≥ 0.7	< 9	< 145	≥ 6	≥ 148	268
14	0.71	< 6	≥ 21	≥ 0.7	< 9		< 6	≥ 148	144
15	0.77	< 4	≥ 21	$0.7 - 2.4$	< 9	≥ 145	≥ 6	≥ 148	115
16	0.82	$5 - 6$	≥ 21	≥ 1.3	≥ 9	< 155			271
17	0.87	< 4	≥ 21	$0.7 - 2.1$	≥ 9	$135 - 165$	≥ 6		336
18	0.95	$4 - 5$	≥ 21	≥ 1.3	≥ 9				1,114
19	0.96	< 4	≥ 21	≥ 0.7	≥ 9	≥ 195			337

TABLE VII
DECISION RULES FOR THE MODEL AEB2 CP1

Rule	$\hat{\pi}_j$	osp1	vp1	rp1	t	otp1	vv0	otv0	vp2	otv2	n_j
1	0.00	≥ 9									5901
2	0.00	$6 - 7$	$0.97 - 1.50$								139
3	0.01	$7 - 9$	< 2.08								1143
4	0.03	$4 - 7$	< 0.97								868
5	0.05	$4 - 5$	$0.97 - 1.77$	≥ 2	< 21	$105 - 165$					44
6	0.06	< 4	< 0.97				≥ 9	≥ 168			18
7	0.06	$7 - 9$	≥ 2.08		≥ 21						125
8	0.08	$7 - 9$	≥ 2.08		< 21		< 2				36
9	0.19	< 6	≥ 0.97	< 2			< 2				31
10	0.22	< 4	< 0.97					< 168			136
11	0.24	$6 - 7$	≥ 1.50	≥ 4					≥ 0.97		37
12	0.25	< 6	≥ 1.77	≥ 2							485
13	0.28	$6 - 7$	≥ 1.50	< 4							182
14	0.29	< 6	$0.97 - 1.77$	≥ 2		≥ 165					126
15	0.34	< 6	$0.97 - 1.77$	≥ 2	≥ 21	< 165				≥ 128	143
16	0.69	$6 - 7$	≥ 1.50	≥ 4					< 0.97		99
17	0.75	< 6	≥ 0.97	< 2			≥ 2				220
18	0.78	< 6	$0.97 - 1.77$	≥ 2	≥ 21	< 165				< 128	18
19	0.81	$7 - 9$	≥ 2.08		< 21		≥ 2				170
20	0.81	< 4	$0.97 - 1.77$	≥ 2	< 21	$105 - 165$					16
21	0.84	< 5	$0.97 - 1.77$	≥ 2	< 21	< 105					37
22	0.91	< 4	< 0.97				< 9	≥ 168			45
23	0.91	$5 - 6$	$0.97 - 1.77$	≥ 2	< 21	< 165					125

is used most frequently and included every rule. We obtain a close to zero probability to observe a crash if *osp1* is greater than or equal to nine, as described in *Rule 1* which is supported by 5,901 observations in the data. A general observation on *Rule 20,21,23* with crash probability greater than 80% is that they share an *osp1* value below nine, a *pedestrian1* start speed *vp1* greater than 0.97, a *pedestrian1* rate *rp1* greater than or equal to 2 and an ego target speed *t* smaller than 21. For *pedestrian2* crashes the constructed AEB2 model is very compact and precise. The six decision rules we obtained are graphically displayed in Figure 2 and summarized in Table VIII. The decision rules comprise between one and five parameters. Here the *pedestrian2* start speed *vp2* is included in all rules and if *vp2* is smaller than 2.4 the crash probability for is zero as described in *Rule 2* and supported by 9,141 observations in the data. The highest crash probability is described by the condition in *Rule 6*, when *pedestrian2* start speed *vp2* is greater than or equal to 2.4, *pedestrian2* distance towards the street *osp2* is greater than or equal to -9 and *pedestrian2* rate *rp2* is greater than or equal to six. If *rp2* is smaller than six, there is still a reasonable probability of 69% to observe a crash, if *pedestrian1* rate *rp1* is smaller than 2 and the ego target speed *t* is smaller than 21. In other words this

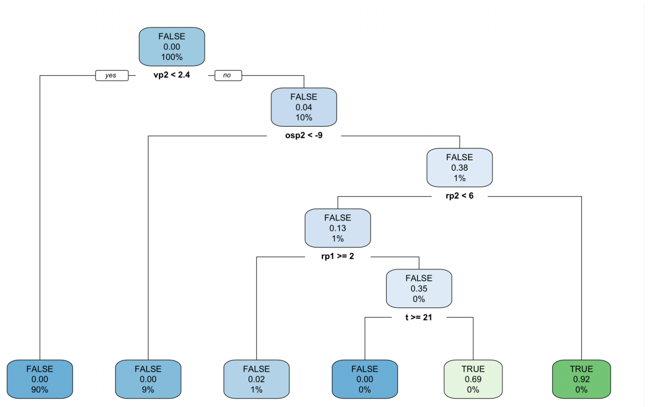

Fig. 2. Decision Tree For The Model AEB2 CP2

TABLE VIII
DECISION RULES FOR THE MODEL AEB2 CP2

Rule	$\hat{\pi}_j$	vp2	osp2	rp2	rp1	t	n_j
1	0.00	≥ 2.4	≥ -9	< 6	< 2	≥ 21	13
2	0.00	< 2.4					9141
3	0.00	≥ 2.4	< -9				888
4	0.02	≥ 2.4	≥ -9	< 6	≥ 2		52
5	0.69	≥ 2.4	≥ -9	< 6	< 2	< 21	13
6	0.92	≥ 2.4	≥ -9	≥ 6			37

means that the probability for a *pedestrian1* crash to happen first is reduced, if we consider the rules for *pedestrian1*.

Table IX shows the 9 decision rules we obtained from the AEB2 model for crashes with *vehicle1*. The decision rules comprise between one and four parameters and the *vehicle1* start speed *vv0* is included in all rules. Based on the model, if *vv0* is greater than or equal to 20 there is a close to zero probability to observe a crash with *vehicle1*. The highest crash probability is observed in *Rule 9* based on the combination of four parameters. He a crash with *vehicle1* happens with 89% probability, when *vv0* is smaller than 20, the *ego vehicle* target speed *t* is smaller than 21, the *vehicle1* rate *rv0* is smaller than six and the distance between both vehicles is greater than or equal to 198.

TABLE IX
DECISION RULES FOR THE MODEL AEB2 FCV

Rule	$\hat{\pi}_j$	vv0	t	rv0	otv0	osp1	otv1	n_j
1	0.07	< 20	≥ 21			< 6		457
2	0.07	≥ 20						1308
3	0.35	< 8	< 21	$4 - 6$	< 198			1259
4	0.38	< 20	≥ 21			≥ 6	< 193	1036
5	0.59	< 20	≥ 21			≥ 6	≥ 193	1224
6	0.60	< 8	< 21	< 4	< 198			501
7	0.68	< 20	< 21	≥ 6				3895
8	0.72	$8 - 20$	< 21	< 6	< 198			290
9	0.89	< 20	< 21	< 6	≥ 198			174

IV. Discussion

In the previous chapter, we presented the experimental results of the classification and regression tree analysis to investigate the number of parameter combinations required to find pedestrian and vehicle crashes. First we selected an appropriate configuration for building a tree model and demonstrated the capability of this method to identify parameter combinations and regions in the input space that result in a crash with high probability. For AEB1 we build a tree model, where the obtained decision rules suggest, that a minimum combination of four parameters is required to find a crash with *pedestrian1*.

The tree model build for AEB2 also suggest a minimum combination of four parameter to find crashes with *pedestrian1* and a minimum of three parameter combinations to find crashes with *pedestrian2*. All vehicle crashes found for AEB2 can be traced back to the combination of four parameters. Even though, for both models we also identified rules that require a higher interaction of parameter combinations we can not exclude the possibility that all crashes happen based on the same root cause, since we are testing black-box AEB implementations.

Because of the obtained findings, we would recommend for testing methods such as combinatorial testing to consider at least parameter combinations of strength four or higher to provably cover all crash types found by search-based testing, considering the given data base and decision tree models. This recommendation is in line with other similar observations based on different systems, where a combinatorial strength of 4–6 seems to be enough for detecting all faults (see [9]).

Threads to validity of our analysis include the following issues: We are relying on a particular test case database comprising tests for two AEB implementations obtained when applying search-based testing and in particular a genetic algorithm. It might be the case that the test case data is biased due to issues in the used genetic algorithm. However, this seems to be unlikely because the tests cover a wide range of different scenarios leading to crashes.

Because the analysis presented in the paper is based on only two AEB implementations, the results can hardly be generalized. Therefore, we tried to avoid making generalized statements but use the obtained results as indication what to expect and as a motivation for further future research activities. In addition, the presented analysis might serve as a framework applied to other test case generation approaches of different automated and autonomous driving functions.

V. Related research

In [10] Zeller argues that search-based techniques are best suited for testing at system level. Many researchers studied this claim and used search-based testing approaches for testing advanced driver assistance systems and autonomous driving functions [6], [11]–[16].

In [6] we used a genetic algorithm for parameter optimization in scenario-based testing of an automated braking function. In [11] Ben Abdessalem et. al. applied multi-objective search and learnable evolutionary algorithms [12] to produce test cases that violate safety requirements of self-driving systems. Bühler and Wagner carried out evolutionary functional testing of a vehicle brake assistant system [14] and an autonomous parking assistant [13] concluding that the system errors detected are unlikely to be found with conventional testing techniques. In [15] researchers applied a search-based test algorithm to reveal interaction failures between independent units of functionality in autonomous vehicle systems. In [16] researchers performed a comprehensive study to demonstrate the effectiveness of search-based testing methods in generating challenging test cases for autonomous vehicle software compared to full combinatorial testing and monte-carlo-testing.

Whereas research in the mentioned studies primarily focused on the efficiency and effectiveness of testing, they did not investigate which parameters in the input space are relevant and what degree of parameter combinations is required to detect a fault. In the present study we addressed this topic and performed an in-depth analysis of experimental results obtained form search-based testing that revealed important parameters and parameter interactions that lead to failure in the driving function with high probability.

VI. Conclusions

Testing automated and autonomous driving functions is a necessity for assuring that the vehicle making use of these functions is safe. As discussed testing requires to come up with critical scenarios, i.e., interactions between the vehicle and its environment including other cars and pedestrians, to reveal faults. Search-based testing, because of its capabilities to guide search towards reaching predefined goals like the minimizing time to collision, seems to be an appropriate testing approach.

In this paper, we present the results from an in-depth analysis of test suites obtained when applying search-based testing to two different automatic emergency braking functions. The analysis revealed important parameters of tests that are required to meet conditions in order to cause crashes. Moreover, we saw that combinations of at least 4 parameters are required to come up with critical scenarios of a high probability. These results may indicate that other approaches like combinatorial testing when applied to autonomous and automated driving should take care of a higher number of parameter combinations for assuring that faults can be revealed. In addition, the presented analysis may also be used as a blueprint for analysing the test suites obtained using other test case generation approaches.

Future research has to include extending the evaluation and analysis to other automated or autonomous driving functions, to generalize the findings of this paper. In addition, investigations on a comparison of different testing techniques should carried out to clarify the question which testing methodology is most appropriate for testing in the context of autonomous driving.

REFERENCES

[1] Y. Li, J. Tao, and F. Wotawa, "Ontology-based test generation for automated and autonomous driving functions," vol. 117, no. October 2019, 2020.

[2] F. Klück, M. Zimmermann, F. Wotawa, and M. Nica, "Performance comparison of two search-based testing strategies for adas system validation," in *Testing Software and Systems*, C. Gaston, N. Kosmatov, and P. Le Gall, Eds. Cham: Springer International Publishing, 2019, pp. 140–156.

[3] F. Wotawa, "On the use of available testing methods for verification & validation of ai-based software and systems," in *SafeAI@AAAI*, ser. CEUR Workshop Proceedings, vol. 2808. CEUR-WS.org, 2021.

[4] N. Euro, "Test protocol - aeb car-to-car systems," *Brussels, Belgium: Eur. New Car Assess. Programme (Euro NCAP)*, 2017.

[5] ——, "Test protocol - aeb vru systems," 2017.

[6] 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.

[7] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Montery, CA: Wadsworth and Brooks, 1984.

[8] B. R. Terry Therneau, Beth Atkinson, *rpart: Recursive Partitioning and Regression Trees*, 2019, r package version 4.1-15. [Online]. Available: https://CRAN.R-project.org/package=rpart

[9] D. Kuhn, R. Kacker, Y. Lei, and J. Hunter, "Combinatorial software testing," *Computer*, pp. 94–96, August 2009.

[10] A. Zeller, "Search-based testing and system testing: A marriage in heaven," in *2017 IEEE/ACM 10th International Workshop on Search-Based Software Testing (SBST)*, 2017, pp. 49–50.

[11] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing advanced driver assistance systems using multi-objective search and neural networks," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE 2016. New York, NY, USA: Association for Computing Machinery, 2016, p. 63–74. [Online]. Available: https://doi.org/10.1145/2970276.2970311

[12] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 1016–1026.

[13] O. Bühler and J. Wegener, "Automatic testing of an autonomous parking system using evolutionary computation," *SAE Technical Papers*, 03 2004.

[14] ——, "Evolutionary functional testing," *Computers Operations Research*, vol. 35, pp. 3144–3160, 10 2008.

[15] R. Ben Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2018, pp. 143–154.

[16] M.-C. Chiu, K. M. Betts, and M. D. Petty, "Automated search-based robustness testing for autonomous vehicle software," *Modelling and Simulation in Engineering*, vol. 2016, p. 5309348, 2016. [Online]. Available: https://doi.org/10.1155/2016/5309348