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

PrePrint 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

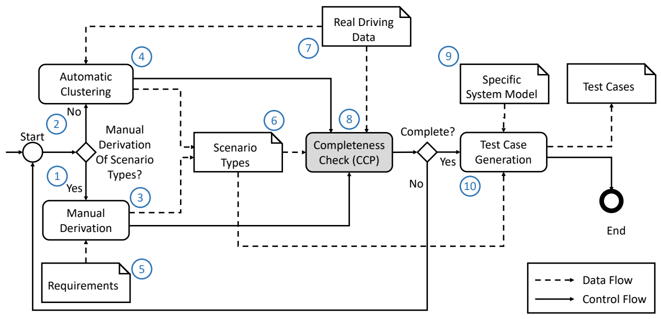

Fig. 1. Big Picture

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

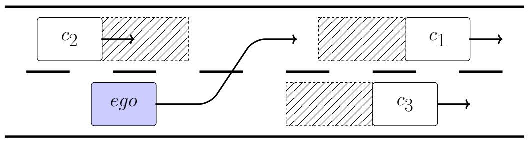

Fig. 2. Example test case for testing lane change functionality

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 σ 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 τ. Therefore, we search for the smallest number of samples S, for which the added up occurrences are equal to or greater than τ.

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

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

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.

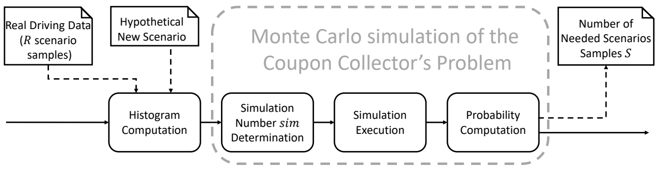
Real Driving Data (R scenario samples)
Hypothetical New Scenario
Monte Carlo simulation of the Coupon Collector's Problem
Number of Needed Scenarios Samples S
Histogram Computation
Simulation Number sim Determination
Simulation Execution
Probability Computation

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 τ.

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 τ, 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 τ 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 τ 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.

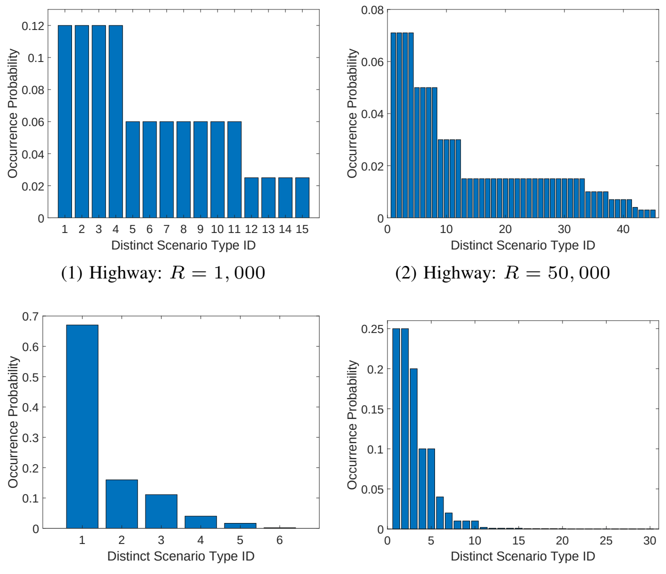

Fig. 4. Histograms of the four distributions used in the experiments

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 τ. 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 σ 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	σ	S	σ
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 τ. 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-toys-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.