On the importance of system testing for assuring safety of AI systems

Franz Wotawa[1]

[1]CD Lab for Quality Assurance Methodologies for Autonomous Cyber-Physical Systems,
TU Graz, Institute for Software Technology, Graz, Austria.
wotawa@ist.tugraz.at

Abstract

Rigorous testing of automated and autonomous systems is inevitable especially in case of safety-critical systems like cars or airplanes. There exist several functional safety standards that have to be fulfilled like IEC 61508 explicitly stating that AI methodologies are not recommended to be used in case of systems with higher safety requirements. Hence, there is a necessity to adopt these standards in a direction where AI methodology is allowed to be used providing to fulfill certain standardized quality assurance method to be taken care of during development. In this paper, we contribute to this endeavor and discuss the urgent need for system testing in the context of safety-critical systems comprising AI methodologies. In particular, we argue based on one example from the automotive industry that it is strongly recommended to consider not only subsystems but instead the whole system interacting with its environment when carrying out tests. The discussed example is an advanced driver-assistance systems used to break in case of an emergency that does not rely on machine learning but comprises a decision part that invokes breaking once the sensors identify an obstacle that might be hit otherwise. Results obtained from an already reported testing methodology, revealed that when using tests considering the environment of an automated emergency breaking systems, we obtain critical scenarios that might otherwise have not been detected. From this observation, we conclude that rigorous system testing becomes even more important for systems with AI methodology based on machine learning or allowing to adapt the system's behavior during operation.

1 Introduction

Safety-critical systems are systems where internal fault or a malfunction may cause death or serious injury to people, loss or severe damage to property, or environmental harm. For such systems, software and system engineering standards have been introduced like IEC 61508 or ISO 26262 for the automotive industry. The latter introduces automotive safety integrity levels (ASIL) where ASIL A is the weakest and ASIL D the strongest requiring specific considerations during development in order to keep risks below an acceptable level. It is well known that standards like IEC 61580 do not recommend AI methodology to be used in systems with a higher safety integrity level (see IEC 61580-3:2010 Table A.2). It is interesting to note that this restriction applies also to the automotive industry where we see an increasing use of automated and autonomous functions some of them also based on machine learning technologies. There is obviously a gap between the safety standards and the use of AI methodology in practice requiring adaptations in the standards. See for example, Henriksson and colleagues [Henriksson *et al.*, 2018] contributed ideas for evolving the standards towards capturing machine learning applications. In addition, there are new standards coming up like ISO/PAS 21448:2019 considering safety of the intended functionality for road vehicles that consider situations comprising complex sensors and processing algorithms including the application of machine learning.

There have been many papers dealing with the general challenge of verifying and validating autonomous vehicles like Koopman and Wagner [Koopman and Wagner, 2016], Wotawa [Wotawa, 2016a], Schuldt and colleagues [Schuldt *et al.*, 2018], or Wotawa and colleagues [Wotawa *et al.*, 2018]. An essential challenge in this context is how to assure to test such systems in a way that can be considered as good enough? Kalra and Paddock [Kalra and Paddock, 2016] answered this question stating that an autonomous vehicle has to operate for 275 million miles for verification purposes. In their calculation, Kalra and Paddock considered the fatality rate of driving in the USA and assumed that an autonomous vehicle should have a far lower fatality rate. Despite the fact that such a huge number of miles can hardly be achieved with a small fleet of test cars, there is also another hidden assumption behind the calculation, i.e., during testing on streets the autonomous vehicle has to deal with all critical scenarios, which seems to be somehow unrealistic. Hence, as a consequence, researchers have proposed to use ontologies for testing automated and autonomous vehicles, e.g., [Xiong *et al.*, 2013; Geyer *et al.*, 2014; Menzel *et al.*, 2018]. These contributions deal with testing the whole system using different scenarios.

The intention behind this paper is to discuss the need for system testing in the context of automated and autonomous driving and to learn important requirements that can be used

in other application areas utilizing AI technology as well. In particular, we discuss results obtained when applying two different testing techniques to verify the functionality of the advanced driver-assistance system (ADAS) autonomous emergency breaking (AEB). The AEB comprises sensors for detection obstacles, and a decision system that controls automated breaking whenever necessary to avoid or mitigate collisions. Currently, AEB systems utilize different sensor technologies like radar, cameras, or LIDAR, together with sensor fusion capabilities. In cases where obstacles need to be classified, e.g., as persons or bicyclists, machine learning methods might be applied to learn the vision sensor distinguishing categories. Verification of AEB requires to verify its subsystems as well as the system itself during operation or at least in an environment that is close to the real use.

There are similarities and expected differences when considering systems comprising AI methods like machine learning as subjects of testing. For example, in case of machine learning the outcome depends on the data used for learning a certain model and the underlying machine learning approach. Hence, the outcome of the finally obtained model might vary. Other parts of the overall system relying on such a varying outcome have to deal with this uncertainties in an appropriate way not causing safety hazards. Hence, any verification approach needs to consider this variation and try find a critical situation. In addition, sensor based on machine learning may not always deliver the correct classification results. Even if coming up with a correct classification in 99% of the cases, we have to assure that the 1% of the remaining cases does not lead to safety violations. Hence, the overall system has to compensate for inaccuracies that might be higher than for ordinary sensors, and there is a strong requirement to verify the system especially considering all the cases where classification goes wrong.

In the following, we will introduce such a testing environment that is used together with test case generation methods to carry out system testing automatically without the need of user interference. Interestingly, we will see that there are testing approaches that reveal faults in an AEB that very much likely would not have been found otherwise. This testing approach combines environmental ontologies with combinatorial testing [Kuhn *et al.*, 2012; Kuhn *et al.*, 2015]. The underlying assumption is that there is a need for finding combinations of environmental entities for revealing faults. Hence, we argue that it is not only necessary to carry out system tests but also to consider environmental interactions in case of autonomous systems. In addition, we discuss some further challenges of testing autonomous systems, i.e., providing some sort of guarantees and methods for estimating the residual risk, i.e., the risk of still comprising a fault even after carrying out a proposed testing methodology. We will see that the use of environmental models allow for specifying such guarantees based on the degree of considering interactions between environmental entities and the degree to which the environmental model has been used for verifying a particular system.

This paper is organized as follows: In Section 2 we discuss related research focusing on AI-based systems. Afterwards, and to be self-contained, we introduce the results from another paper dealing with system testing of an AEB in Section 3, from which we are going to derive requirements necessary to verify safety-critical systems using AI methods (Section 4). Finally, we conclude the paper in Section 5.

2 Related research

Testing AI-based system itself is not a novel research area. Starting in the 90s of the last century, R. Plant [Plant, 1991; Plant, 1992] reported on the development of expert systems and also considering testing. Together with S. Murell, R. Plant [Murrell and Plant, 1997] published also a survey on tools for verifying and validating knowledge-based systems that had been published between 1985 and 1995. Later El-Korany and colleagues [El-Korany *et al.*, 2000] presented a structured approach, and Hartung and Håkansson [Hartung and Håkansson, 2007] an automated approach for testing such systems. Other work, in the field of knowledge-based system testing include [Hayes and Parzen, 1997], [Felfernig *et al.*, 2005] dealing with testing recommender systems, [Tiihonen *et al.*, 2002], and [Wotawa and Pill, 2014]. Most recently, there has been some paper dealing with testing specific properties of logic reasoning engines [Wotawa, 2018a], the general challenge of testing such systems [Wotawa, 2018b], and testing subsystems of logic reasoning engines like their compilers [Koroglu and Wotawa, 2019]. In any of these cases, the focus were on testing the reasoning engine alone not considering its use in a specific application like a mobile robot or any other application that requires reasoning capabilities. In contrast, Wotawa [Wotawa, 2016b] presented a testing approach of adaptive systems that make use of models of the system itself, i.e., knowledge of system's internal structure and behavior. There the author considers fault injection for testing whether the adaptive system handles internal faults as expected.

In case of machine learning and in particular neural networks there have been many papers dealing with testing including [Ma *et al.*, 2018a], [Sun *et al.*, 2018], [Pei *et al.*, 2017], [Ma *et al.*, 2018b], and [Ma *et al.*, 2018c] applying different well-known testing techniques, like mutation testing, combinatorial testing or whitebox testing approaches to neural networks. Chetouane and colleagues [Chetouane *et al.*, 2019] discussed a slightly different approach to testing neural networks using mutation testing and coverage in a more ordinary setting. In addition, it is well known that neural networks are vulnerable against adversarial inputs where only changing one pixel in an image lead to a wrong classification. See for example, Su and colleagues [Su *et al.*, 2019] work. Adversarial attacks can also be more tailored towards more realistic attacks. We refer the interested reader to Wicker and colleagues [Wicker *et al.*, 2018] for one example. Counter measures against adversarial input has been considered. Most recently, Goddfellow and colleagues [Goodfellow *et al.*, 2018] discuss and summarize some of them.

In the context of automated and autonomous driving we discussed related papers in the introduction. The main focus is on identifying critical scenarios using ontologies from which test cases can be derived, e.g., [Xiong *et al.*, 2013; Geyer *et al.*, 2014; Menzel *et al.*, 2018]. In addition, there is

a shift from carrying out vehicle tests on the road to a simulation environment capturing physics as well as 3D models. In such an environment more tests in less time can be carried out. Because of the improvement in simulation technology, the carried out tests become closer to reality finding faults that would also been detected in a real environment. In an explorative study Sotiropoulos and colleagues [Sotiropoulos *et al.*, 2017] showed that simulation of a mobile robot in deed revealed faults that had been also detected when carrying out test in our physical world.

In addition, there have been research work on formally verifying machine learning and AI-based solutions. Seshia and Sadigh [Seshia and Sadigh, 2016] discussed the use of formal methods for verifying AI including how to handle involved challenges. Gauerhof et al. [Gauerhof *et al.*, 2018] tackled the case of the use of machine learning in the context of autonomous driving focusing on validation issues.

What we can take with us from the mentioned related research is the following: (1) Different AI methodologies like knowledge-based systems or machine learning require specific testing methods, (2) improved 3D and physics simulation allow for revealing faults, and (3) in case of automated and autonomous driving the use ontologies capturing the environment to generate critical scenarios seems to be of particular importance.

3 System testing for automated driving functions

In this section, we recapitulate an approach for system testing an ADAS functionality relying on an environmental ontology and combinatorial testing for generating test cases. The content of this section relies on Tao and colleague's paper [Tao *et al.*, 2019]. The intention of this section is to show the necessity of capturing interactions between environmental entities in the case of autonomous driving and ADAS functionality for revealing faults.

The underlying idea behind Tao et al.'s work is to automatically extract test cases from an environmental ontology directly. The basic foundations behind have been outlined in other papers. Wotawa and Li [Wotawa and Li, 2018] presented a first algorithm that allows to convert ontologies into input models of combinatorial testing [Kuhn *et al.*, 2012; Kuhn *et al.*, 2015]. An input model captures basically necessary parameters and their domains. In case of automated and autonomous driving the parameters are road fragments together with their conditions, other cars or pedestrians, the weather conditions, and so on. In order to find critical scenarios, it would be required to consider all different combinations of parameter values, which of course is not feasible. In combinatorial testing, we are not considering all combinations but only all combinations for an arbitrary subset of the set of parameters of size t. If t is smaller than the total number of parameters, we have to generated substantially fewer tests. A test suite where all combinations for all subsets of the parameters of size t are considered, is called a t-way combinatorial test suite or a test suite of strength t. When applying combinatorial testing to the domain of autonomous and automated driving, the underlying assumption is that it is sufficient to restrict testing to all combinations of all subsets of size t instead of considering all combinations of parameters.

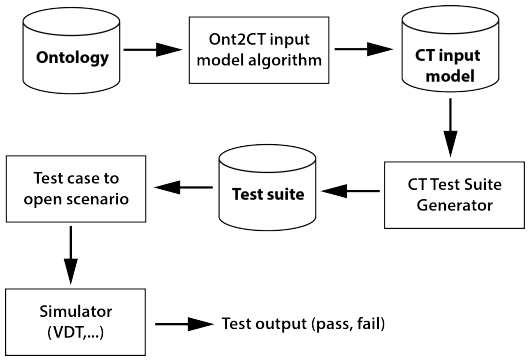

Figure 1: Overview of the testing process: from ontology to execution

Klück and colleagues [Klück *et al.*, 2018] improved the conversion algorithm from ontologies to combinatorial tests and also introduced how to apply the proposed testing methodology in a practical setup. In Figure 1 (from [Tao *et al.*, 2019]), we depict the overall application process that can be fully automated. The process starts with an ontology that captures the environment of the system under test (SUT). From this ontology, we obtain a combinatorial testing (CT) input model that is used to generate tests. Note that the generated tests are abstract tests and need to be further concretized. For example, in the ontology we may only distinguish road fragments to be straight, or a left or a right curve. The details about the length or the radius of a curve are not given. Hence, in a concretization step we have to set these values. Afterwards, the SUT can be simulated. In case of [Tao *et al.*, 2019] this is done using certain tools for 3D simulation and physical simulation like VTD or ModelConnect.

In [Tao *et al.*, 2019], the authors make use of an AEB as a SUT. Instead of considering a general ontology that captures all different scenarios that might occur during driving, the authors focus on the scenarios from the European New Car Assessment Program (Euro NCAP), which is a well-known organization for car safety performance assessment providing consumers with a safety performance assessment for the majority of the most popular cars. In Figure 2 from NCAP Euro [Euro, 2017; Euro and Protocol, 2017], we see some typical scenarios that have to be considered when testing an AEB implementation. This includes the ego vehicle, i.e., the SUT, to approach another vehicle but also to pass by parking cars and also to consider pedestrians that may cross the street. [Tao *et al.*, 2019] made use of these scenarios to come up with an adapted ontology for automated AEB testing.

Using the conversion algorithm from ontologies into CT input models and a CT algorithm for generating test cases, Tao et al. were able to generate 993 test cases from 39 parameters and a domain size of maximum 27 considering a combinatorial strength of 2 only. From these tests, Tao et al. identified 17 that lead to a crash. Interestingly to note that in the test suite we have two test cases that distinguishes only in

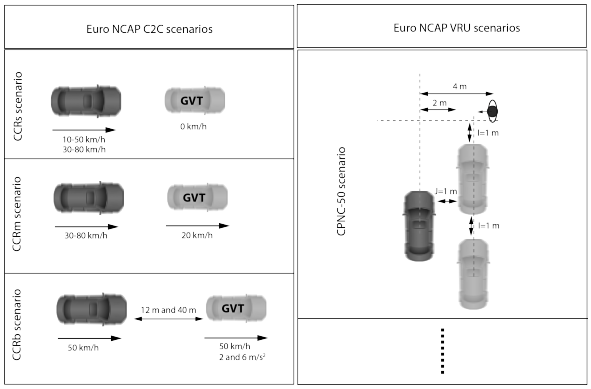

Figure 2: Some AEB scenarios from the EuroNcap protocol

the fact that one captures the situation of a dry road where the other requires the road to be wet. The latter test case leads to a crash whereas the other does not. In addition, Tao et al. also identified a case where a crash with a pedestrian happened but only because two pedestrians cross the road one from left to right and the other from right to left in close proximity. These results show that it is important to take care of the interaction of the parameters. Moreover, it was not necessary to consider all interactions, i.e., all combinations but only a few, at least partially confirming the underlying assumption that we only need to take care of all combinations for all subsets of parameters of size t.

In summary, we can conclude from the described example application of testing in this section the following: (1) System testing based on 3D and physical simulation is able to reveal faults in ADAS, (2) ontologies capturing the environment of the SUT are good enough to detect faults, (3) interactions of parameters of scenarios are required for fault detection, and (4) it seems to be sufficient considering only a restricted number of combinations of parameters of scenarios.

4 Testing safety-critical AI

In this section, we want to summarize the finding obtained from testing autonomous and automated driving functions, and in addition, generalize the finding to other application areas with safety requirements. In particular, we discuss the necessity of carrying out system tests automatically using test cases obtained from environmental knowledge, and outline challenges and partial solution regarding guarantees of testing and the prediction of remaining risks after testing. Especially, in the context of safety-critical systems the last two issues are of importance.

Automated system testing: System testing is of uttermost importance not only in the context of AI-based systems. Even in the case that each subcomponent of a SUT might be tested or formally verified thoroughly, the interaction between subsystems may reveal a faulty behavior. In the context of AI-based systems the system tests is even more important. There are two reasons: (1) An AI-based system, e.g., a vision sensor used for classification obstacles in case of an AEB, might not always deliver correct results. Hence, we have to check how the whole system deals with this fact. (2) A system implementing more and more autonomy, e.g., an AEB autonomously making a decision about invoking emergency breaking, has to be tested as a whole in very much detail. Such systems usually are at least very much complicated if not even complex. We have to assure that the system fulfills its specification under a sheer amount of potential interactions between the system and its surrounding environment. Therefore, it is also very much important to carry out testing in an automated way making use of simulation environments. Note that we do not restrict simulation in this context to simulation where all hardware parts are represented as virtual models. We may also test the whole system including hardware and software using a test bench where the hardware directly can be stimulated.

There is also another reason why automated system testing becomes increasingly important. There is a growing need for making changes in the system after deployment because of the increasing amount of software in such AI-based systems. When dealing with safety-critical systems every change causes the SUT to be again tested thoroughly. Without automation such and endeavor would be impossible to achieve considering available budget, effort and time constraints. Previous research – some briefly discussed in this paper – also demonstrates the usefulness of automated system testing for finding faults in autonomous systems using simulation environments (e.g., see [Sotiropoulos *et al.*, 2017] and [Tao *et al.*, 2019]). Therefore, we may consider this type of testing as a best practice also for safety-critical AI-based systems intended to interact with entities of our physical world.

Consider environmental knowledge: When dealing with testing the question is always how to obtain test cases? In practice, test cases are often manually crafted even in case of safety-critical systems. In addition, other approaches like model-based testing (MBT) [Schieferdecker, 2012] are used, which utilizes a model of the SUT for generating tests. MBT is without any doubt an important method for test case generation to guarantee covering the functionality of the SUT. However, in the case of AI-based systems interacting with our physical world, it is at least equally important to also consider the environmental interactions, which might also come from independent entities like other cars or pedestrians crossing the streets. Hence, finding a way to represent the knowledge we have about our world is essential for testing AI-based systems with increased autonomy or adaptive behavior. This requirement is very well supported by the large amount of research papers dealing with the use of environmental ontologies for testing such systems in the context of the autonomous driving. Without environmental models finding critical situations that might cause the SUT to violate its specification can hardly be achieved.

Providing guarantees: Testing is well-known to be incom-

plete. Only in case of a failure revealing test, we know that the SUT is still faulty. If the SUT passes all tests, either we have not used the right test case, or the SUT itself is really fulfilling its specification. Because of the fact that we cannot test forever the questions of when to stop testing and also about the consequences are of uttermost importance. Providing guarantees like knowing to achieve a certain type of coverage or mutation score is especially important when testing safety-critical systems where standards require to test for reaching a given coverage criteria. Testing the whole system usually is carried out without knowing internal details of the system i.e., as black-box testing. In order to come up with a testing criteria for testing AI-based systems we may borrow some ideas from CT.

In CT the strength t is used as means for representing coverage. This is due to the fact that t represents the number of interactions between any t parameters. Hence, t guarantees that all interactions of size t has been captured. When using CT like in [Tao *et al.*, 2019] work, the strength used for generating the test can be used as a guarantee. What is missing is a detailed analysis about meaningful values for t in the context of autonomous systems. For ordinary systems like web browsers etc. Kuhn and colleagues [Kuhn *et al.*, 2009] showed that at the maximum 6 interactions are necessary to reveal all previously detected faults. For the autonomous systems domain such an analysis is missing.

Another way of coming up with a measure representing some sort of guarantees would be to consider the used underlying ontologies themselves. For example, if we have a generally agreed ontology of the environment, we are able to judge testing with respect to the use of the environmental entities. Ontology coverage may be the percentage of concepts used in testing from such an ontology. Again research is needed for (1) coming up with such an ontology for the application domain, and (2) to define ontology coverage formally.

Estimate the residual risk: The residual risk in case of testing corresponds to the risk that the SUT after testing will fail during operation causing serious harm. Hence, the remaining risk after applying verification and validation, is proportional to the risk of missing important test cases, i.e., those leading to critical situations. When considering parameters like the combinatorial strength or ontology coverage as discussed before, the residual risk should be proportional to these parameters. As far as we know there has been no research trying to estimate the residual risk based on metrics used to specify some sort of guarantees.

In order to be of use in practice, we have to search for a method that allows to estimate the residual risk of testing automated and autonomous systems based on certain parameters like coverage, mutation score, or combinatorial strength.

5 Conclusions

Testing AI-based systems has been in the focus of research for several decades. Because of the increasing importance and use of AI methodologies ranging from knowledge-based systems to machine learning, there is a strong need for testing methodologies that come with certain guarantees. Especially, for safety-critical systems such a testing methodology would be required. In this paper, we argue that the automated system test is of uttermost importance comprising test case generation from environmental models and test execution using simulation. When relying on environmental models, i.e., ontologies, and testing techniques like combinatorial testing, the ontology coverage and the combinatorial strength can be used for giving guarantees and also for estimating potential residual risks.

Future research has to consider studies mapping the combinatorial strength to the number of not detected faults in case of autonomous and AI-based systems. In addition, we have to come up with other coverage definitions like ontology coverage and a prediction of the residual risk in case of testing.

Acknowledgment

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

References

[Chetouane *et al.*, 2019] Nour Chetouane, Lorenz Klampfl, and Franz Wotawa. Investigating the effectiveness of mutation testing tools in the context of deep neural networks. In *In Proceedings of the 15th International Work-Conference on Artificial Neural Networks (IWANN)*, Gran Canaria, Spain, 2019. Springer.

[El-Korany *et al.*, 2000] Abeer El-Korany, Ahmed Rafea, Hoda Baraka, and Saad Eid. A structured testing methodology for knowledge-based systems. In *11th International Conference on Database and Expert Systems Applications (DEXA)*, pages 427–436. Springer, 2000.

[Euro and Protocol, 2017] NCAP Euro and AEB VRU Test Protocol. Test protocol - aeb vru test, 2017.

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

[Felfernig *et al.*, 2005] A. Felfernig, K. Isak, and T. Kruggel. Testing knowledge-based recommender systems. *OEGAI Journal*, 4:12–18, 2005.

[Gauerhof *et al.*, 2018] Lydia Gauerhof, Peter Munk, and Simon Burton. Structuring validation targets of a machine learning function applied to automated driving. In Barbara Gallina, Amund Skavhaug, and Friedemann Bitsch, editors, *Computer Safety, Reliability, and Security*, pages 45–58, Cham, 2018. Springer International Publishing.

[Geyer *et al.*, 2014] S. Geyer, M. Baltzer, B. Franz, S. Hakuli, M. Kauer, M. Kienle, S. Meier, T. Weissgerber,

K. Bengler, R. Bruder, F. Flemisch, and H. Winner. Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance. *IET Intelligent Transport Systems*, 8(3):183–189, May 2014.

[Goodfellow *et al.*, 2018] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. Making machine learning robust against adversarial inputs. *Commun. ACM*, 61(7):56–66, June 2018.

[Hartung and Håkansson, 2007] Ronald Hartung and Anne Håkansson. Automated testing for knowledge based systems. In Bruno Apolloni, RobertJ. Howlett, and Lakhmi Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 4692 of *Lecture Notes in Computer Science*, pages 270–278. Springer Berlin Heidelberg, 2007.

[Hayes and Parzen, 1997] Caroline C. Hayes and Michael I. Parzen. Quem: An achievement test for knowledge-based systems. *IEEE Transactions on Knowledge and Data Engineering*, 9(6):838–847, November/December 1997.

[Henriksson *et al.*, 2018] Jens Henriksson, Markus Borg, and Cristofer Englund. Automotive safety and machine learning: initial results from a study on how to adapt the iso 26262 safety standard. In *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, May 2018. DOI: 10.1145/3194085.3194090.

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

[Klück *et al.*, 2018] Florian Klück, Yihao Li, Mihai Nica, Jianbo Tao, and Franz Wotawa. Using ontologies for test suites generation for automated and autonomous driving functions. In *In Proc. of the 29th IEEE International Symposium on Software Reliability Engineering (ISSRE2018) - Industrial Track*, 2018.

[Koopman and Wagner, 2016] Philip Koopman and Michael Wagner. Challenges in autonomous vehicle testing and validation. *SAE Int. J. Trans. Safety*, 4:15–24, 04 2016.

[Koroglu and Wotawa, 2019] Yavuz Koroglu and Franz Wotawa. Fully automated compiler testing of a reasoning engine via mutated grammar fuzzing. In *In Proc. of the 14th IEEE/ACM International Workshop on Automation of Software Test (AST)*, Montreal, Canada, 27th May 2019.

[Kuhn *et al.*, 2009] D.R. Kuhn, R.N. Kacker, Y. Lei, and J. Hunter. Combinatorial software testing. *Computer*, pages 94–96, August 2009.

[Kuhn *et al.*, 2012] D. R. Kuhn, R. N. Kacker, and Y. Lei. Combinatorial testing. In Phillip A. Laplante, editor, *Encyclopedia of Software Engineering*. Taylor & Francis, 2012.

[Kuhn *et al.*, 2015] D. Richard Kuhn, Renee Bryce, Feng Duan, Laleh Sh. Ghandehari, Yu Lei, and Raghu N. Kacker. Combinatorial testing: Theory and practice. In *Advances in Computers*, volume 99, pages 1–66. 2015.

[Ma *et al.*, 2018a] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 120–131. ACM, 2018.

[Ma *et al.*, 2018b] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, pages 100–111. IEEE, 2018.

[Ma *et al.*, 2018c] Lei Ma, Fuyuan Zhang, Minhui Xue, Bo Li, Yang Liu, Jianjun Zhao, and Yadong Wang. Combinatorial testing for deep learning systems. *arXiv preprint arXiv:1806.07723*, 2018.

[Menzel *et al.*, 2018] Till Menzel, Gerrit Bagschik, and Markus Maurer. Scenarios for development, test and validation of automated vehicles. In *arXiv:1801.08598*, 2018. appeared in Proc. of the IEEE Intelligent Vehicles Symposium.

[Murrell and Plant, 1997] Stephen Murrell and Robert T. Plant. A survey of tools for the validation and verification of knowledge-based systems: 1985-1995. *Decision Support Systems*, 21(4):307–323, 1997.

[Pei *et al.*, 2017] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18. ACM, 2017.

[Plant, 1991] Robert Plant. Rigorous approach to the development of knowledge-based systems. *Knowl.-Based Syst.*, 4(4):186–196, 1991.

[Plant, 1992] Robert T. Plant. Expert system development and testing: A knowledge engineer's perspective. *Journal of Systems and Software*, 19(2):141–146, 1992.

[Schieferdecker, 2012] Ina Schieferdecker. Model-based testing. *IEEE Software*, 29(1):14–18, Jan/Feb 2012.

[Schuldt *et al.*, 2018] Fabian Schuldt, Andreas Reschka, and Markus Maurer. A method for an efficient, systematic test case generation for advanced driver assistance systems in virtual environments. In Hermann Winner, Gunter Prokop, and Markus Maurer, editors, *Automotive Systems Engineering II*. Springer International Publishing AG, 2018.

[Seshia and Sadigh, 2016] Sanjit A. Seshia and Dorsa Sadigh. Towards verified artificial intelligence. *CoRR*, abs/1606.08514, 2016.

[Sotiropoulos *et al.*, 2017] T. Sotiropoulos, H. Waeselynck, J. Guiochet, and F. Ingrand. Can robot navigation bugs be found in simulation? an exploratory study. In *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 150–159, July 2017.

[Su *et al.*, 2019] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2019.

[Sun *et al.*, 2018] Youcheng Sun, Xiaowei Huang, and Daniel Kroening. Testing deep neural networks. *arXiv preprint arXiv:1803.04792*, 2018.

[Tao *et al.*, 2019] Jianbo Tao, Yihao Li, Franz Wotawa, Hermann Felbinger, and Mihai Nica. On the industrial application of combinatorial testing for autonomous driving functions. In *Proceedings of the International Workshop on Combinatorial Testing (IWCT)*. IEEE, 2019.

[Tiihonen *et al.*, 2002] Juha Tiihonen, Timo Soininen, Ilkka Niemelä, and Reijo Sulonen. Empirical testing of a weight constraint rule based configurator. In *Proceedings of the ECAI 2002 Configuration Workshop*, pages 17–22, 2002.

[Wicker *et al.*, 2018] Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. Feature-guided black-box safety testing of deep neural networks. In Dirk Beyer and Marieke Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I*, volume 10805 of *Lecture Notes in Computer Science*, pages 408–426. Springer, 2018.

[Wotawa and Li, 2018] Franz Wotawa and Yihao Li. From ontologies to input models for combinatorial testing. In Inmaculada Medina-Bulo, Mercedes G. Merayo, and Robert Hierons, editors, *Testing Software and Systems*, pages 155–170. Springer International Publishing, 2018.

[Wotawa and Pill, 2014] Franz Wotawa and Ingo Pill. Testing configuration knowledge-bases. In *Proceedings of the 16th Workshop on Configuration*, Novi Sad, Serbia, September 2014.

[Wotawa *et al.*, 2018] Franz Wotawa, Bernhard Peischl, Florian Klück, and Mihai Nica. Quality assurance methodologies for automated driving. *Elektrotechnik & Informationstechnik*, 135(4–5), 2018. https://doi.org/10.1007/s00502-018-0630-7.

[Wotawa, 2016a] Franz Wotawa. Testing autonomous and highly configurable systems: Challenges and feasible solutions. In D. Watzenig and M. Horn, editors, *Automated Driving*. Springer International Publishing, 2016. DOI 10.1007/978-3-319-31895-0 22.

[Wotawa, 2016b] Franz Wotawa. Testing self-adaptive systems using fault injection and combinatorial testing. In *Proceedings of the Intl. Workshop on Verification and Validation of Adaptive Systems (VVASS 2016)*, Vienna, Austria, 2016.

[Wotawa, 2018a] Franz Wotawa. Combining combinatorial testing and metamorphic testing for testing a logic-based non-monotonic reasoning system. In *In Proceedings of the 7th International Workshop on Combinatorial Testing (IWCT) / ICST 2018*, April 13th 2018.

[Wotawa, 2018b] Franz Wotawa. On the automation of testing a logic-based diagnosis system. In *In Proceedings of the 13th International Workshop on Testing: Academia-Industry Collaboration, Practice and Research Techniques (TAIC PART) / ICST 2018*, April 9th 2018.

[Xiong *et al.*, 2013] Zhitao Xiong, Hamish Jamson, Anthony G. Cohn, and Oliver Carsten. *Ontology for Scenario Orchestration (OSO): A Standardised Scenario Description in Driving Simulation*. ASCE Library, 2013.