Assuring the Safety of Advanced Driver Assistance Systems through a Combination of Simulation and Runtime Monitoring

Malte Mauritz, Falk Howar, and Andreas Rausch

Institute for Applied Software Systems Engineering (IPSSE)
Clausthal University of Technology
Clausthal, Germany
{malte.mauritz|falk.howar|andreas.rausch}@tu-clausthal.de

Abstract. Autonomous vehicles will share the road with human drivers within the next couple of years. One of the big open challenges is the lack of established and cost-efficient approaches for assuring the safety of Advanced Driver Assistance Systems and autonomous driving. Product liability regulations impose high standards on manufacturers regarding the safe operation of such systems. Today's conventional engineering methods are not adequate for providing such guarantees in a cost-efficient way. One strategy for reducing the cost of quality assurance is transferring a significant part of the testing effort from road tests to (system-level) simulations. It is not clear, however, how results obtained from simulations transfer to the road. In this paper, we present a method for ensuring that an Advanced Driver Assistance System satisfies its safety requirements at runtime and operates within safe limits that were tested in simulations. Our approach utilizes runtime monitors that are generated from safety requirements and trained using simulated test cases. We evaluate our approach using an industrial prototype of a lane change assistant and data recorded in road tests on German highways.

Keywords: Advanced Driving Assistance Systems, Lane Change Assistant, Simulation-based Testing, Runtime Verification

1 Introduction

The automotive industry is heading towards autonomous vehicles that are expected to share the road with human drivers within the next couple of years (cf.[16]). Today, Adaptive Cruise Control (ACC) systems already take over longitudinal control and parking assistants take over lateral control in cars. With the release of its autopilot in 2015, Tesla Motors has rolled out a system that controls lateral and longitudinal movement of its cars autonomously on highways. All major car manufacturers, and even companies like Google, are working on sophisticated Advanced Driver Assistance Systems (ADAS) [23].

One of the big open challenges is the current lack of established and cost-efficient approaches for assuring the safety of such driving assistants [25]. Regulatory authorities require such systems to meet highest standards for ensuring road safety. The product and producer liability (e.g., in Germany: ProdHaftG §1, BGB §823 I, BGB §433) oblige manufacturers to ensure that ADAS safely operate in their highly dynamic environments and to eliminate harm for drivers, vehicles, and any persons or objects in their environments.

Today's conventional engineering methods are not adequate for providing such guarantees in a cost-efficient way: Common vehicle field tests are too expensive; they require too many miles to be driven in order to demonstrate that a system is sufficiently safe (cf. [25]). One strategy for reducing the cost of quality assurance is transferring a significant part of the testing effort from road tests to (system-level) simulations. Two challenges have to be addressed in order to use results obtained in simulations for assuring the safety of ADAS on the road (cf. [2]); First, it has to be ensured that the behavior of the tested system is comparable on the road and in simulations. Second, it has to be ensured that simulations cover relevant and realistic driving situations.

In this paper, we report the results of a research project that conducted initial research on novel approaches for combining simulations and road tests for ensuring the safety of ADAS. We have developed a framework for combining a pair of runtime monitors that record relevant driving situations and ensure that an ADAS operates within specified and tested limits. In system-level simulations, one monitor checks functional correctness, the other one observes and learns new driving situations. Together, these two monitors allow to test and verify the behavior of an ADAS. During road tests or operation the ADAS can then be monitored in order to identify unsafe or untested operation conditions and behavior. These conditions and behavior can then be transferred back to the simulation. We evaluate our method using an industrial prototype of a lane change assistant (LCA), and data recorded in road tests on national highways in Germany.

One of the main technical challenges of this approach is that the monitors have to abstract from concrete tested driving scenarios in order for our approach to become effective — otherwise testing will never cover a significant fraction of relevant driving situations. On the other hand, an abstraction cannot be too coarse. Otherwise unsafe concrete driving situations might be identified as tested. We show how we derived such an abstraction from safety requirements.

We have presented the research project in which we have developed our approach in [14] and reported preliminary findings from a small case study that we implemented in Java in a workshop paper [13]. In this paper, we present the complete approach and report on the results of an evaluation with the LCA.

Outline. The remainder of this paper is organized as follows. In Section 2, we describe the basic concepts and architecture of our runtime monitoring solution. Section 3 describes the abstraction of traffic situations used in our case study — a lane change assistant (LCA). In Section 4, we present results from the evaluation of our approach. We conclude and discuss future work in Section 6.

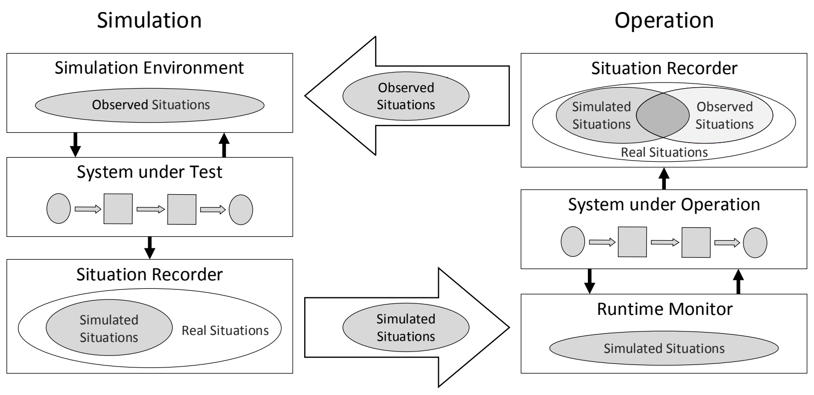


Fig. 1: Bridging the gap between simulation and road tests by (a) training driving situations in simulations, and (b) monitoring and recording driving situations on the road.

2 Core Concept and Architecture

In this section, we present our approach for transferring results obtained from simulations to the road. We ensure that the ADAS behaves consistently in both worlds, and that realistic driving conditions are simulated, by utilizing a set of runtime monitors.

Advanced Driver Assistance Systems. Modern ADAS consist of many components and often multiple assistants have to be coordinated. We group those components following the Input-Process-Output (IPO) pattern, where the environment is perceived by the vehicle's sensors and then preprocessed for a consistent view of the vehicle's environment (Input). Based on this internal view of the environmental situation the main function of the ADAS computes the necessary actions (Process). Finally, the computed actions are post-processed, coordinated, and transformed into commands for actuators, e.g., the braking system or the engine (Output).

Our work addresses the safety of the main function (IPO-Part: Process) of an ADAS by combining the runtime monitoring in simulations and at operation. Safety assurance for sensors, sensor fusion, and for actuators are complex challenges in their own domain. Please note that in this work we do not address the on-line selection and execution of appropriate counter measures but only evaluate if it is safe to operate the ADAS in the current situation.

We use a lane change assistant (LCA) as running example and as a basis for our evaluation (cf. Section 3). A LCA is an automated driving function that controls a vehicle on highways and performs lane changes autonomously. The main function of the LCA receives *traffic situations* as input, i.e., a map, containing roads with lanes, and objects with positions and velocities in the vehicles vicinity. Based on this map, the function computes a *target point*, i.e., a geo-spatial point that should be incorporated in the future trajectory of the controlled vehicle (the so-called *ego vehicle*).

Relating Simulation and Road Tests. One of the challenges when verifying the correctness and safety of ADAS arises from the complex environments these systems operate in: Field tests will never cover the uncountable number of situations an ADAS may encounter on the road. In order for simulations to become efficient substitutes for field tests, simulations have to model realistic traffic situations. Our approach for transferring information between simulations and road tests is sketched in Figure 1: Initially, a set of test cases is executed in a system-level simulation (left half of the Figure 1). In these simulations, sequences of traffic situations are recorded and the behavior of the ADAS is verified using test oracles generated from safety requirements. The simulated and verified traffic situations are recorded in a database. This database is used by a runtime monitor to check if an observed situation has been tested. During operation (cf. right side of the Figure 1), we check the ADAS with this monitor[1]. New situations encountered during operation are used to enhance the set of test cases (derived from sequences of traffic situations) for the ADAS in further simulations.

In order for the approach to be efficient, it is essential to abstract from concrete traffic situations. Figure 2 details this: A concrete traffic situation $c \in C$ consist of a set of objects with attributes from infinite domains, e.g., precise geo-spatial positions and velocities. It is thus impossible to simulate and record all real concrete situations. We introduce *abstract traffic situations* that abstract the position and other properties of a concrete object, to, e.g., *behind the ego vehicle on the lane to the left*. An abstract traffic situation $a \in A$ is a set of objects with predicates as indicated in the upper left of the figure. Due to the limited range of a cars sensors, we can assume that the number of objects in concrete and abstract traffic situations is finite.

Table 1: Combination of two monitors checking if traffic situations are deemed safe during operation, i.e., if encountered situations have been tested and if the ADAS behaves correctly.

Tested	Correct	Safe
Yes	Yes	Yes
Yes	No	No
No	Yes	No
No	No	No

When abstracting concrete traffic situations to abstract ones, it is important that the abstraction $A_I : C \mapsto A$ is not too coarse. Otherwise one abstract traffic situation could correspond to safe and unsafe concrete traffic situations. We ensure this as follows. We derive the predicates for our abstraction (e.g., *behind the ego vehicle*) from the safety-critical requirements of the system (for details cf. Section 3). This allows us to generate monitors for the functional requirements at the level of abstract traffic situations. These monitors serve

[1] The development environment in our industrial case study allows running identical components in simulations and on the real road.

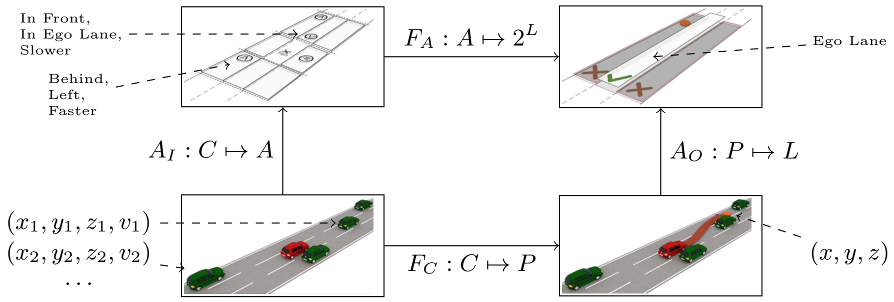


Fig. 2: A pair of abstractions A_I and A_O and an abstract function F_A are derived from safety requirements and enable the monitoring of abstract driving situations.

two purposes. First, they are used as test oracles in simulations. Secondly, the monitors are used during road tests or operation to monitor if the abstraction is fine enough: We test the behavior of the ADAS for one concrete instance of an abstract (i.e., specified) situation in a simulation. The ADAS is supposed to behave identically (at the abstract level) for all other concrete instances of the same abstract situation. The monitors check this during operation (as shown in Table 1).

Monitoring Architecture. The architecture of our runtime monitoring consists of two layers as shown in Figure 3. The lower layer is the ADAS itself. We depict the ADAS following the Input-Process-Output (IPO) pattern. The upper layer depicts our runtime monitoring addressing the monitoring of the correctness of the system's behavior and of tested environmental situations. The architecture has two important features: First, monitoring components simply use the existing interfaces between components of the ADAS. Monitoring can easily be added to existing functions without compromising other assurance efforts. Second, identical components are used for simulation and operation. The only difference is the initialization of the abstract situation database.

The main function F_C of the ADAS takes concrete traffic situations from C as input and computes target points $p \in P$. The correctness of the main function of the ADAS is monitored by the *abstract function* F_A at the level of abstract traffic situations. The abstract function $F_A : A \mapsto 2^L$ computes a set of actions (in our concrete example: lane change operations) that the ADAS may perform in the current situation. The set of actions is used by the *conformance monitor* for the evaluation of the concrete action taken by the ADAS. The concrete action of the ADAS is abstracted to the level of the abstract function by function $A_O : P \mapsto L$. For a lane change, e.g., the concrete target point is abstracted to a target lane. The conformance monitor evaluates if the abstracted action of the ADAS is in the set of safe actions processed by the abstract function, i.e., if $A_O(F_C(x)) \in F_A(A_I(x))$ for $x \in C$ (cf. also Figure 2).

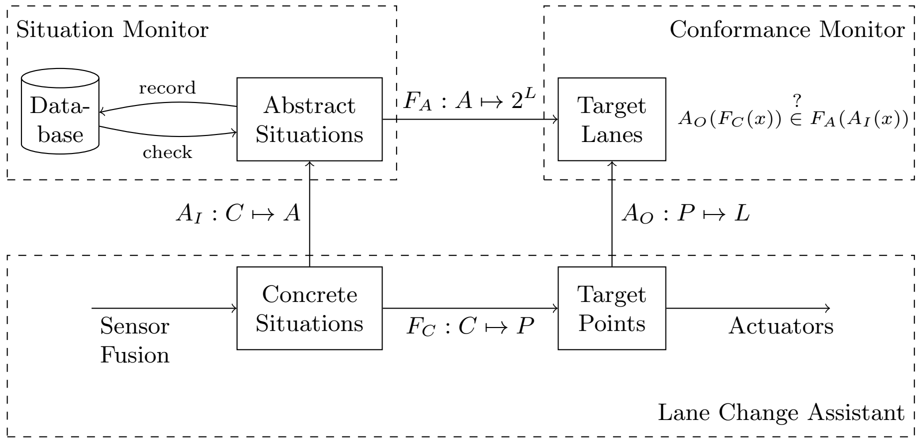


Fig. 3: Runtime Monitoring Architecture.

A second set of components monitors the encountered environmental situations. The *situation monitor* (upper left of Figure 3) records encountered abstract traffic situations. For each encountered situation, the situation monitor evaluates at runtime if the encountered abstract traffic situation has been tested in simulations. We use a canonical representation of abstract traffic situations and identify situations that are equivalent up to names of objects as equal. We refer to these canonic abstract situations as *unique* situations.

Combining the verdicts of monitoring of the conformance of the ADAS and familiarity of environmental situations, we can deduce a verdict about the safety of the ADAS. In case an observed situation has not yet been tested in simulations or if the ADAS does violate its requirements, operation of the ADAS is not safe, as detailed in Table 1.

3 Case Study: The Lane Change Assistant

In this section we describe how we have implemented the approach discussed in the previous section for testing the prototype of an industrial lane change assistant (LCA). We start by presenting a brief overview of how the lane change assistant operates on concrete traffic situations. We then describe how the *abstraction* (A_I, A_O) was derived from a set of functional safety requirements and how monitors for these requirements constitute the *abstract function* F_A used for monitoring the LCA.

LCA and Abstract Domain. As sketched in Figure 2, the LCA operates on *concrete traffic situations*. A concrete traffic situation is a geo-spatial map of the ego vehicle's surroundings, containing information produced by environment recognition (i.e., sensing and sensor fusion) about driving lanes and other vehicles and objects. Two concrete traffic situations may differ only in the distance of

Table 2: Properties of other vehicles derived from safety requirements of LCA.

Object	Property	Abstract Domain
Ego Vehicle	Velocity	{Sufficient; Insufficient}
	Lane Center Offset	{Sufficient; Insufficient}
Vehicle	Lane Position	{Left_Neighbor; Ego; Right_Neighbor}
	Relative Velocity	{Higher; Lower}
	Relative Position	{Front; Next; Behind}
	Distance to Ego	{Sufficient; Insufficient}

one surrounding vehicle (and by as little as a couple of inches). The LCA (F_C) computes a *concrete target point* (marked in orange in the lower right corner of Figure 2). This target point is an actual point in space that is processed together with target points of other assistants (e.g., an Advanced Cruise Control) into the final future trajectory of the ego vehicle.

As described in the previous section, abstraction A_I maps concrete traffic situations to *abstract traffic situations*. The abstraction A_O maps concrete target points to *target lanes*. The abstract function F_A computes the set of *target lanes* allowed for the lane change assistant in the current abstract situation. The LCA operates in safe limits as long as the abstracted concrete target point is in the set of allowed target lanes.

Abstraction and Abstract Function. Abstraction and abstract function are derived from a set of requirements that were provided for the LCA: From a total of 58 requirements (functional and non-functional), we restricted our attention to the 17 safety-relevant requirements restricting the performance of lane changes based on position and behavior of objects in the vicinity of the ego vehicle. These 17 requirements were decomposed and formalized into abstraction and abstract function.

We used a pattern based analysis to derive formal predicates and formulas from requirements based on their structure (following the ideas of [10]). We exemplify the steps and result of the analysis the specific requirement;

"The system shall be able to handle fast objects approaching the ego vehicle from behind with at least 10 m/s relative velocity".

In a first step, we discussed the requirements with the developers of the LCA and replaced imprecise language in requirements by more precise expressions. In this case we specified "to handle" to mean "not to perform a lane change" since it is deemed unsafe to perform a lane change into a lane if a vehicle on this lane approaches the ego vehicle from behind with a higher relative velocity.

In a second step, we reformulated requirements in a conditional structure consisting of an activity and a condition under which the activity is prohibited, resulting in properties like

"No lane change to right lane if vehicle on right lane behind ego vehicle with more than 10 m/s relative velocity".

Abstraction. We derive the abstraction from the conditional parts of the requirements. We are interested in relations between objects. In our example relations are: *(vehicle) behind the ego vehicle*, and *(vehicle's) relative velocity is more than 10 m/s*, and *vehicle on right lane*. In each relation there is one subject (the vehicle in this case). We derive properties of subjects and abstract domains for those properties from the relations: This, e.g., yields *relative position* as a property of other vehicles and *behind* as one of its abstract values. Table 2 shows the properties and corresponding abstract values for the ego vehicle and other vehicles.

Overall, we defined 14 properties for ego vehicle, other vehicles, lanes, the general environment, and the human driver from the 17 analyzed safety requirements. A concrete traffic situation is abstracted by keeping all objects (lanes, vehicles, etc.) and by computing the abstract values of their properties.

Abstract Function. The conditional parts of the 17 safety requirements can be implemented as checks on ego vehicle, lanes, other vehicles, and their properties in abstract traffic situations. The abstract function evaluates these checks for each road lane and surrounding object. For each unsatisfied check, the corresponding lane is excluded from the set of valid target lanes. As result remains the set of lanes the vehicle may safely change to.

For this work, we analyzed requirements that are formulated in natural language and manually implemented abstractions and abstract function. We envision that this process can be automated by using formal specification languages (e.g., Alloy [9] or Z [21]).

From Simulations to the Road and Back. The ultimate goal of this work is to bridge the gap between simulations and road tests as is shown in Figure 1.

Simulation to Road. In our case study with the LCA, we used the described abstractions and abstract function to record abstract traffic situations during simulations. The simulation was driven by manually designed test scenarios. These scenarios were designed to test the conformance of the LCA to its specification. We then used the database of recorded unique abstract traffic situations as a basis for the situation monitor (cf. Figure 3) when evaluating the LCA in road tests.

Road to Simulation. We extracted abstract situations from data recorded during road tests and used these situations as a basis for defining new test cases that were then replayed in simulations. Please note, that this is not trivial: The simulator computes concrete vehicle trajectories. These have to be generated from abstract traffic situations, which only contain relational information (e.g., behind, slower, left, etc.). For this work, we manually designed test cases that simulate a sequence of abstract traffic situations. This was sufficient for an initial evaluation of the loop between simulation and road tests.

Table 3: Coverage of traffic situations after training the situation monitor.

Experiment	Duration [Min:Sec]	Situations [#] Sum	Situations [#] Unique	Tested [%] Correct	Tested [%] Incorr.	Not Tested [%] Correct	Not Tested [%] Incorr.
Simulated (1)	10:38	17,879	518	16.74	0.0	82.63	0.63
Simulated (2)	10:37	18,419	567	14.92	0.0	85.07	0.01
Road (3-lane, A2)	5:53	7,078	1,078	0.68	0.0	96.62	2.70
Road (2-lane, A39)	6:34	7,885	974	1.57	0.0	96.45	1.98

In a next step we plan to generate models from the recorded sequences of abstract traffic situations and use these models as a basis for generating realistic traffic situations in simulations. We plan to use automata learning algorithms (e.g. [12]) for generating probabilistic abstract models.

4 Evaluation

We have evaluated our approach on a prototype of a LCA that is being developed as part of an industrial highway autopilot. Our partner's development[2]- and simulation[3]- environment allows running the same LCA components in simulations and on the road. Since our monitors only listen to situations and target points on existing interfaces, it possible to use recorded data in our evaluation. We use two sets of data as a basis for our evaluation.

Simulated. Data recorded while the LCA operates in randomly generated traffic on two tracks in the simulation environment.

Road. Data recorded in two road tests with the LCA on German highways A2 and A39; A2 is a three lane highway, A39 is a two lane highway.

Using the simulation environment, we can replay these recordings with our monitoring components in the loop. During replay, the situation monitor operates in *checking mode* in which observed abstract traffic situations are compared to situations in the database (as detailed in Section 2). The database is initialized with data recorded while simulating the LCA in 23 manually developed test cases. These test cases were designed by project partners to test the conformance of the LCA to its specification.

We use our monitoring components to compute some statistics over the abstract traffic situations on the individual recordings (e.g., frequencies and distribution).

We evaluate the following three conjectures.

H1. *The abstraction generated from the functional requirements is not too coarse.* We evaluate this hypothesis by analyzing abstract traffic situations that are classified as incorrect when replaying the *Simulated* and *Road* data sets.

[2] ADTF - http://www.elektrobit.com/products/eb-assist/adtf/
[3] Virtual Test Drive (VTD) - http://www.vires.com/products.html

H2. *Randomly simulated and manually tested traffic situations are not realistic in the simulation environment that is used for validating the LCA.* We evaluate this hypothesis by comparing frequencies and distribution of abstract traffic situations in the *Simulated* and *Road* data sets.

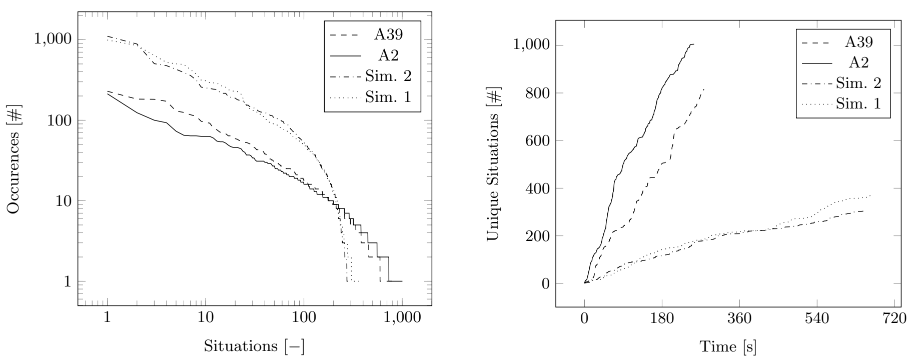
H3. *It is possible to achieve satisfiable coverage of situations when training monitors with realistic traffic.* We evaluate this hypothesis by initializing the situation monitor using increasing sets of realistic test cases and then replaying the *Road (A2)* data set while analyzing the performance of the situation monitor.

H1: Coarse Abstraction. Table 3 reports duration, numbers of total and unique abstract traffic situations, and coverage results for the *Simulated* and *Road* (A2 and A39) data set. In the *Simulated* data set, approximately 18,000 situations are observed. The number of unique abstract traffic situations is between 500 and 600 for both experiments. The percentage of safe (i.e., tested and correct) abstract traffic situations ranges from 14.92% to 16.74%. The vast majority of abstract situations (82.63% and 85.07%) is untested while the LCA conforms to its safety-critical requirements. Only 0% to 0.63% of abstract traffic situations were untested while the LCA violates its safety-critical requirements.

For the *Road* data set, the recordings each contain data for about six minutes. The more complex abstract traffic situations recorded by the situation monitor lead to an absolute increase in the number of unique abstract traffic situations (in comparison to the *Simulated* data set) to around 1,000. The percentage of tested abstract traffic situations drops to around 1% for these experiments. Around 96% of all observed abstract traffic situations are untested while the LCA performs within the limits of the functional requirements. The fraction of situations in which the LCA violates the functional requirements in untested abstract traffic situations increases to 1.98% and 2.70%.

We did not encounter any tested abstract traffic situations in which the LCA does not conform to its safety-critical requirements. This indicates that the used abstraction is adequate. We have manually analyzed the untested situations in which the LCA apparently does not conform to the specification. We did not find any indication of the abstraction being too coarse for these cases but rather discovered that the implementation on the LCA actually violates the functional requirements by performing a lane change to the right onto a lane with a slower vehicle ahead of the ego vehicle.

H2: Realistic Traffic. As an indicator for the complexity of traffic we analyze the rate at which new unique abstract traffic situations are discovered over time as well as the total of unique abstract traffic situations in one experiment. Figure 4a shows the number of occurrences for each unique abstract traffic situation. Both axes are scaled logarithmically. The *Simulated* series exhibit a similar distribution. The number of occurrences is distributed exponentially: there are many abstract traffic situations with few occurrences and only few abstract traffic situations with many occurrences. For the *Road* data set, the exponential distribution of occurring situations is more pronounced than for *Simulated* data



(a) Distribution of frequencies. (b) New situations over time.

Fig. 4: Unique abstract traffic situations and: Distribution of frequencies and discovery of new situations over time.

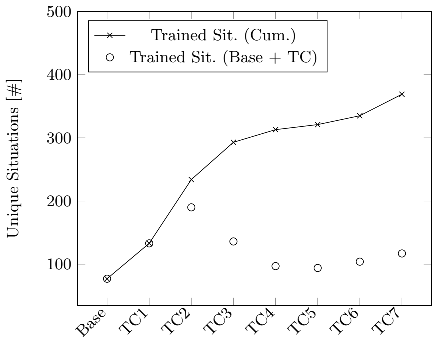
set. This indicates that the exponential distribution becomes less expressed for longer experiments when a representative sample of situations is observed.

Figure 4b shows how the number of observed unique abstract traffic situations evolves over time in the experiments. The rates are significantly lower in the *Simulated* data set than the rates for the *Road* data set. On the two lane highway (A39) new situations occur at a slightly lower rate than on the three lane highway (A2). This indicates that the traffic generated by the simulation environment is less complex than real traffic on German highways and hence not realistic enough for testing the LCA in simulations sufficiently. Though we were not able to observe this in our experiments due to their limited duration, we expect that in longer data sets the increase in new situations to tail off at some point.

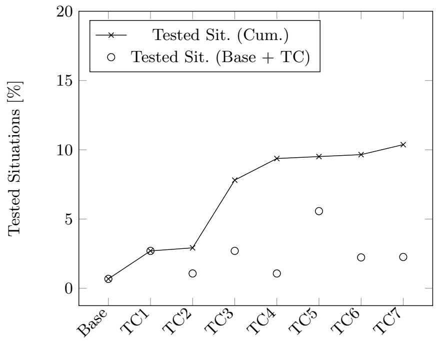
Additionally, the data presented in Table 3 suggests that test cases derived from the system's requirements are simpler than real traffic (and rightfully so: those tests are minimal scenarios for testing certain behavior of the LCA). Trained with these tests, the situation monitor classifies as little as $0,69\%$ (resp. 1.57%) of abstract traffic situations from the *Road* data set as tested.

H3: Situation Coverage. The initial set of test cases does not initialize the database with realistic traffic situations as the test cases model situations with few vehicles. In order to investigate the impact of training the situation monitor with realistic traffic situations, we update the database of known situations in this experiment. We derived seven additional test cases from the observed but untested abstract traffic situations of the recording on the highway A2. We selected different adjoint sequences of abstract traffic situations from the recorded data and used these sequences as a reference for manually modeling test cases.

Figure 5a displays the increase in trained unique abstract traffic situations that is achieved by re-training the situation monitor in simulations with the additional test cases (TC 1 to TC 7) — compared to the baseline of only training the monitor with data recorded from simulating the initial test cases (cf. the



(a) Simulated unique abstract situations.



(b) Coverage of observed situations.

Fig. 5: Coverage after training monitors with additional test cases generated from observed situations.

test case data set). The figure compares the cumulative increase (line) with the increase achieved by individual test cases (circles). From 77 tested abstract traffic situations for the initial set of test cases, the seven test cases increased the total number of trained unique abstract traffic situations to 369. The encountered situations in independent simulations of each test case correlate with the increase of the set of trained unique abstract traffic situations of all test cases accumulated.

The graph in Figure 5b shows the changes in the percentage of tested abstract traffic situations for the re-monitoring of the recording on the highway A2 with the trained abstract traffic situations of the additional test cases. The graph compares cumulative percentage (line) with the changes achieved by individual test cases (circles). The cumulated set of 369 trained abstract traffic situations increases the accumulated percentage of tested abstract traffic situations for the recording on the highway A2 from originally 0.67% to 10.37% (cf. Figure 5b).

In comparison to the results for trained abstract traffic situations of the test cases (cf. Figure 5a), the recorded abstract traffic situations of each test case do not directly correspond to the percentage of tested abstract traffic situations at the operation of the LCA. As for the test case TC 2, the accumulated percentage of tested abstract traffic situations does not directly correspond to its large number of trained abstract traffic situations.

Discussion. Our evaluation shows promising results: We were able to find evidence for the effectiveness of the proposed approach. The experiments show that test cases derived from real traffic situations enable a more efficient and realistic simulation of traffic situations. The results also show that state-of-the-art simulation (at least in this particular case) does not generate operation conditions that resemble conditions on the road closely enough.

Finally, our experiments indicate that abstracting concrete traffic situations to the level of the considered requirements is a valid idea. Engineers tend to reason about systems' behavior on an abstract level similar to the level of the requirements. This idea is even more valid in the context of safety-critical functions, where every line of code can be traced to a functional requirement.

5 Related Work

The current trend in the automotive domain - testing autonomous system in simulations has been addressed in multiple publications. Schuldt et al. present in [18] and [19] a modularized virtual test tooling kit. They address the generation of relevant test cases as well as the definition of assessment criteria for systematic test evaluation. In [22], Ulbrich et al. present their approach for testing and validating algorithms for the tactical behavior planning of a LCA. They present the modeling and execution of test scenarios and test cases in closed-loop environments. In [3] and [5], Berger et al. use test scenarios modeled as graph structure for the assessment and validation of vehicle active safety systems in simulations. The graph structures model variations of US NCAP and EuroNCAP scenarios. In [4], the authors enhance the evaluation of the test results by introducing tolerance ranges. Olivares et al. use in [17] a stochastic approach for the construction of test scenarios with respect to relevant parameter combinations, which might have been omitted by manually defined sets of scenarios. Zofka et al. employ in [26] parameterizable simulation models to define relevant test scenarios. None of the approaches considers real road traffic as input for the generation of test cases. Without consideration of real traffic, it is unlikely that an ADAS is tested in realistic situations.

For the validation of ADAS, several authors have proposed approaches combining field tests with simulations. In [6], Bock describes the use of augmented reality to inject virtual obstacles into the view of human drivers. In [20], Sefati et al. describe the integration of scientific objects into the object recognition of the ADAS for testing purposes. Both approaches are only used in field tests and do no benefit from the performance and reproducibility of simulations.

Other authors use data from field tests as input for simulations and testing. Lages et al. describe in [11] the semi-automatic generation of test scenarios for Software-in-the-Loop tests from recorded real world drivings. Bach et al. describe in [1] the problem of inadequate domains of the recorded test data for usage in virtual verification. They transform the test data into coherent stimuli for the system under test (SUT) by identifying dependencies between the input data streams of the SUT. Wachenfeld et al. present in [24] a runtime validation method called Virtual Assessment of Automation in Field Operation (VAAFO). VAAFO accesses automation's safety and identifies relevant test cases for closed-loop simulations by simulating the ADAS and its world alongside the human driver in the vehicle at operation.

None of these approaches generate data for new test cases at the abstract level of a specification. Neither does any approach transfer results of the simulation back to the road. Our approach introduces a comprehensive methodology to iteratively improve the safety of ADAS by using results from the road in the simulation and also results from simulations on the road.

6 Conclusion and Future Work

We have presented a method for ensuring that an advanced driver assistance system satisfies its safety requirements at runtime and operates within safe limits that were tested in simulations. Our main technical contribution is a conceptual framework for establishing a relation between simulations and road tests using a set of monitoring components for computing, checking, and learning relevant abstract traffic situations. We have evaluated our approach using an industrial prototype of a LCA and data recorded in road tests on German highways. We were able to identify relevant traffic situations in road tests, train monitors with abstract traffic situations in simulations, and then use those monitors to check tested and correct behavior of the ADAS.

There are many open questions for future research: One obvious question is how monitors perform aboard an actual vehicle under real-time conditions. A second direction of research would be extending our monitoring framework to stateful and time-dependent behavior, e.g., to cover maneuvers that require different phases. We plan to investigate how automata learning (e.g. [12, 15]) can be used for producing stateful models of traffic situations. These stateful models can then become the basis for generating realistic traffic in simulations. Then, for our evaluation, abstraction and abstract function were implemented manually based on the derived properties and abstract values. We imagine that abstraction and abstract function could be generated from requirements in future applications in an automated fashion when using a formal specification language (e.g., Alloy [9] or Z [21]). In this work we did not address the refinement of a chosen abstraction that may become necessary. We plan to investigate if counterexample-guided abstraction refinement (CEGAR) can be applied in our setting [7]. Some automata learning algorithms already incorporate abstraction refinement [8]. Using such an algorithm would allow us to integrate model generation and abstraction refinement.

Finally, the ultimate goal is the integration of safety monitors into the planning of autonomous behavior of higher-level functions.

References

1. J. Bach, K.-L. Bauer, M. Holzpfel, M. Hillenbrand, and E. Sax. Control based driving assistant functions test using recorded in field data. In *Proc. 7. Tagung Fahrerassistenzsysteme*, 2015.
2. C. Berger. From autonomous vehicles to safer cars: selected challenges for the software engineering. *Computer Safety, Reliability, and Security*, 2012.
3. C. Berger, D. Block, S. Heeren, C. Hons, S. Kuhnel, A. Leschke, D. Plotnikov, and B. Rumpe. Simulations on consumer tests: A systematic evaluation approach in an industrial case study. In *ITSC14*, 2014.

4. C. Berger, D. Block, S. Heeren, C. Hons, S. Kühnel, A. Leschke, D. Plotnikov, and B. Rumpe. Simulations on consumer tests: Systematic evaluation of tolerance ranges by model-based generation of simula-tion scenarios. *Proc. Fahrerassistenzsysteme und Integrierte Sicherheit*, 2014.
5. C. Berger, D. Block, S. Heeren, C. Hons, S. Kuhnel, A. Leschke, D. Plotnikov, and B. Rumpe. Simulations on consumer tests: A systematic evaluation approach in an industrial case study. *Intelligent Transportation Systems Magazine, IEEE*, 7(4), 2015.
6. T. Bock. Bewertung von Fahrerassistenzsystemen mittels der Vehicle in the Loop-Simulation. In *Handbuch Fahrerassistenzsysteme*. Vieweg+Teubner Verlag, 2012.
7. E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM (JACM)*, 50(5):752–794, 2003.
8. F. Howar, B. Steffen, and M. Merten. Automata learning with automated alphabet abstraction refinement. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 263–277. Springer, 2011.
9. D. Jackson. *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, 2006.
10. A. Kane. *Runtime Monitoring for Safety-Critical Embedded Systems*. PhD thesis, Carnegie Mellon University, 2015.
11. U. Lages, M. Spencer, and R. Katz. Automatic scenario generation based on laser-scanner reference data and advanced offline processing. In *2013 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, 2013.
12. H. Mao, Y. Chen, M. Jaeger, T. D. Nielsen, K. G. Larsen, and B. Nielsen. Learning probabilistic automata for model checking. In *Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on*, pages 111–120. IEEE, 2011.
13. M. Mauritz, F. Howar, and A. Rausch. From Simulation to Operation : Using Design Time Artifacts to Ensure the Safety of Advanced Driving Assistance Systems at Runtime. *International Workshop on Modelling in Automotive Software Engineering*, 2015.
14. M. Mauritz, A. Rausch, and I. Schaefer. Dependable ADAS by Combining Design Time Testing and Runtime Monitoring. In *FORMS/FORMAT 2014, 10th Int. Symp. on Formal Methods*, pages 28–37, 2014.
15. M. Merten, B. Steffen, F. Howar, and T. Margaria. Next generation learnlib. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 220–223. Springer, 2011.
16. R. Okuda, Y. Kajiwara, and K. Terashima. A survey of technical trend of ADAS and autonomous driving. *Proceedings of Technical Program - 2014 International Symposium on VLSI Technology, Systems and Application, VLSI-TSA 2014*, 2014.
17. S. P. Olivares, N. Rebernik, A. Eichberger, and E. Stadlober. Virtual stochastic testing of advanced driver assistance systems. In *Advanced Microsystems for Automotive Applications 2015*. Springer International Publishing, 2016.
18. F. Schuldt, B. Lichte, M. Maurer, and S. Scholz. Systematische Auswertung von Testfällen für Fahrfunktionen im modularen virtuellen Testbaukasten. In *9. Workshop Fahrerassistenzsysteme*, 2014.
19. F. Schuldt, F. Saust, B. Lichte, and M. Maurer. Effiziente systematische Testgenerierung für Fahrerassistenzsysteme in virtuellen Umgebungen. *AAET2013 - Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel*, 2013.

20. M. Sefati, A. Stoff, and H. Winner. Testing method for autonomous safety functions based on combined steering/braking maneuvers for collision avoidance and mitigation. In *6. Tagung Fahrerassistenz*, 2013.
21. J. M. Spivey and J. Abrial. *The Z notation*. Prentice Hall Hemel Hempstead, 1992.
22. 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 M. Horn and D. Watzenig, editors, *Automated Driving - Safer and more efficient future driving*. Springer International Publishing AG, 2016. (accepted to appear).
23. Verband der Automobilindustrie e.V. Automation: From Driver Assistance Systems to Automated Driving. *VDA Magazine - Automation*, AUTOMATION, 2015.
24. W. Wachenfeld and H. Winner. Virtual assessment of automation in field operation a new runtime validation method. In *10. Workshop Fahrerassistenzsysteme*, 2015.
25. H. Winner. *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*, chapter ADAS, Quo Vadis? Springer International Publishing, 2014.
26. M. R. Zofka, F. Kuhnt, R. Kohlhaas, C. Rist, T. Schamm, and J. M. Zllner. Data-driven simulation and parametrization of traffic scenarios for the development of advanced driver assistance systems. In *18th International Conference on Information Fusion*, 2015.